LANTRONIX®



PFAL Command Reference

FOX3-2G/3G/4G Series
BOLERO40 Series

Part Number PMD-00011
Revision E May 2023

Intellectual Property

© 2023 Lantronix, Inc. All rights reserved. No part of the contents of this publication may be transmitted or reproduced in any form or by any means without the written permission of Lantronix.

Lantronix is a registered trademark of Lantronix, Inc. in the United States and other countries.

Patented: https://www.lantronix.com/legal/patents/. Additional patents pending.

Windows and *Internet Explorer* are registered trademarks of Microsoft Corporation. *Firefox* is a registered trademark of the Mozilla Foundation. *Chrome* is a trademark of Google Inc. All other trademarks and trade names are the property of their respective holders.

Warranty

For details on the Lantronix warranty policy, please go to our web site at https://www.lantronix.com/support/warranty/.

Contacts

Lantronix, Inc.
48 Discovery, Suite 250
Irvine, CA 92618, USA
Toll Free:800-526-8766
Phone: 949-453-3990
Fax: 949-453-3995

Technical Support
Online: www.lantronix.com/technical-support/

Sales Offices
For a current list of our domestic and international sales offices, go to the Lantronix web site at www.lantronix.com/about-us/contact/

Disclaimer

All information contained herein is provided “AS IS.” Lantronix undertakes no obligation to update the information in this publication. Lantronix does not make, and specifically disclaims, all warranties of any kind (express, implied or otherwise) regarding title, non-infringement, fitness, quality, accuracy, completeness, usefulness, suitability or performance of the information provided herein. Lantronix shall have no liability whatsoever to any user for any damages, losses and causes of action (whether in contract or in tort or otherwise) in connection with the user’s access or usage of any of the information or content contained herein. The information and specifications contained in this document are subject to change without notice.

Open Source Software

Some applications are Open Source software licensed under the Berkeley Software Distribution (BSD) license, the GNU General Public License (GPL) as published by the Free Software Foundation (FSF), or the Python Software Foundation (PSF) License Agreement for Python 2.7.3 (Python License). Lantronix grants you no right to receive source code to the Open Source software; however, in some cases, rights and access to source code for certain Open Source software may be available directly from Lantronix' licensors. Your use of each Open Source component or software is subject to the terms of the applicable license. The BSD license is available at http://opensource.org/licenses. The GNU General Public License is available at http://www.gnu.org/licenses/. The Python License is available at http://cmpt165.csil.sfu.ca/Python-Docs/license.html. Your use of each Open Source component or software is subject to the terms of the applicable license.

OPEN SOURCE SOFTWARE IS DISTRIBUTED WITHOUT ANY WARRANTY, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. SEE THE APPLICABLE LICENSE AGREEMENT FOR ADDITIONAL INFORMATION.

You may request a list of the open source components and the licenses that apply to them. Contact your regional Lantronix sales associate. https://www.lantronix.com/about-us/contact/

Revision History

This table lists revisions starting from October 2019.

Date	Revision	Comments
October 2019	A	Firmware version avl_3.3.0_rc15. No new or modified features were added. Added Lantronix document part number, Lantronix logo, branding, contact information, and links.

Date	Revision	Comments
April 2020	B	Improvements and bug fixes can be found in the document "RelNotes_Firmware_AVL_3.5.0_r8.pdf". Starting from firmware version 3.5.0.0, the following features are added/changed. Added/changed/removed PFAL commands (See Chapter 4): New: TCP.Client.SetCertificate TCP.Client.ShowCertificate TCP.Client2.SetCertificate TCP.Client2.ShowCertificate TCP.Client2.Connect TCP.Client2.Disconnect TCP.Client2.State TCP.Client2.Send,<protocols>,<"text"> TCP.Client2.ClearSendBuffer TCP.Client2.FlushSendBuffer TCP.Client2.TxKey=<"key"> TCP.Client2.RxKey=<"key"> SYS.Lua.Start[,<"script.lua">] SYS.Lua.Clear[,<"script.lua">] SYS.Lua.Info[,<"script.lua">] SYS.Lua.Write[,<"script.lua">] CNF.Load,<"/sys/file.txt"> SYS.CAN.CANopen.enable[,<hex_node_id>] SYS.CAN.CANopen.disable SYS.CAN.CANopen.cmd,"<CIA-309-3 gateway command>" SYS.CANB.CANopen.enable[,<hex_node_id>] SYS.CANB.CANopen.disable SYS.CANB.CANopen.cmd,"<CIA-309-3 gateway command>" Changed: Added parameter to MSG.Event[,<interface>] Added/changed/removed configuration parameters (See Chapter 5): New: TCP.CLIENT2.CONNECT=<auto>,<>,<port> TCP.CLIENT2.ALTERNATIVE=<auto>,<ip_addr>,<port>,<> TCP.CLIENT2.SENDMODE=<>[,<bufflevel>] TCP.CLIENT2.PING=<>,<> TCP.CLIENT2.TIMEOUT=<>,<retry_> TCP.CLIENT2.LOGIN.EXT=<" "> TCP.CLIENT2.DNS.TIMEOUT=<timeout> TCP.CLIENT2.CERT=<On|Off|Optional> WHITELIST.ACTIVE=<first>,<last> Changed: Added interface option (TCP.CLIENT2) to DEVICE.COMM.<interface> configuration parameter **Added/changed/removed events and states (See Chapter 6):** New: TCP.Client2.eConnecting TCP.Client2.eConnected TCP.Client2.eDisconnecting TCP.Client2.eDisconnected TCP.Client2.sConnecting TCP.Client2.sConnected TCP.Client2.sDisonnecting TCP.Client2.sDisconnected TCP.Client2.sIdle TCP.Client2.eCO.PDO="pdo <PDOnum><nvals><val1>…<valn>" Changed: GPS.Nav.eCellLocate

Date	Revision	Comments
April 2020, continued	B	**Added/changed/removed dynamic variables (See Chapter 7):** New: &(CO.pdo) &(CanERR) &(TCPClient2Online) &(TCPClient2) &(TCP2Text)
August 2021	C	Improvements and bug fixes can be found in the document "RelNotes_Firmware_AVL_3.14.0_rc5.pdf". Starting from firmware version 3.14.0.0, the following features are added/ changed. Added/changed/removed PFAL commands (See Chapter 4): New: Sys.RUpdate.Init,<type>,<option>,<size>,<sectors>,<config> -- Start FW update TCP.Client.ClearCertificate TCP.Client2.ClearCertificate SYS.Webupdate.ShowCertificate SYS.Webupdate.ClearCertificate SYS.Webupdate.SetCertificate Secured,"0d" Secured,"0e[CF-ID:32 Byte]" Secured,"0f[CF-KEY:32 Byte]" TCP.MQTT.Connect TCP.MQTT.Disconnect TCP.MQTT.State TCP.MQTT.Send,"<topic>@<message>" TCP.MQTT.ClearBuffer TCP.MQTT.GetRootCA TCP.MQTT.SetRootCA TCP.MQTT.GetCertificate TCP.MQTT.SetCertificate TCP.MQTT.GetPrivateKey TCP.MQTT.SetPrivateKey Changed: **Added/changed/removed configuration parameters (See Chapter 5):** New: GPS.HISTORY.SIZE=<SIZE> DBG.EN=<0|1>[,SERIAL|SERIAL1|USB|SDCARD] TCP.SLA=<Mode>[,<ServerIP:Port>] DEVICE.CFID DEVICE.CFKEY MQTT.CLIENT.CONNECT=<auto>,<url>,<port> MQTT.CLIENT.TIMEOUT=<timeout>,<retry_wait> MQTT.CLIENT.SENDMODE=<mode> MQTT.CLIENT.ID=<ThingID> MQTT.CLIENT.PING=<period>,<topic>@<message> MQTT.CLIENT.LASTWILL=<topic>@<message>

Date	Revision	Comments
August 2021, continued	C	Changed: **Added/changed/removed events and states (See Chapter 6):** New: TCP.MQTT.eConnecting TCP.MQTT.eConnected TCP.MQTT.eDisconnecting TCP.MQTT.eDisconnected TCP.MQTT.ePacketSent TCP.MQTT.ePingSent TCP.MQTT.eReceived TCP.MQTT.eBufferEmpty TCP.MQTT.eFlashBufferEmpty TCP.MQTT.sConnecting TCP.MQTT.sConnected TCP.MQTT.sDisconnecting TCP.MQTT.sDisconnected TCP.MQTT.sIdle Changed: **Added/changed/removed dynamic variables (See Chapter 7):** New: **Document changes:** - Lua – Lua updates are available in new application note: Using Lua Scripts Application Note, (APP-0117). Lua reference was removed from the appendix. - Refactored compatibility tables in reference chapters.
January 2023	D	Improvements and bug fixes can be found in the document "RelNotes_Firmware_AVL_3.16.0_rc9.pdf". Starting from firmware version 3.16.0.0, the following features are added/ changed. Added/changed/removed PFAL commands (See Chapter 4): New: TCP.MQTT.ClearCertificate Improvements and bug fixes can be found in the document "RelNotes_Firmware_AVL_3.17.0_rc5.pdf". Starting from firmware version 3.17.0.0, the following features are added/ changed. Added/changed/removed PFAL commands (See Chapter 4): New: SYS.Device.SetSerialID,"<serial>" SYS.ModBus.Enable,<serial>,<baudrate|0> SYS.ModBus.Disable SYS.ModBus.ReadRegister,<slave>:<LE|BE>,<reg>:<fmt> SYS.ModBus.Poll Added/changed/removed Configuration settings (See Chapter 5) New: DEVICE.MODBUS=<serial>,<baudrate|0> MODBUS.POLL=<slave>:<LE|BE>,<reg>:<fmt>:<period>,... Added/changed/removed Events and States (See Chapter 6) New: SYS.NFC.eStart Added/changed/removed Dynamic variables (See Chapter 7) New: &(SerialID) &(ProductionID) &(ModBus:reg[*<factor><[+-]offset>])

Date	Revision	Comments
May 2023	E	Improvements and bug fixes can be found in the document "RelNotes_Firmware_AVL_3.18.0_rc9.pdf". Starting from firmware version 3.18.0.0, the following features are added/ changed. Added/changed/removed PFAL commands (see Chapter 4) Changed: Whitelist extended from 3000 to 5000 Added/changed/removed Configuration Settings (See Chapter 5) New: DEVICE.SERIAL1.MODE485 CF.CLIENT.CONNECT CF.CLIENT.DEVICE_NAME CF.CLIENT.DEVICE_DESCRIPTION CF.CLIENT.STATUS_UPDATE_INTERVAL CF.CLIENT.CONTENT_CHECK_INTERVAL CF.CLIENT.GROUPS_CAP_EXCHANGE_DATA CF.CLIENT.GROUPS_CAP_SELECTION_DATA CF.CLIENT.GROUPS_TELEMETRY_DATA MQTT.CLIENT.USER Changed: DEVICE.SERIAL<port>.BAUDRATE

For the latest revision of this product document, please check our online documentation at www.lantronix.com/support/documentation.

Legacy Revision History

This table lists document revisions before October 2019, ordered from newest to oldest modified date.

Version	Author	Changes	Modified Date
3.2.0.4	F. Beqiri	Added command for the service server (MSG.Send.Service,<protocols>,<"text">)	27/08/2019

Version	Author	Changes	Modified Date
3.2.0.3	F.Beqiri	Document version adapted to the version of firmware release. Improvements and bug fixes can be found in the document "RelNotes_Firmware_AVL_3.2.0_rev6.pdf" - Starting from firmware version 3.2.0_rc39, the following features are added/ changed: - Added/changed/removed PFAL commands: - New command Alarm.Call,<index> - see chapter 4.1.4 - New command Sys.Can.Timeout,<timeout_s> - see chapter 4.2.15.13 - New command Sys.CanB.Timeout,<timeout_s> - see chapter 4.2.16.11 - New command SYS.BLE.ListDev[=<filter>] - see chapter 4.2.19.6 - New command SYS.BLE.ListAdd[=<filter>] - see chapter 4.2.19.7 - New command SYS.BLE.ListRel[=<filter>] - see chapter 4.2.19.8 - New command SYS.NFC.play,"<tone1><tone2>....<tonen>" - see chapter 4.2.21.4 - New command SYS.Lua.Start - see chapter 4.2.18.1 - New command SYS.Lua.Stop - see chapter 4.2.18.2 - New command SYS.Lua.Dump - see chapter 4.2.18.3 - New command SYS.Lua.Lock,<"password"> - see chapter 4.2.18.4 - New command SYS.Lua.Unlock,<"password"> - see chapter 4.2.18.5 - New command SYS.Lua.Dump[,<"password">] - see chapter 4.2.18.6 - New command SYS.Lua.Clear - see chapter 4.2.18.7 - New command SYS.LUA.Event,<id>,<"text"> - see chapter 4.2.18.8 - New command Sys.Can.OBDII.Enable[,<format>] - see chapter 4.2.15.11 - New command Sys.Can.DTCO.FMS.<mode> - see chapter 4.2.15.15 - New command Sys.Can.DTCO.SendAPDU,<TA>,"bytes" - see chapter 4.2.15.16 - New command Sys.CanB.OBDII.Enable[,<format>] - see chapter 4.2.16.9 - New command Sys.CanB.DTCO.FMS.<mode> - see chapter 4.2.16.13 - New command GPS.Nav.HoldPosition=<mode> - see chapter 4.5.1.18 - New command WLAN.Connect,<id> - see chapter 4.9.2 - New command Msg.Send.RawFlashBuffer,<protocols>,<"text"> - see chapter 4.10.1.7 - New command Msg.Send.RawTCP,<protocols>,<"text"> - see chapter 4.10.1.10 - New command Msg.Send.RawTCPBuffer,<protocols>,<"text"> - see chapter 4.10.1.12 - New command Msg.ClearBuffer,<port> - see chapter 4.10.2 - New command MSG.Event[,<interface>],<"text"> - see chapter 4.10.4.1 - Added/changed/removed configuration parameters: - Added new parameter DEVICE.CAN.OBD.STARTUP=<format>,<port> - see chapter 5.1.18 - Added new parameter DEVICE.CAN.FMS.STARTUP=<format>,<port> - see chapter 5.1.19 - Added new parameter DEVICE.CAN.STARTUP<port>=<mode>,<baud>,<mode> - see chapter 5.1.20 - Added new parameter DEVICE.CAN.DTCOFMS.STARTUP=<mode>,<port> - see chapter 5.1.21 - Added new parameter DEVICE.CAN.ERR.EVENTS=<mode> - see chapter 5.1.22 - Added new parameter DEVICE.DTCO.D8=B0,<format> - see chapter 5.1.24 - Added new parameter WLAN.NET<id>.SSID=<ssid> - see chapter 5.13.3 - Added new parameter WLAN.NET<id>.TYPE=dhcp - see chapter 5.13.4 - Added new parameter WLAN.NET<id>.TYPE=static,<own_ip>,<netmask>,<gatewayIP>,<dnsIP> - see chapter 5.13.4 - Added new parameter WLAN.NET<id>.PSK=<psk> - see chapter 5.13.5 - Added new parameter WLAN.NET<id>.SECURITY=<type> - see chapter 5.13.6 - Added new parameter WLAN.NET<id>.IP=<ip> - see chapter 5.13.7 - Added new parameter WLAN.NET<id>.PORT=<port> - see chapter 5.13.8 - Added new parameter WLAN.RSSIMIN - see chapter 5.13.9 - Added new parameter WLAN.MODE=<mode>- see chapter 5.13.1	04/07/2019

Version	Author	Changes	Modified Date
		- Added/changed/removed events and states: - New event Can.error=<cond>[,<cond>] - see chapter 6.1.4 - New event Can.eStat=<mode> - see chapter 6.1.4 - New event Canb.eStat=<mode> - see chapter 6.1.5 - New event for ContiPressureCheck™ system - see chapter 6.1.8 - New event SYS.eCan.DTCO.Confirm - see chapter 6.1.9 - New event SYS.eCan.DTCO.Incoming - see chapter 6.1.9 - New event SYS.sCan.DTCO.Confirm - see chapter 6.1.9 - New event SYS.Device.eStart=Reset,Watchdoglobox - see chapter 6.1.10 - Added new parameter DEVICE.CAN.DTCOFMS.STARTUP=<mode>,<port> - see chapter - New event SYS.IoBox.eLost - see chapter 6.1.10 - New state SYS.1Wire.sAvailable=whitelist - see chapter 6.1.19 - New event SYS.WLAN.TCP.ePingSent - see chapter 6.1.22 - New event SYS.Iobox.eReset[=<cond>[...<cond>]] - see chapter 6.1.24 - Added/changed/removed dynamic variables: - New dynamic variable &(RAT) - see chapter 7 - New dynamic variable &(UnixTime2) - see chapter 7 - New dynamic variable &(UserEventText) - see chapter 7 - New dynamic variable &(BLE.ListDev[:<filter>]) - see chapter 7 - New dynamic variable &(BLE.ListAdd[:<filter>]) - see chapter 7 - New dynamic variable &(BLE.ListRel[:<filter>]) - see chapter 7 - New dynamic variable &(BLE.Name[:index]) - see chapter 7 - New dynamic variable &(BLE.MAC[:index]) - see chapter 7 - New dynamic variable &(BLE.UUID[:index]) - see chapter 7 - New dynamic variable &(BLE.Major[:index]) - see chapter 7 - New dynamic variable &(BLE.Minor[:index]) - see chapter 7 - New dynamic variable &(BLE.relName[:index]) - see chapter 7 - New dynamic variable &(BLE.relMAC[:index]) - see chapter 7 - New dynamic variable &(BLE.relUUID[:index]) - see chapter 7 - New dynamic variable &(BLE.relMajor[:index]) - see chapter 7 - New dynamic variable &(BLE.relMinor[:index]) - see chapter 7 - New dynamic variable for ContiPressureCheck™ system - see chapter 7	
3.1.0.2	F.Beqiri	Document version adapted to the version of firmware release. Improvements and bug fixes can be found in the document "RelNotes_Firmware_AVL_3.1.0_rev3.pdf" - Starting from the firmware version 3.1.0_rc33, the following features are added/ changed: - Added/changed/removed configuration parameters: - Added new parameter BLE.SCANDURATION=<ScanDuration> - see chapter 5.14.1 - Added/changed/removed dynamic variables: - New dynamic variable &(BLE.relName) - see chapter 7 - New dynamic variable &(BLE.relMAC) - see chapter 7 - New dynamic variable &(BLE.relUUID) - see chapter 7 - New dynamic variable &(BLE.relMajor) - see chapter 7 - New dynamic variable &(BLE.relMinor) - see chapter 7	30/10/2018

Version	Author	Changes	Modified Date
3.1.0.1	F.Beqiri	Document version adapted to the version of firmware release. - Starting from the firmware version 3.1.0, the following features are added/ changed: - Added/changed/removed PFAL commands: - Increased number of Timers, Triggers, Counters and Macros to 40 - Increased number of global CAN messages to max. 50 (FOX3 with Cortex has 28 / IOBOX-CAN has 14) - Increased number of global CAN variable slots to max. 50 - Increased the buffer size of the command Msg.Send.Serial<index> to 4096 bytes - New parameters for SYS.Device.Reset[,<user_reset_with_reset_time> - see chapter 4.2.4.1 - New command SYS.Device.BackupReset - see chapter 4.2.4.9 - New counter command SYS.Counter<index>.Add=<value> - see chapter 4.2.12.7 - New counter command SYS.Counter<index>.Sub=<value> - see chapter 4.2.12.8 - New command SYS.BLE.ClearList - see chapter 4.2.18.5 - New command SYS.BLE.Select,<index> - see chapter 4.2.18.6 - New command SYS.BLE.Show,<index>,<"text"> - see chapter 4.2.18.7 - New command SYS.BlueID.SetTicket,<"NFCCardUID">,<start_date>,<stop_date> - see chapter 4.2.19.1 - New command SYS.BlueID.ListTickets - see chapter 4.2.19.2 - New command SYS.BlueID.ClearTickets - see chapter 4.2.19.3 - New command SYS.BlueID.ClearLastTicket - see chapter 4.2.19.4 - New command SYS.BlueID.SetState,<state> - see chapter 4.2.19.5 - New command SYS.NFC.Enable=<Serial0|Serial1> - see chapter 4.2.20.1 - New command SYS.NFC.Disable - see chapter 4.2.20.2 - New command SYS.NFC.Reset - see chapter 4.2.20.3 -New command SYS.NFC.LED[1..3],<Low|High|hPulse|lPulse|Cyclic>[,<htime>,<ltime>,<cycles>] - see chapter 4.2.20.4 - New command MSG.Event - see chapter 4.2.24 - New command GSM.SMS.SendMulti - see chapter 4.7.5.3 - Implementation of concurrent reception with three satellite navigation systems - see chapter 4.5.1.18 - New command TCP.Client.FlushSendBuffer - see chapter 4.8.1.8 - New command MSG.Send.TCPBuffer,<protocol>,<"text"> - see chapter 4.10.1.10 - New command MSG.Version.AddTag,<"tag"> - see chapter 4.10.3.7 - New command MSG.Version.DelTag,<"tag"> - see chapter 4.10.3.8 - Added new protocol $GP3DP with a hex value of 8000 - Added/changed/removed configuration parameters: - Increased the size of the URL name for the parameter TCP.CLIENT.CONNECT to 255 characters - New configuration settings for DEVICE.COMM.BINEVENT=raw,<timeout> - see chapter 5.1.8 - Added new configuration parameter DEVICE.VIN=<"VIN"> - see chapter 5.1.24 - Extended settings (0x10 for USB, 0x20 for BLE, 0x3F (default) for all interfaces) for configuration parameter DEVICE.CMD.PFAL.EN - see chapter 5.1.5 - Added new configuration parameter DEVICE.GPS.NAVDISTMUL - see chapter 5.1.16 - Added new configuration parameter DEVICE.DTCO.D8=B0,<format> - see chapter 5.1.22 - Added new configuration parameter DEVICE.NFC=<Serial0|Serial1|off> - see chapter 5.1.23 - Extended settings for configuration parameter GSM.MODEPREF - see chapter 5.9.10	14/05/2018

Version	Author	Changes	Modified Date
		- Added new configuration parameter GSM.SIMSLOT,<1|2> - see chapter 5.9.11 - Added new configuration parameter GPRS.APN2=<apn> - see chapter 5.10.2 - Added new configuration parameter PPP.USERNAME2=<user> - see chapter 5.11.2 - Added new configuration parameter PPP.PASSWORD2=<pwd> - see chapter 5.11.4 - New configuration settings TCP.CLIENT.SENDMODE=2[,<buffer_level>]- see chapter 5.12.8 - Added new configuration parameter BLE.WHITELIST=<None|Public|Name|MAC> - see chapter5.14.2 - Added new configuration parameter BLUEID.DEVID=<"friendlyName">,<"soid"> - see chapter 5.15.1 - Added new configuration parameter BLUEID.KEY1=<"PublicKey"> - see chapter 5.15.2 - Added new configuration parameter BLUEID.KEY2=<"PrivateKey"> - see chapter 5.15.3 - Added new configuration parameter UDP.CLIENT.CONNECT - see chapter 5.16.1 - Added new configuration parameter UDP.CLIENT.TIMEOUT - see chapter 5.16.2 - Added/changed/removed events and states: - New comparators ("=","!=") for SYS.eSerialData0<comp>whitelist- see chapter 6.1.1 - New comparators ("=","!=") for SYS.eSerialData1<comp>whitelist - see chapter 6.1.1 - New comparators ("=","!=") for SYS.eUSBData<comp>whitelist - see chapter 6.1.2 - New wakeup conditions for Doze mode SYS.Device.eStart=Doze[,<reason>] - see chapter 6.1.8 - New comparators ("=","!=") for SYS.1Wire.eRegister<comp>whitelist - see chapter 6.1.18 - New comparators ("=","!=") for SYS.1Wire.eRelease<comp>whitelist - see chapter 6.1.18 -New event SYS.NFC.eCARD=<"ID"> - see chapter 6.1.20 -New state SYS.NFC.sCARD=<"ID|None"> - see chapter 6.1.20 -New event SYS.NFC.eCARD<comp>whitelist - see chapter 6.1.20 -New event SYS.NFC.eCARD<comp>whitelist - see chapter 6.1.20 -New event SYS.NFC.eLOST[<comp>whitelist] - see chapter 6.1.20 -New state SYS.NFC.sCARD<comp>whitelist - see chapter 6.1.20 -New event SYS.eDTCO.DRIVER.STATE - see chapter 6.1.21 -New event Sys.eUserText[=<"text">] - see chapter 6.1.22 -New event BLUEID.eDATA - see chapter 6.2.1 -New event BLUEID.eCMD=<"Command"> - see chapter 6.2.1 -New event BLUEID.eTICKETS - see chapter 6.2.1 -New event GSM.eMCC[<comp><MCC>] - see chapter 6.6.2 -New event TCP.UDP.eReceived[=<"text">] - see chapter 6.7.3 - Added/changed/removed dynamic variables: -New dynamic variable &(Replace<index>) - see chapter 7 -New dynamic variable &(UserEventText) - see chapter 7 -New dynamic variable &(UDPText) - see chapter 7 -New dynamic variable &(MaxMem) - see chapter 7 -New dynamic variable &(UserEventText) - see chapter 7 -New dynamic variable &(BLE.List2) - see chapter 7 -New dynamic variable &(BLE.MAC) - see chapter 7 -New dynamic variable &(BLE.UUID) - see chapter 7 - New dynamic variable &(BLE.Major) - see chapter 7 - New dynamic variable &(BLE.Minor) - see chapter 7 - New dynamic variable &(TicketID) - see chapter 7 - New dynamic variable &(MobileDeviceID) - see chapter 7 - New dynamic variable &(NFCUID) - see chapter 7	

Version	Author	Changes	Modified Date
		- New dynamic variable &(DTCO.D8.BLOB) - see chapter 7 - New dynamic variable &(DTCO.D8.ECU_ID) - see chapter 7 - New dynamic variable &(DTCO.D8.ECU_SW_ID) - see chapter 7 - New dynamic variable &(DTCO.D8.DATE) - see chapter 7 - New dynamic variable &(DTCO.D8.TIME) - see chapter 7 - New dynamic variable &(DTCO.D8.TIME_ZONE) - see chapter 7 - New dynamic variable &(DTCO.D8.VEHICLE_STATE) - see chapter 7 - New dynamic variable &(DTCO.D8.VEHICLE_SPEED) - see chapter 7 - New dynamic variable &(DTCO.D8.VEHICLE_ID) - see chapter 7 - New dynamic variable &(DTCO.D8.VEHICLE_REG) - see chapter 7 - New dynamic variable &(DTCO.D8.DRIVER1.ID) - see chapter 7 - New dynamic variable &(DTCO.D8.DRIVER2.ID) - see chapter 7 - New dynamic variable &(DTCO.D8.DRIVER1.WORKSTATE) - see chapter 7 - New dynamic variable &(DTCO.D8.DRIVER1.TIMESTATE) - see chapter 7 - New dynamic variable &(DTCO.D8.DRIVER2.TIMESTATE) - see chapter 7 - New dynamic variable &(DTCO.D8.DRIVER1.CARDSTATE) - see chapter 7 - New dynamic variable &(DTCO.D8.DRIVER2.CARDSTATE) - see chapter 7 - New dynamic variable &(DTCO.D8.DRIVER1.OVERSPEED) - see chapter 7 - New dynamic variable &(DTCO.D8.DRIVER2.OVERSPEED) - see chapter 7 - New dynamic variable &(DTCO.D8.DISTANCE) - see chapter 7 - New dynamic variable &(DTCO.D8.TRIPDIST) - see chapter 7 - New dynamic variable &(DTCO.D8.D1) - see chapter 7 - New dynamic variable &(DTCO.D8.D2) - see chapter 7 - New dynamic variable &(DTCO.D8.IGN) - see chapter 7 - New dynamic variable &(DTCO.D8.DRAWER) - see chapter 7 - New dynamic variable &(DTCO.D8.OPMODE) - see chapter 7	

Version	Author	Changes	Modified Date
2.16.0.0	F.Beqiri	- Starting from the firmware version 2.16.0, the following features are added/ changed: - Added/changed/removed PFAL commands: - Increased number of Timers, Triggers, Counters and Macros to 40 - Increased number of global CAN messages to max. 50 (FOX3 has 31 / IOBOX-CAN has 14) - Increased number of global CAN variable slots to max. 50 - Increased the buffer size of the send commands to 4096 bytes - New command MSG.Event - see chapter 4.2.24 - New command TCP.Client.FlushSendBuffer - see chapter 4.8.1.8 - New command GSM.SMS.SendMulti - see chapter 4.7.5.3 - Implementation of concurrent reception with three satellite navigation systems – see chapter 4.5.1.18 - Added/changed/removed configuration parameters: - Increased the size of the URL name for the parameter TCP.CLIENT.CONNECT to 255 characters - New configuration settings for DEVICE.COMM.BINEVENT=raw,<timeout> - see chapter 5.1.8 - New configuration settings TCP.CLIENT.SENDMODE=2[,<buffer_level>]- see chapter 5.12.8 - Added/changed/removed events and states: - New event Sys.eUserText[=<"text">] - see chapter 6.1.22 - New comparators ("=","==","!=") for GSM.eMCC[<comp><MCC>] - see chapter 6.6.2 - New comparators ("=","==","!=") for GSM.eOpLost[<comp><ID|"name">] - see chapter 6.6.2 - New comparators ("=","==","!=") forGSM.eOpFound[<comp><ID|"name">] - see chapter 6.6.2 - New comparators ("=","==","!=") for GSM.sOpValid[<comp><ID|"name">]- see chapter 6.6.2 - New comparators ("=","!=") for SYS.1Wire.eRegister<comp>whitelist - see chapter 6.1.18 - New comparators ("=","!=") for SYS.1Wire.eRelease<comp>whitelist - see chapter 6.1.18 - New comparators ("=","!=") for SYS.eSerialData0<comp>whitelist- see chapter 6.1.1 - New comparators ("=","!=") for SYS.eSerialData1<comp>whitelist - see chapter 6.1.1 - New comparators ("=","!=") for SYS.eUSBData<comp>whitelist - see chapter 6.1.2 - New wakeup conditions for Doze mode SYS.Device.eStart=Doze[,<reason>] - see chapter 6.1.8 Added/changed/removed dynamic variables: - New dynamic variable &(Replace<index>) - see chapter 7 - New dynamic variable &(UserEventText) r - see chapter 7	20/3/2018
3.0.0.0	F.Beqiri	- Initial release of the firmware 3.0.0 (applies only to devices as mentioned in chapter 1). - Please refer to the corresponding release note about the changes and new features added in this firmware release.	04/07/2017
2.15.0.0	F.Beqiri	- Initial version of the firmware 2.15.0 (applies only to devices as mentioned in chapter 1). - Please refer to the corresponding release note about the changes and new features added in this firmware release.	3/31/2017

Contents

1: Introduction 26

1.1. Scope of the document 27
1.2. Quick reference table 28
1.3. Related documents 29

2: General Information about Firmware 31

2.1. Features of the operating firmware 31
2.2. The principle operation 32
2.3. TCP/IP Overview 34

3: PFAL Command Syntax Reference 35

3.1. PFAL structure and responses 35
3.1.1. Command types 36
3.1.2. Aliases 37
3.1.3. Identifiers (Optional) 37
3.1.4. Response structure 38

4: PFAL Commands 40

4.1. Alarm 51
4.1.1. Alarm.Info – Displays all conditions defined in a specific alarm 51
4.1.2. Alarm.Clear – Clears and erases a specific alarm 51
4.1.3. Alarm.Reload – Reloads all alarms 52
4.1.4. Alarm.Call – Executes actions on the specified alarm index 52
4.2. SYS 53
4.2.1. Sys.Security 53
4.2.2. Sys.RUpdate 55
4.2.3. Sys.WebUpdate 61
4.2.4. Sys.Device 64
4.2.5. Sys.Power 76
4.2.6. Sys.Set/GetTime 77
4.2.7. Sys.1-Wire 78
4.2.8. Sys.GSM 80
4.2.9. Sys.GPS 81
4.2.10. Sys.Timer 82
4.2.11. Sys.Trigger 89
4.2.12. Sys.Counter 92
4.2.13. Sys.nvCounter 96
4.2.14. Sys.Macro 99

4.2.15. Sys.CAN - 1st CAN bus 100
4.2.16. Sys.CANB - Second CAN bus 114
4.2.17. Sys.WLAN 126
4.2.18. Sys.LUA 127
4.2.19. Sys.BLE 132
4.2.20. Sys.BlueID 136
4.2.21. Sys.NFC 139
4.2.22. Sys.UserEvent 142
4.2.23. Sys.WhiteList 143
4.2.24. Sys.Bat 145
4.2.25. SYS.ModBus 148
4.2.26. Sys.MSG 152
4.3. CNF 153
4.3.1. Cnf.Set,<parameter_name=value> - Checks/Sets/Stores the parameter settings into the device 153
4.3.2. Cnf.Get,<parameter_name> - Gets parameter settings from device 154
4.3.3. Cnf.Clear,<parameter_name> - Clears settings of the parameter name 155
4.3.4. Cnf.ShowUser - Shows the configuration settings defined by user 155
4.3.5. Cnf.ShowDefault - Returns default configuration settings 156
4.3.6. Cnf.Show - Returns all used parameter settings 156
4.3.7. Cnf.Search,<parameter_name> – Searches for a parameter name 156
4.3.8. Cnf.Backup - Backup current user configuration 157
4.3.9. Cnf.EraseBackup – Erases the backed up user configuration settings 157
4.3.10. Cnf.Restore – Restores the backed up user configuration settings 158
4.3.11. Cnf.Write,<parameter_name=value> - Saves parameter settings to device 158
4.3.12. Cnf.Write,Device.CAN.Transceiver=<on> - Ensures proper operation of IO2 & IO3 159
4.3.13. CNF.Load – Overwrites the existing config with a file from flash 159
4.4. IO COMMANDS 160
4.4.1. Firmware-indexed IOs 160
4.4.2. IO<index>.Set=<conf_type> – Defines the output behaviour 162
4.4.3. IO<index>.Get - Returns the current function and level of IO 164
4.4.4. IO<index>.GetDI – Returns the level of the digital inputs 164
4.4.5. IO<index>.GetAI – Returns the level of the analog inputs 165
4.4.6. IO<index>.GetDO – Returns the level of the digital output 165
4.4.7. IO<index>.Config - Configures the functionality and behaviours of IOs 166
4.4.8. IO<index>.Calibrate - Calibrates the offset or gain of analog input 170
4.4.9. IO<index>.ClearCalibration– Erases the stored calibration values for IO 172
4.4.10. IO<index>.Info – Returns the current function of specified IO 173
4.4.11. IO.Counter 173
4.5. GPS COMMANDS 175
4.5.1. GPS.Nav 175
4.5.2. GPS.History 188

4.5.3. GPS.Geofence 197
4.5.4. GPS.MultiGeofence 199
4.5.5. GPS.WPGeofence 202
4.6. EcoDrive 207
4.6.1. EcoDrive.TripStart - Starts a new EcoDrive trip 207
4.6.2. EcoDrive.TripStop - Ends a started EcoDrive trip 207
4.6.3. EcoDrive.CurrentTrip - Reports current trip data 208
4.6.4. EcoDrive.LastTrip - Reports last trip data 209
4.7. GSM 209
4.7.1. GSM General 209
4.7.2. GSM.CMB 214
4.7.3. GSM.VoiceCall 215
4.7.4. GSM.Audio 217
4.7.5. GSM.SMS 226
4.7.6. GSM.DataCall 230
4.7.7. GSM.GPRS 232
4.7.8. GSM.SetInternal/ExternalAntenna 234
4.8. TCP COMMANDS 235
4.8.1. TCP. Client 235
4.8.2. TCP.Storage 240
4.8.3. TCP.SMTP 243
4.8.4. TCP.Client2 246
4.8.5. TCP.MQTT 251
4.9. WLAN 256
4.9.1. WLAN.Scan - Scan for new WLAN networks 256
4.9.2. WLAN.Connect – Connect to a WLAN network profile 256
4.9.3. WLAN.Send,<protocol>,<"text"> – Sends protocols & user text via WLAN to server 257
4.9.4. WLAN.Client.Disconnect – Disconnects from a TCP server 257
4.9.5. WLAN.Disconnect – Disconnects from WLAN network 258
4.9.6. WLAN.MAC – Shows the MAC Address of the WIFI module 258
4.10. MSG 258
4.10.1. MSG.Send 258
4.10.2. Msg.ClearBuffer - Clears all information in a buffer 268
4.10.3. MSG.Mode 269
4.10.4. MSG.Event 271
4.10.5. MSG.Version 273
4.10.6. MSG.Info 276
4.10.7. MSG.Feature 278

5: Configuration Settings 280

5.1. DEVICE 286
5.1.1. DEVICE.NAME 286

5.1.2. DEVICE.SERIAL<port>.BAUDRATE 286
5.1.3. DEVICE.SERIAL<port>.FORCEON=<on_off> 287
5.1.4. DEVICE.SERIAL<port>.HANDSHAKE 288
5.1.5. DEVICE.CMD.PFAL.EN 289
5.1.6. DEVICE.COMM.<interface> 291
5.1.7. DEVICE.SERIAL1.MODE485=<on_off> 292
5.1.8. DEVICE.COMM.BINEVENT<port> 292
5.1.9. DEVICE.BAT.MODE 294
5.1.10. DEVICE.BAT.CHARGEMODE 295
5.1.11. DEVICE.IGNTIMEOUT 296
5.1.12. DEVICE.GSM.STARTUP 297
5.1.13. DEVICE.GPS.AUTOCORRECT 297
5.1.14. DEVICE.GPS.CFG 300
5.1.15. DEVICE.GPS.TIMEOUT 302
5.1.16. DEVICE.GPS.NAVDISTMUL 303
5.1.17. DEVICE.PFAL.SEND.FORMAT 303
5.1.18. DEVICE.CAN.OBD.STARTUP 304
5.1.19. DEVICE.CAN.FMS.STARTUP 305
5.1.20. DEVICE.CAN.STARTUP 306
5.1.21. DEVICE.CAN.DTCOFMS.STARTUP 307
5.1.22. DEVICE.CAN.ERR.EVENTS 308
5.1.23. DEVICE.CAN.EVENT_<slot> 308
5.1.24. DEVICE.DTCO.D8 311
5.1.25. DEVICE.LOWPOWER 311
5.1.26. DEVICE.GPS.JAMMINGLEVEL=<min>,<max> 312
5.1.27. DEVICE.WLAN.STARTUP 313
5.1.28. DEVICE.NFC 313
5.1.29. DEVICE.VIN 314
5.1.30. DEVICE.SERIAL1.MODE485 314
5.2. REPLACE 315
5.2.1. REPLACE<index> 315
5.3. USER 317
5.3.1. USER<index>=<user_text> 317
5.4. MOTION 317
5.4.1. MOTION.FILTER 318
5.4.2. MOTION.BEARING 319
5.4.3. MOTION.FORCE 320
5.4.4. MOTION.3DFORCE 321
5.5. ALIAS 322
5.5.1. ALIAS.<type> 322
5.6. DBG 323
5.6.1. DBG.EN 323
5.7. PROT 324

5.7.1. PROT.<message_id> 324
5.7.2. PROT.START.BIN 325
5.7.3. PROT.ERR 326
5.7.4. PROT.NA 326
5.7.5. PROT.EMPTY 326
5.8. ECODRIVE 327
5.8.1. ECODRIVE.CAR 327
5.8.2. ECODRIVE.LIMITS 329
5.8.3. ECODRIVE.TOPOLOGY 330
5.8.4. ECODRIVE.AUTOSTART 330
5.9. GSM 331
5.9.1. GSM.PIN 331
5.9.2. GSM.BALANCE.DIAL 332
5.9.3. GSM.CALLID.EN 332
5.9.4. GSM.OPERATOR.BLACKLIST 333
5.9.5. GSM.OPERATOR.SELECTION 333
5.9.6. GSM.EXT.WHITELIST 336
5.9.7. GSM.OPERATOR.GPRSCHECK 337
5.9.8. GSM.OPLOST.RESTART 337
5.9.9. GSM.SMS.RESPONSE 338
5.9.10. GSM.MODEPREF 339
5.9.11. GSM.SIMSLOT 339
5.10. GPRS 340
5.10.1. GPRS.APN 340
5.10.2. GPRS.APN2 340
5.10.3. GPRS.APN.ALTERNATIVE 341
5.10.4. GPRS.AUTOSTART 342
5.10.5. GPRS.DIAL 342
5.10.6. GPRS.TIMEOUT 343
5.10.7. GPRS.QOSMIN 343
5.10.8. GPRS.QOS 346
5.11. PPP 348
5.11.1. PPP.USERNAME 348
5.11.2. PPP.USERNAME2 349
5.11.3. PPP.PASSWORD 349
5.11.4. PPP.PASSWORD2 349
5.11.5. PPP.AUTOPING 350
5.11.6. PPP.AUTH 350
5.12. TCP 351
5.12.1. TCP.CLIENT.CONNECT 351
5.12.2. TCP.CLIENT.ALTERNATIVE 352
5.12.3. TCP.CLIENT.PING 353
5.12.4. TCP.CLIENT.TIMEOUT 354

5.12.5. TCP.CLIENT.DNS.TIMEOUT 355
5.12.6. TCP.CLIENT.LOGIN 355
5.12.7. TCP.CLIENT.LOGIN.EXT 357
5.12.8. TCP.CLIENT.SENDMODE 358
5.12.9. TCP.SERVICE.CONNECT 360
5.12.10. TCP.SERVICE.TIMEOUT 360
5.12.11. TCP.STORAGE 361
5.12.12. TCP.SMTP.CONNECT 362
5.12.13. TCP.SMTP.SUBJECT 362
5.12.14. TCP.SMTP.LOGIN 363
5.12.15. TCP.SMTP.FROM 364
5.12.16. TCP.SLA 364
5.12.17. TCP.CLIENT2.CONNECT 365
5.12.18. TCP.CLIENT2.ALTERNATIVE 366
5.12.19. TCP.CLIENT2.SENDMODE 367
5.12.20. TCP.CLIENT2.PING 370
5.12.21. TCP.CLIENT2.TIMEOUT 370
5.12.22. TCP.CLIENT2.LOGIN.EXT 371
5.12.23. TCP.CLIENT2.LOGIN 372
5.12.24. TCP.CLIENT2.DNS.TIMEOUT 373
5.12.25. MQTT.CLIENT.CONNECT 374
5.12.26. MQTT.CLIENT.PING 375
5.12.27. MQTT.CLIENT.TIMEOUT 375
5.12.28. MQTT.CLIENT.SENDMODE 376
5.12.29. MQTT.CLIENT.WELCOME 377
5.12.30. MQTT.CLIENT.LASTWILL 377
5.12.31. MQTT.CLIENT.ID 377
5.12.32. MQTT.CLIENT.USER 377
5.13. WLAN 377
5.13.1. WLAN.MODE 378
5.13.2. WLAN.CLIENT.LOGIN 378
5.13.3. WLAN.NET<id>.SSID 378
5.13.4. WLAN.NET<id>.TYPE 379
5.13.5. WLAN.NET<id>.PSK 380
5.13.6. WLAN.NET<id>.SECURITY 380
5.13.7. WLAN.NET<id>.IP 381
5.13.8. WLAN.NET<id>.PORT 381
5.13.9. WLAN.RSSIMIN 381
5.13.10. WLAN.MODE 382
5.14. BLE 382
5.14.1. BLE.ADVNAME 382
5.14.2. BLE.WHITELIST 383
5.14.3. BLE.SCANDURATION 383

5.15. BLUEID 384
5.15.1. BLUEID.DEVID 384
5.15.2. BLUE.KEY1 384
5.15.3. BLUE.KEY2 385
5.16. UDP 385
5.16.1. UDP.CLIENT.CONNECT 385
5.16.2. UDP.CLIENT.TIMEOUT 386
5.17. GF 386
5.17.1. GF.CONFIG 389
5.17.2. GF.AREA<id> 390
5.17.3. GF<id> 391
5.18. AL - Set alarm configuration 396
5.18.1. AL<index>= <conditions>:<actions> 396
5.19. ConsoleFlow Settings 402
5.19.1. CF.CLIENT.CONNECT 402
5.19.2. CF.CLIENT.DEVICE_NAME 402
5.19.3. CF.CLIENT.DEVICE_DESCRIPTION 402
5.19.4. CF.CLIENT.STATUS_UPDATE_INTERVAL 403
5.19.5. CF.CLIENT.CONTENT_CHECK_INTERVAL 403
5.19.6. CF.CLIENT.GROUPS_CAP_EXCHANGE_DATA 403
5.19.7. CF.CLIENT.GROUPS_CAP_SELECTION_DATA 404
5.19.8. CF.CLIENT.GROUPS_TELEMETRY_DATA 404
5.20. Optional Settings 404
5.20.1. STORAGE<id> 404
5.20.2. DEVICE.CAN.STARTUP 405
5.20.3. GPS.HISTORY.MODE 405
5.20.4. GPS.HISTORY.PUSHMODE 405
5.20.5. DEVICE.CAN.MSG 405
5.20.6. DEVICE.CAN.VAR 405
5.20.7. DEVICE.OBD.STARTUP 405
5.20.8. DEVICE.FMS.STARTUP 406
5.20.9. GSM.BANDPREF 406
5.20.10. GSM.PROFILE.AUDIO<prof_index> 406
5.20.11. GSM.PROFILE.CURRENTAUDIO 407
5.20.12. MACRO<index> 407
5.20.13. DEVICE.GPS.HEADING 408
5.20.14. DEVICE.GPS.HEADING2 408
5.20.15. DEVICE.RUPDATE.SELECTION 409
5.20.16. DEVICE.WLAN.STARTUP 409
5.20.17. DEVICE.RUPDATE.SELECTION 409
5.20.18. DEVICE.CANopen.STARTUP 409
5.20.19. WHITELIST.ACTIVE 410
5.20.20. DEVICE.MODBUS 410

5.20.21. MODBUS.POLL 410

6: Events & States 411

6.1. SYS 412
6.1.1. Sys.eSerialData 412
6.1.2. Sys.eUSBData 414
6.1.3. Sys.UserEvent 415
6.1.4. Sys.CAN 415
6.1.5. Sys.CANB 418
6.1.6. Sys.eOBDII 418
6.1.7. Sys.eFMS 419
6.1.8. Sys.eJ1939 419
6.1.9. Sys.eCanDTCO 420
6.1.10. Sys.Device 421
6.1.11. Sys.Timer 424
6.1.12. Sys.Trigger 425
6.1.13. Sys.Counter 426
6.1.14. Sys.nvCounter 427
6.1.15. Sys.Power 428
6.1.16. Sys.Bat 428
6.1.17. Sys.Info 430
6.1.18. Sys.eTimeSync 430
6.1.19. Sys.1Wire 431
6.1.20. Sys.BLE 432
6.1.21. Sys.NFC 433
6.1.22. Sys.WLAN 434
6.1.23. Sys.eDTCO.DRIVER.STATE 434
6.1.24. Sys.eUserText 434
6.1.25. Sys.Iobox 435
6.1.26. Sys.eco.PDO 435
6.2. BLUEID 436
6.2.1. BLUEID.e/s 436
6.3. IO 436
6.3.1. IO.e/s 436
6.3.2. IO.Motion 437
6.3.3. IO.PulseCnt 438
6.4. GPS 439
6.4.1. GPS.eJamming 439
6.4.2. GPS.Nav 439
6.4.3. GPS.Time 441
6.4.4. GPS.History 444
6.4.5. GPS.Geofence 445
6.4.6. GPS.Area 446

6.4.7. GPS.WPGF 447
6.4.8. GPS.eExtAnt(Un)Plugged 448
6.4.9. GPS.MultiGeofence 448
6.5. EcoDrive 449
6.6. GSM 450
6.6.1. GSM.eJamming 450
6.6.2. GSM (Operator) 450
6.6.3. GSM.eCellChange 452
6.6.4. GSM.VoiceCall 452
6.6.5. GSM.SMS 453
6.6.6. GSM.DataCall 454
6.6.7. GSM.GPRS 455
6.7. TCP 456
6.7.1. TCP.Client 456
6.7.2. TCP.SMTP 457
6.7.3. TCP.UDP 458
6.7.4. TCP.CLIENT2 458
6.7.5. MQTT Client 459
6.8. WLAN 460

7: Dynamic Entries/Variables 462

8: Application Guide 488

8.1. Rules to be considered 488
8.2. Start a GPRS/TCP connection 488

9: Sending SMS to Lantronix Devices 490

10: NMEA and Lantronix Messages 492

10.1. Description of NMEA output messages 493
10.2. $GPGGA message 494
10.3. $GPRMC message 495
10.4. $GPGSV message 496
10.5. $GPGSA message 497
10.6. $GPVTG message 498
10.7. $GLGSA message 499
10.8. $GLGSV message 500
10.9. $GPGLL message 501
10.10. $GPIOP message 502
10.11. $GPGSM message 503
10.12. $GPAREA message 504
10.13. $GP3DP message 505

10.14. BIN protocol 506

11: Appendix 508

11.1. How to update new firmware 508
11.2. Supported protocols 508
11.3. Supported character sets 509
11.3.1. GSM alphabet tables and UCS2 character values 510
11.4. How to convert the coordinates 518
11.5. Explanation of the History Binary Data 518
11.6. AVL Device Configuration Examples 519
11.6.1. Basic Configuration Examples 520
11.6.2. Advanced Examples 523
11.7. ISP, GPRS configuration parameters of German service providers 530
11.8. Used abbreviations 531

List of Figures

Figure 2-1 Interfaces to access the Remote Server via GPRS Network _____ 33
Figure 4-1 Pin outs and accessories _____ 160
Figure 5-1 Filtering in motion _____ 319
Figure 5-2 Possibilities of defining Geofences within an area _____ 387
Figure 5-3 Determining a Zone's Grid Coordinates _____ 388
Figure 5-4 Determining the sign of coordinate values _____ 388
Figure 5-5 Covering an area using several Geofences _____ 394
Figure 5-6 Covering a route using geofences in different areas _____ 395

List of Tables

Table 1-1 Lantronix products with AVL firmware 3.x.x ____ 26
Table 1-2 Lantronix products with firmware 2.15.x and 2.16.x ____ 26
Table 3-1 PFAL command syntaxes ____ 35
Table 3-2 Supported command types <c_type> ____ 37
Table 3-3 Identifier syntaxes ____ 37
Table 3-4 Response messages structure ____ 38
Table 4-1 List of PFAL Commands ____ 40
Table 4-2 Binary update command syntax ____ 58
Table 4-3 List of binary commands ____ 59
Table 4-4 Inputs and Outputs of AVL devices ____ 160
Table 5-1 List of Configuration Parameters ____ 280
Table 5-2 Events and states grouped by major category ____ 399
Table 6-1 Comparators List ____ 411
Table 6-2 UDP States and Events ____ 458
Table 8-1 Adapt configuration settings to application conditions ____ 488
Table 10-1 Description of NMEA output messages ____ 493
Table 10-2 Description of NMEA output messages ____ 494
Table 10-3 GPRMC message data format ____ 495
Table 10-4 GPGSV message data format ____ 496
Table 10-5 GPGSV message data format ____ 497
Table 10-6 GPVTG message data format ____ 498
Table 10-7 GLGSA message data format ____ 499
Table 10-8 GLGSV message data format ____ 500
Table 10-9 GPGLL message data format ____ 501
Table 10-10 GPIOP message data format ____ 502
Table 10-11 GPGSM message data format ____ 503
Table 10-12 GPAREA message data format ____ 504
Table 10-13 GP3DP message data format ____ 505
Table 11-1 Supported protocols ____ 508
Table 11-2 Examples for character definitions depending on alphabet. ____ 510
Table 11-3 Main character table of GSM 03.38 alphabet ____ 511
Table 11-4 Maximum values & the time the history space will be used up ____ 519
Table 11-5 Service provider information, valid 16.10.2001 ____ 531
Table 11-6 Used Abbreviations ____ 531

1: Introduction

This document describes the functions implemented in the firmware versions 2.15.x and higher and 3.x.x and it refers only to the following Lantronix products.

Table 1-1 Lantronix products with AVL firmware 3.x.x

Lantronix products with AVL firmware 3.x.x		
Series	**Devices**	**Hardware Revision[1]**
FOX3-2G Series	FOX3-2G	13, 15, 17, 19, 20, 21
FOX3-3G Series	FOX3-3G	06, 11, 13, 15, 17, 19, 20, 21
	FOX3-3G-BLE	
	FOX3-3G-AU	
	FOX3-3G-DR	
FOX3-4G Series	FOX3-4G-EU	All
	FOX3-4G-M1	
	FOX3-4G-C4-NA	
	FOX3-4G-C1-NA	
BOLERO40 Series	BOLERO41	All
	BOLERO43	
	BOLERO45	

Table 1-2 Lantronix products with firmware 2.15.x and 2.16.x

Lantronix products with firmware 2.15.x & 2.16.x		
Series	**Devices**	**Hardware Revision[1]**
FOX3-2G Series	FOX3	01,02,03,04,05,07,08,09,10,12,14,16
	FOX LITE	
FOX3-3G Series	FOX3-3G	01,02,03,04,05,07,08,09,10,12,14,16
	FOX3-3G-AU	
	FOX3-3G-LITE	
BOLERO-LT2-MS-B1	-	05

[1]The hardware revision of the device can be read out with the command **$PFAL,MSG.Version.HardwareRev**. The hardware revision is also printed on the product label, located on the back panel of the device. In the Serial Number field there are 3 digits in parenthesis, for example, S/N: 60148(9**XX**)50600014, and the number “XX” is the hardware revision of the device. If the number is “11”, it means that the hardware revision is 11.

Caution: ***This document is strictly referred to the above listed products with a reference to both firmware. Never try to upgrade devices operating with firmware 2.x.x to the firmware 3.x.x and vice versa. You will not be able to operate with your device anymore. It is highly recommended to double check your products/devices regarding the firmware supported before upgrading/downgrading them to a new/old firmware. To get the***

best performance from your Lantronix AVL device you should make sure you're running the latest firmware release. For information about how to update your device, please refer to the AVL Firmware Release Notes document available on the Lantronix website. Before using an AVL device or upgrading it to a new firmware version, please read the latest product information provided in the hardware manual, see 1.3. Related documents.

Note: *Please refer to the chapter 1.2. Quick reference table to understand the symbols used throughout this document regarding the use of the PFAL commands and firmware.*

1.1. Scope of the document

The document is separated into the following chapters:

1. Chapter *2: General Information about Firmware* presents the firmware's basic benefits, lists key features and describes the principle of its operation. Figuratively are represented the general system requirements for each access method. It also provides a brief overview of the TCP/IP protocol.
2. Chapter *3: PFAL Command Syntax Reference* represents the structure of PFAL commands.
3. Chapter *4: PFAL Commands* gives a detailed description of the PFAL commands.
4. Chapter *5: Configuration Settings* gives a detailed description of configuration settings. Default values and example about commands are listed after each command description. It also includes the steps for the creation of applications, more especially how to specify alarm sources and the configuration possibilities by using a range of events, states and actions
5. Chapter *6: Events & States* gives a detailed description of events and states which are used to create alarm notifications. Each category of events and states is described separately. It also shows the differences between Events and States.
6. Chapter *7: Dynamic Entries/Variables* gives a detailed description of supported dynamic variables which can be used in alarms to report specific messages or values.
7. Chapter *8: Application Guide* describes how to transfer input messages from your PC to your AVL device, how to test and evaluate it. How to set alarms when an event occurs and how to configure input lines to release alarms/actions. It also describes how to communicate remotely with AVL device via a TCP-server.
8. Chapter *9: Sending SMS to Lantronix Devices* presents how SMS messages can be sent from your mobile phone to the AVL device and more precisely, how to configure AVL device via GSM (SMS).
9. Chapter *10: NMEA and Lantronix Messages* describes the output protocols (NMEA messages) supported by the AVL device.
10. Chapter *11: Appendix* represents how to reprogram the internal FLASH memory of the AVL device with new firmware, the supported character set, the default setting of the released firmware, configuration examples etc.

1.2. Quick reference table

Each PFAL command, configuration setting, and event/state description includes a quick reference table similar to the following example.

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	×

The table is intended as a quick reference to indicate the following functions:

PFAL: Is the PFAL command, configuration command or event/state supported by the firmware and/or hardware of the AVL devices?

✓	Yes
×	No
?	Optional (hardware/accessory option)
🗝	PREMIUM feature (Paid feature)

1, 2, 3 …¦ : Is the referenced parameter value supported by the hardware of the AVL device e.g. FOX3 Series? The numbers ([1, 2, 3 ..]) in the table indicate the referenced parameter values. Such a number is added only if the value related to that PFAL command is not supported by all devices. "1" references number "1" in the table, "2" references number "2" and so on.

✓	Yes
×	No
?	Optional (hardware/accessory option)
🗝	PREMIUM feature (Paid feature)

Example

Command Syntax	MSG.Send.Serial<port>,<protocols>,<"text">"
Example	$PFAL,MSG.Send.Serial0,8,"GPS positions"

Command description

Sends the specified protocols and/or user text to serial output.

Parameter description

Parameter	Value	Meaning
<port>	Specifies the serial port on the device.	
	0	Serial port 0
	1[1]	Serial port 1
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Defines the text to be sent to the specified serial port. Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (entry), which are described in chapter *7: Dynamic Entries/ Variables*.	

1.3. Related documents

In addition to this document, the following files comprise the set of Lantronix AVL product manuals. Product documentation for the related devices is available at https://www.lantronix.com/resources/product-index/ and application notes are available at https://www.lantronix.com/resources/app-notes/

NR	Document Title	Description
[1]	***FOX3 Series Hardware Manual***	Contains information about the hardware of the FOX3 device.
[2]	***AVL Firmware Release Notes***	AVL firmware release notes.
[3]	***App Note: Transform History Binary Data in NMEA Format for AVL Devices***	Contains information of how to transform history data that are being transmitted from BOLERO-LT via TCP connection.
[4]	***App Note: Firmware Web Update for AVL Devices***	Contains information how to perform a remote firmware upgrade over the air, using a TCP connection.
[5]	***App Note: In-Vehicle Installation Guidelines for FOX3 and BOLERO40 Series***	This document provides all the necessary information how to install your Lantronix product properly and safely in a vehicle.
[6]	***App Note: Remote Firmware Update with Workbench Software***	Contains information how to upgrade an AVL device to a new firmware version remotely using Workbench software.
[7]	***App Note: CAN Applications with AVL Devices***	Contains information how to connect a FOX3 series with CAN Bus option to an external CAN bus on a car or truck and read the CAN data stream.
[8]	***App Note: Communication of Passive RFID Reader and FOX3 Series Via Serial Link***	Contains information about the RFID reads and how to get stated with them.
[9]	***App Note: How to Collect CAN FMS/ J1939/OBD-II Data with FOX3 Series***	Contains information on how to connect the FOX3-2G/3G/4G series device to the vehicle and how to configure and use the CANBus FMS/J1939.OBDII features.
[10]	***App Note: Eco-Drive-GPS Premium Features in AVL Firmware 2.11.0 and Above***	This document contains information of how to use the features of the GPS -ECO-DRIVE supported as PREMIUM-FEATURE in the firmware versionavl_2.11.0 and above.
[11]	***App Note: AES_TCP Premium Feature in AVL Firmware 2.10.0 and Above***	Contains information about how the Advanced Encryption Standard (AES_TCP) feature in the AVL firmware 2.10.0 and higher works.
[12]	***App Note: Connecting a Bar Code Scanner***	This document contains information of how to connect a bar code scanner to a Lantronix AVL device and how to transmit the scanned data.
[13]	***App Note: Improving Asset Tracking Through Untethered Dead Reckoning With FOX3-3G-DR***	Contains information on the workings of the FOX3-3G-DR with Untethered Dead Reckoning (DR) technology.
[14]	***App Note: Getting Started with 1-Wire Devices***	Contains information how to use the 1-Wire interface on the FOX3/-3G/-4G/-4G Series.

NR	Document Title	Description
[15]	***App Note: Reading Real-time Data from Digital Tachograph with IOBOX-CAN and FOX3 Devices***	Provides information on how to configure your FOX3 series devices in combination with the accessory box IOBOX-CAN/WLAN, connect the IOBOXCAN/WLAN to D8 connector of the digital tachograph, read real time data from this interface and transfer them to your platform server.
[16]	***App Note: How to user IOBOX-WLAN with a FOX3 Series Device***	Provides information on how to configure your FOX3-3G/4G series devices in combination with the accessory box IOBOX-WLAN, connect the IOBOX-WLAN to your AP (access point) and send data over WLAN to your platform server.
[17]	***App Note: How to Use BLE with the FOX3-3G-BLE Device***	This guide shows how to configure your FOX3-3G-BLE device, connect it to the Lantronix evaluation android app and monitor device activities via the BLE interface
[18]	***App Note: Garmin® PND Integration with FOX3 Series***	This guide shows how to connect a Garmin® PND device that support Fleet Management Interface (FMI) to a Lantronix FOX3-2G/3G/4G device and how to configure your Lantronix device, manage the connection to a Garmin® PND and send data to the device from your server.
[19]	***App Note: Using a Garmin® PND with a FOX3 Series device***	This guide shows how to connect a Garmin® PND device that support Fleet Management Interface (FMI) to a Lantronix FOX3-2G/3G/4G device and how to configure your Lantronix device, manage the connection to a Garmin® PND and the connection to the GpsGate server.
[20]	***App Note: ContiPressureCheck™ System Integration into FOX3 Series***	This guide provides information on how to configure a FOX3-2G/3G/4G series device, connect to an in-vehicle installed ContiPressureCheck™ System (CPC-System), read the and transfer data to your platform server for further evaluation and analysis.
[21]	***App Note: Using Lua Scripts for FOX3 and BOLERO40 Series***	This guide provides information on how to get started with Lua script and a FOX3-2G/3G/4G series device, how to load Lua files into a device and start your own script.
[22]	***App Note: CANOpen Gateway Functions for AVL Devices***	This application note provides information on the CAN open gateway functions for AVL devices.
[23]	***App Note: Use of I/Os on AVL Devices***	This application note provides information and recommendations about the use of the I/O pins in the FOX3 when the internal CAN-Bus interface is disabled.

These PDF files are viewable and printable from Adobe Reader or any other PDF viewer programs. If you do not have the Adobe Reader installed, you can download it from https://www.adobe.com

2: General Information about Firmware

This firmware provides performance and flexibility for its users and system integrators to develop high-performance applications. It enables you to implement applications for tracking, controlling and monitoring Lantronix products and third-party devices connected to them as well as to set and poll the configuration remotely via SMS and Internet. This firmware offers you a speedy development of solutions in the fields of:

- Real time online tracking
- Fleet management / monitoring
- Security / emergency services
- Real time satellite navigation
- Territory management
- Personalized drivers' logbook
- Route verification
- Trip management / Distance calculations
- Theft protection
- Toll collection / pay as you drive
- Eco-Drive
- Encrypted TCP communication based on AES 128-bit algorithm and more.

The AVL firmware makes best use of the excellent hardware performance of all AVL devices. It is ideally suited for vehicle security and fleet management purposes. Of course, it is also plausible to monitor stationary devices (such as gas tanks, industrial machines, etc.).

2.1. Features of the operating firmware

The firmware supported by Lantronix AVL device provides the following main features:

- Device settings/behaviour can be fully adapted to user application requirements.
- Intelligent autonomous behaviour using sensors and actors.
- Possibility to gather and exchange information by device and or server.
- Advanced control through different communication channels (Local, GSM and GPRS/TCP communications).
- Allowing access to an internet server over GPRS.
- Advantages in terms of cost and speed, with low costs option for Web- based client-server applications (always on-line - pay for the data you send or receive, rather than the time spent on-line).
- Offering remote configuration and communication over GSM and TCP-connection, more specifically; SMS message, TCP packet generation and voice calls as well as handling of incoming SMS, voice calls and TCP packets.
- Offering remote firmware update.
- Supporting power saving features (sleep modes).
- Automatically switching between GSM and GPRS working modes.
- Automatically connecting/disconnecting to/from the GPRS/TCP services.
- Buffering of GPS positions in case of unexpectedly dropping of the GPRS/TCP connection.
  - The firmware contains a TCP buffer. Thus, the GPS position data can be internally stored in case the connection to the services will be dropped (e.g. bad GSM coverage). Once the connection will be re-established, the stored data will be sent directly to the used remote

server. Following a short overview, how many packets (data) can internally be buffered:

```
binary RMC:approx. 1800 (packets)
RMC+GPIO:approx. 400 (packets)
GGA,GSA,GSV,RMC,GLL,VTG,GPIO,GSM:approx. 80 (packets)
```

- Offering GPRS cell selection and re-selection processes (GSM.OPERATOR.SELECTION configuration-dependent).
- Commands and messages can be routed from one communication interface to another.
- Tracking down the initialization/execution of firmware, monitoring runtime errors from different communication interfaces (Locally, GSM and TCP).
- Locating object (vehicle etc.) position by GPS and sending its position by GPRS to remote servers and World Wide Web (IP-based application) or via SMS, Email or data call.
- Offering location and the tracking of objects (tracks, boats, vehicle etc.) on-line from Internet using WebMap.
- Allows configuration of simple and complex behaviour depending to its current situation.
- Supports user generated events being sent from/to the device in order to report changes or perform actions.
- Allows flexible configuration which allows to adapt its behaviour to almost any environment or situation.
- Detecting the status changes of digital inputs within a short period of time (min. 200msec).
- Supports a flexible power down mode. Various wakeup events be used (even in combination) to wake up the system.
- Up to 20 TIMERs available - TIMERs properties and their configuration methods affect the functionality to activate events handler and execute actions at regular interval.
- Up to 20 TRIGGERs implemented to execute and start various actions to a particular time.
- Up to 20 COUNTERs implemented to limit the number of actions executed automatically.
- History function (stores the waypoints of a vehicles path on-board FLASH memory. The waypoints are downloaded remotely by Internet or after the vehicle is back home by a PC).
- Download all or a part of the history stored data.
- Clear all history stored records.
- Geo-fencing functionality (park-area functionality, sending reports when leaving park-area).
- Up to 100 Geofencing zones included within up to 32 areas (with inside/outside features. Sending reports when device enters in a pre-defined zone, deviates off a pre-defined route or it detects that a vehicle leaves a pre-defined country and many others)

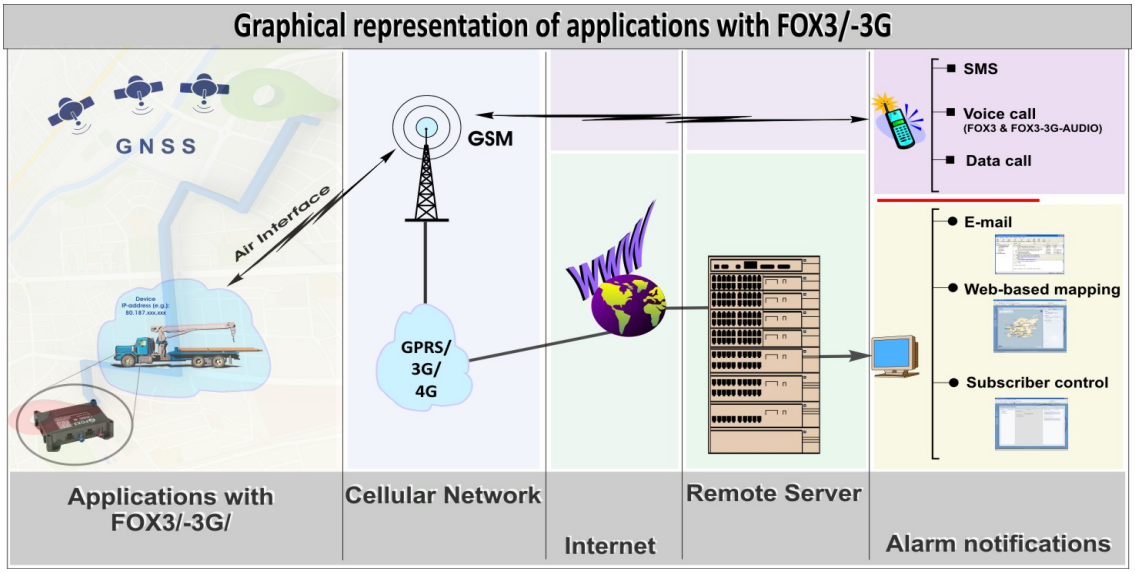
2.2. The principle operation

To integrate Lantronix products into the applications mentioned above you require the following:

- A Lantronix products,
- A GSM/GPS antenna (by insufficient network coverage with internal antennas),
- A SIM card for Voice and/or DATA,
- GPRS settings for cellular-data connectivity
- A remote server,
- Server settings for internet-data connectivity
- User -PC/Laptop

The illustration below represents interfaces that the AVL device uses to access the Remote Server via a GPRS Network. In addition, it shows that TCP communication enables AVL device to be monitored/tracked online from your PC via Internet services.

Figure 2-1 Interfaces to access the Remote Server via GPRS Network



The principle of system operation is simple. Each vehicle (object) is equipped with an AVL device, which consists of:

- an integrated GPS-receiver with external active antenna for reception of signals from GPS satellite system,
- a GSM/GPRS-modem with external antenna for transmission of these GPS data by radio.

The GPS receiver inside the device uses the satellite data to calculate the exact position of the vehicle (object) fitted with an AVL unit. The GPS data received from the AVL unit can be transferred through the GPRS network (IP-based) and the Internet to your remote server for online purposes. A user-developed program installed on the remote server can help you to connect to the AVL units installed in vehicles.

For such purposes, at first you must (re)configure your AVL units locally via serial port (with the help of any terminal program such as Lantronix Workbench software – a developed program to help you configure and evaluate Lantronix devices). All AVL devices are pre-configured to work with the default settings. So all default settings must be changed and adapted to your application conditions (including: GPRS settings of your provider, the remote server settings and the PIN of the used SIM card - see chapter *4.3.1.*). When the configuration of the AVL device is done, it tries to register itself into the GSM network. Once it is successfully registered into the GSM network, it can start automatically to establish a GPRS connection *(depending on the GPRS configuration – see chapter 5.10.)* and by means of TCP settings a TCP connection to the remote server.

Once the AVL device is attached to the GPRS network over the Basis Station, temporary a dynamic IP-address will be allocated by the AVL device from the GPRS network. With the help of this IP-address that constantly changes, the GPRS network enables AVL device to perform a TCP connection to the used remote server (to the user-assigned IP address and Port number). By means of these IP addresses as object identifiers, all AVL device can be direct configured from the remote server.

Such on-line applications enable you tracking and monitoring in a short time several 100 vehicles (objects) equipped with an AVL device.

Furthermore, the data transmitted from AVL device is received in real time. The AVL device can be programmed so that the vehicle location and additional information will be received not only via a TCP server, but also via SMS.

The operating firmware 2.x.x and 3.x.x offer a rich set of events, states and commands that you can use to customize high-performance web solutions. The solution architecture varies with the type of application you decide to create.

2.3. TCP/IP Overview

TCP (Transmission Control Protocol) is the most widely used transport protocol for non-real-time Internet applications like www, e-mail. It provides a connection-oriented end-to-end service ensuring the reliable transfer of data.

As with all other communication protocols, TCP/IP is composed of following layers:

IP	is responsible for moving packet of data from node to node. IP forwards each packet based on a four-byte destination address (the IP number). IP operates on gateway machines that move data from department to organization to region and then around the world.
TCP	is responsible for verifying the correct delivery of data from client to server. Data can be lost in the intermediate network. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received.
PORT	is a name given to the package of subroutines that provide access to TCP/IP on most system.

3: PFAL Command Syntax Reference

3.1. PFAL structure and responses

The input messages provided in the next section can be transmitted to the target device locally via a serial cable with the help of any terminal program, remotely via SMS or via a TCP connection with the help of a remote server.

Each PFAL message containing the command *<cmd>* is distinguished as an alone caption on which you will find a table divided in two rows with following meaning:

- The first row indicates the Command syntax, which could **not** be sent to the device in that form. Within the Command syntax there are invalid characters such as "**<**", "**>**" and assigned name, which are used only to show the Command syntax.
- The second row shows the example(s) how the message(s) **can** be sent to the FOX3 device. The set parameter settings in those examples depend on the user application. All examples can be modified and adapted to the user requirements.

The PFAL messages have the following formats, and in one of these formats the AVL device will accept the sent messages:

Table 3-1 PFAL command syntaxes

Header	Command	Parameter	Checksum	End Sequence
$PFAL	*<cmd>*	*<parameter>*	*[<*CKSUM>]*	*[<CR><LF>]*
$PFAL	*<cmd>*	*<parameter>*	None	*[<CR><LF>]*
$PFAL	*<cmd>*	*<parameter>*	*[<*CKSUM>]*	*[<CR><LF>]*
$PFAL	*<cmd>*	*<parameter>*	None	*[<CR><LF>]*

[$]PFAL	The [$]PFAL is message header.
<cmd>	The *<cmd>* determines the command(s) that will be executed. [$]PFAL and *<cmd>* are comma-separated. Commands are sorted by command types which are separated by dots ["." character] without double quotes. The command could also include values which are separated by an equal ["=" character] without double quotes. According to this explanation, the improved syntax to specify the *<cmd>* command is the following: `<type>.<index>.<subindex>` `<type>.<index>=<value` `<type>.<index>.<subindex>=<value>` Combining *<cmd>* commands on the same command line is allowed. If more than one *<cmd>*command is set on the same command line, they should be separated by semi-colon [";" character] without double quotes. Note that, the maximal length of a *<cmd>* command is limited to *1500 characters*. In this case the common syntax is: `<cmd _1>;<cmd _2>;< cmd_3>….<cmd _n>`
<parameter>	The *<parameter>* may contain different settings. Some parameters do not require any value, so left them empty. According to this explanation, the improved syntax of the *<parameter>* is: `<parameter>=<value>`

[<*CKSUM>]	Checksum is optional. If checksum **<*CKSUM>** is used, it consists of an asterisk ("*" character) without double quotes, followed by two hexadecimal values. Only PFAL commands with valid checksum can be accepted and executed. In order to calculate the Checksum of the command to be sent to the target device, use your own application. Below a small source code written in Visual Basic: `//**************************************` `Public Sub CheckSum(field As String)` `If field = "" then CS = "*"` `CS = 0` `For i = 1 to Len(field)` `CS = CS Xor Asc(Mid$(field, i, 1))` `Next` `CS = Hex(CS)` `If Len(CS) =1 then CS = "0" & CS` `CS = "*" & CS` `END SUB` `//**************************************` Therefore, the string over which the checksum will be calculated is: `field = PFAL,<cmd>,<parameter>` excluding "$" character. The "CS" variable in the CheckSum procedure above must be declared as a global variable.
[<CR><LF>]	*Optional.* Carriage Return and Line Feed (ASCII CODE #13#10 (without any spaces) - hexadecimal: 0x0D 0x0A) According to the above explanation, the improved format to specify a PFAL command is: `$PFAL,<c_type>.<c_index>.<c_subindex>,<parameter>=<value><*CKSUM><CR><LF>` or `$PFAL,<c_type>.<c_index>.<c_subindex>,<parameter>=<value><CR><LF>` or `PFAL,<c_type>.<c_index>.<c_subindex>,<parameter>=<value><*CKSUM><CR><LF>` or `PFAL,<c_type>.<c_index>.<c_subindex>,<parameter>=<value><CR><LF>` For example: `1)$PFAL,SYS.Device.Reset*31<CR><LF>` `2)$PFAL,Sys.Trigger5<CR><LF>` `3)PFAL,IO4.Set=hpulse,2000<CR><LF>` `4)PFAL,IO5.Set=cyclic,500,500;MSG.Send.Serial,0,"IO5 is flashing"<CR><LF>` `5)PFAL,Sys.Timer0.Start=single,5000<CR><LF>` `6)PFAL,Cnf.Set,DEVICE.NAME="myFOX3"<CR><LF>` `7)PFAL,Cnf.Set,AL0=IO.e2=redge:IO6.Set=cyclic,2000,500<CR><LF>` The example 6 above signifies that the device name is specified to *"myFOX3"*. The example 7 above signifies that the digital output 0 will flash cyclic, if a rising edge event on the input 1 is detected.

Warning: ***It is suggested always to use a checksum inside the PFAL commands. Therefore, If the checksum <*CKSUM> will not be used, please prevent using OF ANY asterisk "*" character inside the value of PFAL commands (especially 3 characters before the <CR><LF>).***

3.1.1. Command types

The commnad type <c_type> is used to separate the huge amount of commands to different types. The following command types are currently vailable.

Table 3-2 Supported command types <c_type>

Types	Definition
ALARM	Provides alarm info, clear and reload command.
SYS	Accomplishes a predefined set of system tasks such as: ◆ System management tasks, including Reset, Shutdown/power management etc. ◆ Initialization/interruption of system processes, including Timers, Counters etc.
CNF	The operating firmware provides commands that can set/change or read parameters. Based on the parameter settings some events will occur.
IO	Accomplishes a predefined set of system of system peripheral commands allowing access to the digital and analog, input and output pins.
GPS	Accomplishes a predefined set of GPS commands including navigation, history logging and geo-fencing data.
EcoDrive	Provides commands to evaluate the cost-effectiveness and estimate the fuel consumption of vehicles.
GSM	Accomplishes a predefined set of GSM commands including SMS, voice calls, GPRS services etc.
TCP	Accomplishes a predefined set of TCP connection commands including connecting, disconnecting, and sending of TCP packets to the predefined address of remote server etc.
WLAN	Provided commands for scanning networks, sending data, disconnecting from a server and WLAN access point.
MSG	Accomplishes a predefined set of output messages (GPS protocols) allowing information to be transmitted across serial interface, CSD network or Internet (TCP).

3.1.2. Aliases

This PFAL command allows to specify alias names for all available <c_type>, <c_index>, and <c_subindex>.

The following alias names are defined as default:

- **Sys** is the alias name of **System**.
- **Cfg** is the alias name of **Config**.

If an alias name has been specified it can freely used instead of the original word (or number). To sepcify alias names please refer to chapter *5.5.*

Notes

- To avoid misinterpretation, please assure that no equal or very similar aliases are used inside the same command type/index.
- Also avoid starting aliases with numbers as they might be misinterpreted original index numbers.

3.1.3. Identifiers (Optional)

Table 3-3 Identifier syntaxes

Syntax1	$PFAL:id<idtxt>,<commands>*<CS><CRLF>
Syntax2	$PFAL:id<idtxt>,<commands><CS><CRLF>

Syntax3	PFAL:id<idtxt>,<commands*<CS><CRLF>
Syntax4	PFAL:id<idtxt>,<commands><CS><CRLF>

<idtxt>

Specifies an optional text, which does not contain any comma (,). The case sensitive text will be returned within the response to that PFAL command.

<commands>

Comprises onw or more PFAL commands.

<CS>

NMEA Checksum (see description of the *<*CKSUM>*).

<CRLF>

Carriage return Line Feed (ASCII Code 13 10 (without any spaces) (hexadecimal: 0x0D 0x0A)

3.1.4. Response structure

A configuration report is presented in text format, which conatins parameters listed in the table below.

Table 3-4 Response messages structure

Response messages if a single command is executed in one line	**Structure**
Example	**$PFAL,Cnf.Set,DEVICE.NAME=myFOX3<CRLF>**
From read messages	$<cmd><CR><LF> $report of executed parameter<CR><LF> $SUCCESS or $ERROR<CR><LF> $<end>
Example 1	$<Cnf.Get> $NAME=unnamed FOX3 $SUCCESS $<end>
From execution messages	$<cmd><CR><LF> $report of executed parameter<CR><LF> $SUCCESS or $ERROR<CR><LF> $<end>
Example 2	$<Cnf.Set> $Name written to flash $SUCCESS $<end>
Response messages if several commands executed in one line	**Structure**
Example	**$PFAL,Sys.Trigger0;Sys.Trigger5<CRLF>**
From read messages	$<Sys.Trigger0><CR><LF> $Trigger0=high<CR><LF> $<Sys.Trigger5><CR><LF> $Trigger5=low<CR><LF> $SUCCESS<CR><LF> $<end><CR><LF>

Response messages if identifiers are used	**Structure**
Example	**PFAL:id001,Sys.Trigger0<CRLF>**
From read messages	$Sys.Trigger0><CR><LF> $Trigger0=high <CR><LF> $SUCCESS<CR><LF> $<end:001><CR><LF>

Notes

- If the first command fails (i.e. used wrong or it cannot be executed correctly), the system will stop the execution of this command.
- If identifiers are submitted within PFAL commands, they will be returned inside PFAL responses

4: PFAL Commands

This chapter provides a complete list of the PFAL commands used to manage/administrate different parts of an application built on the system FOX3/-3G/-4G Series. Table 4-1 provides an index of all the PFAL commands with a summary and hyperlink to the command reference.

Note:

- *Commands listed in the table below cannot be sent to the target device in that form.*
- *Sending PFAL commands sequentially may fill up the command buffer of a device and/or result in communication errors / syntax errors due to corrupted or incomplete data.*
- *To prevent communication errors, it is recommended to send the next PFAL command after receiving an answer to the previous command.*

Only straight quotation marks (" ") should be used for PFAL commands, configuration parameters, and so on. Some quotation marks in the table below and ela might have been transformed to the directional (curly) quotation marks, which won't be accepted by the device.

Table 4-1 List of PFAL Commands

PFAL COMMANDS	MEANING	SECTION
ALARM COMMANDS		
Alarm.Info,<index>	Displays all set conditions related to the selected alarm index. Index ranges from 0 to 99.	*4.1.1.*
Alarm.Clear,<index>	Clears all settings of the specified alarm index. Index ranges from 0 to 99.	*4.1.2.*
Alarm.Reload	Reloads all alarms.	*4.1.3.*
Alarm. Call	Executes actions on the specified alarm index	*4.1.4.*
SYSTEM COMMANDS		
Sys.Security.Lock,"password"	Locks the AVL device	*4.2.1.1.*
Sys.Security.Unlock,"password"	Unlocks the AVL device	*4.2.1.2.*
Sys.Security.RemoveLock,"password"	Removes the lock on the AVL device	*4.2.1.3.*
Sys.Security.HideAlarm,"password"	Hides alarm configurations from being read out	*4.2.1.4.*
Sys.Security.UnhideAlarm,"password"	Removes the read protection of alarms	*4.2.1.5.*
Sys.RUpdate.Init	Initialize a remote firmware update	*4.2.2.1.*
Sys.RUpdate.Abort	Aborts a remote update and allows history being written again.	*4.2.2.2.*
Sys.RUpdate.DataMode,<msg_input>	Define firmware upgrade channel & continue upgrading	*4.2.2.3.*
Sys.RUpdate.Finish	Finish remote update	*4.2.2.4.*
Sys.WebUpdate.Start	Starts an activated web update	*4.2.3.1.*
Sys.WebUpdate.Stop	Stops a started web update	*4.2.3.2.*
Sys.WebUpdate.State	Reads the state of the started web update	*4.2.3.3.*
Sys.WebUpdate.SetCertificate	Sets certificate for the web update connection	*4.2.3.4.*
Sys.WebUpdate.ClearCertificate	Clears the certificate used by TLS library	*4.2.3.5.*
Sys.WebUpdate.ShowCertificate	Shows the certificate used by TLS library	*4.2.3.6.*

PFAL COMMANDS	MEANING	SECTION
Sys.Device.Reset	Resets the system	*4.2.4.1.*
Sys.Device.Shutdown	Shuts down the system immediately	*4.2.4.2.*
Sys.Device.FactoryReset	Resets the user-configuration to factory default	*4.2.4.3.*
Sys.Device.ChargeSleep=<Wakeup_condition>	Sends device into sleep and activates charging while sleeping	*4.2.4.4.*
Sys.Device.Sleep=<Wakeup_condition>	Sets the system into the sleep mode until the specified wakeup condition is detected.	*4.2.4.5.*
Sys.Device.Doze=<Wakeup_condition>	Sends device into the doze mode	*4.2.4.6.*
Sys.Device.CfgUpdateMode	Saves the reconfigured alarm settings	*4.2.4.7.*
Sys.Device.ClearConfig	Erases the current configuration settings	*4.2.4.8.*
Sys.Device.BackupReset	Factory reset and restore backup	*4.2.4.9.*
Sys.Device.RestoreBios	Upgrades the on-board BIOS	*4.2.4.10.*
Sys.Device.ClearAGPS	Erases the existing AGPS file from the device	*4.2.4.11.*
Sys.Device.PwrManagement.Set,<Options>,<Off_Time>,<On_Time>	Sets the power management options for deep sleep mode - for ROCK device only	*4.2.4.12.*
Sys.Device.PwrManagement.EnterSleep	Enters the ROCK device into deep sleep mode	*4.2.4.13.*
SYS.Device.SetSerialID,"SerialID"	Sets the serial id of the device	*4.2.4.14.*
Sys.Power.Mode	Sets device into a low power mode	*4.2.5.1.*
Sys.SetTime	Sets system time	*4.2.6.1.*
Sys.GetTime	Gets system time	*4.2.6.2.*
Sys.1Wire.Enable	Enables requests on the connected 1-Wire devices	*4.2.7.1.*
Sys.1Wire.Disable	Disables requests on the connected 1-Wire devices	*4.2.7.2.*
Sys.1Wire.Devices	Lists IDs of the connected 1-Wire devices	*4.2.7.3.*
Sys.1Wire.Temperature	Lists temperatures of the connected 1-Wire devices.	*4.2.7.4.*
Sys.GSM.<mode>	Powers on/off the GSM module	*4.2.8.1.*
Sys.GSM.Reset	Resets GSM engine	*4.2.8.2.*
Sys.GPS.<mode>	Powers on/off the GPS engine	*4.2.9.1.*
Sys.GPS.Reset	Resets the GPS receiver.	*4.2.9.2.*
Sys.Timer<index>.Configure=<mode>,<timeout>	Configures a system timer	*4.2.10.1.*
Sys.Timer<index>.Start=<timer_settings>	Starts/restarts a system timer	*4.2.10.2.*
Sys.Timer<index>.Stop	Stops a running timer	*4.2.10.3.*
Sys.Timer<index>.Pause	Pauses (suspends) a running timer	*4.2.10.4.*
Sys.Timer<index>.Resume	Restarts the execution of a paused timer	*4.2.10.5.*
Sys.Timer<index>.Arm	Arms an initialized and disarmed timer	*4.2.10.6.*
Sys.Timer<index>.Disarm	Disarms an initialized and armed timer	*4.2.10.7.*
Sys.Timer<index>.Erase	Erases the configuration of a timer	*4.2.10.8.*
Sys.Timer<index>.Save<storage_index>	Saves a timer state to a storage slot	*4.2.10.9.*
Sys.Timer<index>.Load<storage_index>	Loads the saved timer state from a storage slot	*4.2.10.10.*
Sys.Timer<index>.State	Reads the state of a used timer	*4.2.10.11.*

PFAL COMMANDS	MEANING	SECTION
Sys.Trigger<index>=<state_type>	Activates/deactivates a system trigger	*4.2.11.1.*
Sys.Trigger<index>	Reads the current trigger state	*4.2.11.2.*
Sys.Trigger<index>.Save<storage_index>	Saves the state of trigger to a storage slot	*4.2.11.3.*
Sys.Trigger<index>.Load<storage_index>	Loads a saved trigger from a storage slot	*4.2.11.4.*
Sys.Counter<index>.Set=<value>	Sets the value of a counter	*4.2.12.1.*
Sys.Counter<index>.Increment=<inc_value>	Increments the existing value of a counter	*4.2.12.2.*
Sys.Counter<index>.Decrement=<dec_value>	Subtracts the existing value of a counter	*4.2.12.3.*
Sys.Counter<index>.Add	Adds a value to a counter	*4.2.12.4.*
Sys.Counter<index>.Sub	Subtracts a value from a counter	*4.2.12.5.*
Sys.Counter<index>.Save<slot_id>	Saves the state of a counter to a storage slot	*4.2.12.6.*
Sys.Counter<index>.Load<slot_id>	Loads a saved counter from the storage slot	*4.2.12.7.*
Sys. Counter<index>.Clear	Sets a specified counter to 0	*4.2.12.8.*
Sys.nvCounter<index>.State	Reads the state of a used non-volatile counter	*4.2.13.1.*
Sys.nvCounter<index>.Clear	Sets a specified non-volatile counter to 0	*4.2.13.2.*
Sys.nvCounter<index>.Set=<value>	Assigns a value to the nvCounter	*4.2.13.3.*
Sys.nvCounter<index>.Increment=<inc_value>	Increments the existing value of a non-volatile counter	*4.2.13.4.*
Sys.nvCounter<index>.Decrement=<dec_value>	Subtracts the existing value of a non-volatile counter	*4.2.13.5.*
Sys.Macro<index>	Activates a configured macro	*4.2.14.1.*
Sys.CAN.Enable	Activates the CAN interface.	*4.2.15.1.*
Sys.CAN.Disable	Deactivates the CAN interface and all CAN dependent commands (except Can.Enable)	*4.2.15.2.*
Sys.CAN.Msg.Add,<msg_type>,<msg_identifier>[,<mask>]	Adds a CAN message to the system.	*4.2.15.3.*
Sys.CAN.Msg.Remove,<msg_type>,<msg_identifier>	Removes a CAN Message from the system	*4.2.15.4.*
Sys.CAN.Msg.Info	Shows a list of all active CAN Messages.	*4.2.15.5.*
Sys.CAN.Var.Add,<variable_slot>,<variable_type>,<notification>,<msg_type><msg_identifier>,<start_byte>,<start_bit>,<stop_byte>,<stop_bit>,<byte_order>[,<src_offset>,<multiplier>,<divider>,<dst_offset>]	Adds a CAN variable to one of up to 50 CAN Variable slots.	*4.2.15.6.*
Sys.CAN.Var.Remove,<variable_slot>	Removes the CAN Variable from the given slot.	*4.2.15.7.*
Sys.Can.Var.Info,<variable_slot>	Shows settings and current value of CAN Variable within the given slot.	*4.2.15.8.*
Sys.CAN.GetTimings	Shows CAN hardware timing	*4.2.15.9.*
Sys.CAN.FMS.<mode>	Enables or disables CAN FMS functionalities	*4.2.15.10.*

PFAL COMMANDS	MEANING	SECTION
Sys.Can.OBDII.Enable[,<format>]	Enables the OBDII functionality and defines the frame format (identifier) on the first CAN interface.	*4.2.15.11.*
Sys.CAN.OBDII.Disable	Disables the OBDII functionality on the first CAN interface.	*4.2.15.12.*
Sys.CAN.Timeout,<timeout>	Defines timeout for Idle/Active events on the first CAN interface.	*4.2.15.13.*
Sys.CAN.OBDII.DTCrq	Requests a diagnostic trouble code message (DTC)	*4.2.15.14.*
Sys.Can.DTCO.FMS.<mode>	Enables or disables tachograph reading via FMS on the first CAN port.	*4.2.15.15.*
Sys.Can.DTCO.SendAPDU,<TA>,"codes"	Transfers messages (requests) to a tachograph.	*4.2.15.16.*
SYS.CAN.CANopen.enable[,<hex_node_id>]	Enables CANopen on the main interface (8 pin connector). Node ID configurable is with <hex_node_id>- Hex number 0 .. 7f	*4.2.15.17.*
SYS.CAN.CANopen.disable	Disable CANopen on the main interface (8-pin connector)	*4.2.15.18.*
SYS.CAN.CANopen.cmd,"<CIA309-3 gateway command>"	Executes the CANopen gateway function on the main interface.	*4.2.15.19.*
Sys.CANB.Enable	Activates the second **CAN** interface.	*4.2.16.1.*
Sys.CANB.Disable	Deactivates the second **CAN** interface and all **CAN** dependent commands (except **Can.Enable**)	*4.2.16.2.*
Sys.CANB.Msg.Add,<type>,<identifier>[<mask>]	Adds a second **CAN** message to the system.	*4.2.16.3.*
Sys.CANB.Msg.Remove,<msg_type>,<identifier>	Removes a second **CAN** Message from the system	*4.2.16.4.*
Sys.CANB.Var.Add,<variable_slot>,<variable_type>,<notification>,<msg_type><msg_identifier>,<start_byte>,<start_bit>,<stop_byte>,<stop_bit>,<byte_order>[<src_offset>,<multiplier>,<divider>,<dst_offset>]	Adds a second **CAN** variable to one of 25 **CAN** Variable slots.	*4.2.16.5.*
Sys.CANB.Var.Remove,<variable_slot>	Removes the second **CAN** Variable from the given slot.	*4.2.16.6.*
Sys.CANB.GetTimings	Shows second CAN hardware timing	*4.2.16.7.*
Sys.CANB.FMS.<mode>	Enables or disables second CAN FMS functionalities	*4.2.16.8.*
Sys.CANB.OBDII.Enable[,<format>]	Enables OBDII functionality and defines the base frame format (identifier) on the second CAN interface.	*4.2.16.9.*
Sys.CANB.OBDII.Disable	Disables the OBDII functionalities on the second CAN interface.	*4.2.16.10.*
Sys.CANB.Timeout,<timeout>	Defines timeout for Idle/Active events on the second CAN interface.	*4.2.16.11.*
Sys.CANB.OBDII.DTCrq	Requests an Diagnostic Trouble Code message (DTC) from the connected bus	*4.2.16.12.*
Sys.CanB.DTCO.FMS.<mode>	Enables or disables tachograph reading via FMS on the second CAN port.	*4.2.16.13.*

PFAL COMMANDS	MEANING	SECTION
SYS.CANB.CANopen,enable[,<hex_node_id>]	Enables CANopen on IOBOX-CAN or IOBOX-WLAN Node ID is configurable with <hex_node_id>- Hex number 0 .. 7f	*4.2.16.14.*
SYS.CANB.CANopen,disable	Disable CANopen on IOBOX-CAN or IOBOX-WLAN	*4.2.16.15.*
SYS.CANB.CANopen.cmd,"<CIA309-3 gateway command>"	Executes the CANopen gateway function on the second interface.	*4.2.16.6.*
Sys.WLAN.< >	Enables or disables WLAN module on IOBOX-WLAN.	*4.2.17.1.*
Sys.Lua.Start	Starts and runs a Lua script that is available in the device.	*4.2.18.1.*
Sys.Lua.Stop	Stops a Lua script that is running.	*4.2.18.2.*
Sys.Lua.Dump	Reads Lua script source code available on device	*4.2.18.3..*
Sys.Lua.Lock,<"password">	Protects Lua script from reading	*4.2.18.4.*
Sys.Lua.UnLock,<"password">	Unlocks a password-protected Lua script	*4.2.18.5.*
Sys.Lua.Dump,<"password">	Reads a password-protected Lua script	*4.2.18.6.*
Sys.Lua.Clear	Clears the Lua script that is available on the device	*4.2.18.7.*
Sys.LUA.Event,<id>,<"text">	Generates a Lua event when it is executed	*4.2.18.8.*
Sys.Lua.Start[,<"script.lua">]	Loads a specific Lua script	*4.2.18.9.*
Sys.Lua.Clear[,<"script.lua">]	Deletes a specific Lua script	*4.2.18.10.*
Sys.Lua.Info[,<"script.lua">]	Comment of specific Lua script	*4.2.18.11.*
Sys.Lua.Write[,<"script.lua">]	Writes a specific Lua script.	*4.2.18.12.*
Sys.BLE.Scan	Scans for new BLE devices	*4.2.19.1.*
Sys.BLE.List	Lists the available BLE devices	*4.2.19.4.*
Sys.UserEvent<index>	Creates a user-event for specific application requirements.	*4.2.22.1.*
Sys.Whitelist.Info	Counts and shows the number of entries in the list	*4.2.23.1.*
Sys.Whitelist.Show	Shows the contents of the entries defined in the list	*4.2.23.2.*
Sys.Whitelist.Clear	Erases all entries from the list	*4.2.23.3.*
Sys.Whitelist.Set<index>=<"text">	Adds or edits an entry to/in the list	*4.2.23.4.*
Sys.Whitelist.Get<index>=<"text">	Reports the text assigned to the specified index	*4.2.23.5.*
Sys.Bat.Voltage	Queries the current battery voltage.	*4.2.24.1.*
Sys.Bat.ChargeMode	Enables or disables the battery charging.	*4.2.24.2.*
Sys.Bat.ChargeState	Gets the current battery state.	*4.2.24.3.*
Sys.Bat.Mode	Manages the battery charging circuit.	*4.2.24.4.*
SYS.ModBus.Enable	Enables access to ModBus for serial	*4.2.25.1.*
SYS.ModBus.Disable	Diasbles access to ModBus	*4.2.25.2.*
SYS.ModBus.ReadRegister	Reads register value from the ModBus device	*4.2.25.3.*
SYS.ModBus.Poll	Read periodically register values from the connected ModBus device	*4.2.25.4.*
CNF COMMANDS		
Cnf.Set,<parameter_name=value>	Enables to set up or change the device configuration settings.	*4.3.1.*

PFAL COMMANDS	MEANING	SECTION
Cnf.Get,<parameter_name>	Enables to read out the device configuration settings.	*4.3.2.*
Cnf.Clear,<parameter_name>	Clears the available configuration settings of the set parameter name.	*4.3.3.*
Cnf.ShowUser	Reads the configuration settings of all modified/added parameters.	*4.3.4.*
Cnf.ShowDefault	Reads the factory default settings.	*4.3.5.*
Cnf.Show	Reads the settings of all used parameters.	*4.3.6.*
Cnf.Search,<parameter_name>	Searches for a parameter name.	*4.3.7.*
Cnf.Backup	Backup user configuration.	*4.3.8.*
Cnf.EraseBackup	Erases the backed up user configuration settings	*4.3.9.*
Cnf.Restore	Restore user configuration.	*4.3.10.*
CNF.Load,<"/sys/file.txt">	PFAL command with which the existing config can be overwritten with a file from flash.	*4.3.13.*
IO COMMANDS		
IO<index>.Set=<config_type>	Specifies the output behaviour.	*4.4.2.*
IO<index>.Get	Returns the current function and level of IO	*4.4.3.*
IO<index>.GetDI	Gets level of the specified digital output (DI) IO.	*4.4.4.*
IO<index>.GetAI	Gets level of the specified analog output (AI) IO.	*4.4.5.*
IO<index>.GetDO	Gets level of the specified digital output (DO) IO.	*4.4.6.*
IO<index>.Config	Configures/changes the functionality and/or the behaviour of the specified IO.	*4.4.7.*
IO<index>.Calibrate	Calibrates the offset or gain of analog input (AI).	*4.4.8.*
IO<index>.Info	Shows the current configuration and all relevant parameter of the specified IO.	*4.4.10.*
IO<index>.Counter.Info	Returns the current state and the counter value of the specified IO.	*4.4.11.1.*
IO<index>.Counter.Start	Starts the IO specified hardware.	*4.4.11.2.*
IO<index>.Counter.Set	Sets the value of the specified hardware counter.	*4.4.11.3.*
GPS COMMANDS		
GPS.Nav.Position<buffer_index>	Reads the distance of the device from a stored location	*4.5.1.1.*
GPS.Nav.Position<buffer_index>=<type>	Saves temporarily a device location or clears the data that exists in the buffer index	*4.5.1.2.*
GPS.Nav.Position<buffer_index>=save<storage_index>	Moves the GPS data from the buffer and stores it to a storage index.	*4.5.1.3.*
GPS.Nav.Position<buffer_index>=load<storage_index>	Loads the GPS data from storage to buffer index for temporarily use.	*4.5.1.4..*
GPS.Nav.Distance	Reads the distance from a start point	*4.5.1.5.*
GPS.Nav.Distance=<value>	Sets/resets the distance to a user defined value	*4.5.1.6.*
GPS.Nav.DeltaDistance	Retrieves the distance in metres from current delta distance counter.	*4.5.1.9.*
GPS.Nav.DeltaDistance=<value>	Sets the distance counter to a fixed value.	*4.5.1.10.*

PFAL COMMANDS	MEANING	SECTION
GPS.Nav.Distance 2	Reads the distance2 from a start point	*4.5.1.11..*
GPS.Nav.Distance2=<value>	Sets/resets the distance2 to a user defined value	*4.5.1.12.*
GPS.Nav.SetHeadingTolerance=<value>	Defines the tolerance for heading feature.	*4.5.1.13.*
GPS.Nav.ResetHeading	Resets heading to the currently used GPS position.	*4.5.1.14.*
GPS.Nav.SetHeading2Tolerance=<value>	Defines the tolerance for heading2 feature.	*4.5.1.15.*
GPS.Nav.ResetHeading2	Resets heading2 to the currently used GPS position.	*4.5.1.16.*
GPS.Nav.SaveLastValid	Saves last valid position, if no GPS-fix valid.	*4.5.1.17.*
GPS.Nav.GNSS	Enables/disables GNSS operation	*4.5.1.19.*
GPS.History.Write,<add_prot_to_memory>,<"text">	Records a GPS position data into the history memory	*4.5.2.1.*
GPS.History.Clear	Clears the history memory.	*4.5.2.2.*
GPS.History.GetStart	Reads the oldest date stored in the history memory.	*4.5.2.3.*
GPS.History.SetRead,<s_date>,<s_time>-<e_date>,<e_time>	Selects the number of records from the history memory to be downloaded.	*4.5.2.4.*
GPS.History.SetRead,<start_index>-<end_index>	Selects the number of history indexes to be downloaded.	*4.5.2.5.*
GPS.History.Read	Downloads the selected records from the history memory.	*4.5.2.6.*
GPS.History.Push	Downloads all selected history at once.	*4.5.2.7.*
GPS.Geofence.Park.Set	Places/activates a virtual circular fence around vehicle (Park area).	*4.5.3.1.*
GPS.Geofence.Park.Remove	Disables an activated park area.	*4.5.3.2.*
GPS.Geofence.GeoState,<geo_id>	Reads the state of a defined geo-fence.	*4.5.3.3.*
GPS.Geofence.AreaState,<area_id>	Reads the state of a defined area.	*4.5.3.4.*
GPS.MultiGeofence.Info	Shows the number of multi-geofences being in use	*4.5.4.1.*
GPS.MultiGeofence.Clear	Clears the list of multi-geofences	*4.5.4.2.*
GPS.MultiGeofence.GetWP	Gets the position and radius of specific multi-geofence	*4.5.4.3.*
GPS.MultiGeofence.SetWP	Sets the position and radius of specific multi-geofence	*4.5.4.4.*
GPS.MultiGeofence.Text,<lat>,<lon>	Test if a specific GPS position is inside the set of multi geofences	*4.5.4.5.*
GPS.WPGeofence.Info	Shows information about the currently used set of waypoints and their configuration settings	*4.5.5.1.*
GPS.WPGeofence.Clear	Erases the entries in the waypoints list	*4.5.5.2.*
GPS.WPGeofence.GetWP,<id>	Gets the position and radius of the specified waypoint	*4.5.5.3.*
GPS.WPGeofence.SetMode2D	Deletes the entries in the waypoint list and sets up the mode of waypoints to 2D, if the mode is 3D (three-dimensional)	*4.5.5.4.*
GPS.WPGeofence.SetMode3D	Deletes the entries in the waypoint list and sets up the mode of waypoints to 3D, if the mode is 2D (two-dimensional)	*4.5.5.5.*
GPS.WPGeofence.SetWP,<id>,<lat>,<lon>,<alt>,<radius>	Adds a new entry into the waypoint list.	*4.5.5.6.*

PFAL COMMANDS	MEANING	SECTION
GPS.WPGeofence.Test,<id>,<lat>,<lon>,<alt>,<radius>	Tests if the specified position is within the corridors of the currently waypoint list.	*4.5.5.7.*
ECO DRIVE COMMANDS		
Ecodrive.TripStart	Starts a new trip with Eco-DRIVE-GPS	*4.6.1.*
Ecodrive.TripStop	Stops a started trip with Eco-DRIVE-GPS	*4.6.2.*
Ecodrive.CurrentTrip	Reports data from the current EcoDrive trip	*4.6.3.*
Ecodrive.LastTrip	Reports data from the last EcoDrive trip	*4.6.4.*
GSM COMMANDS		
GSM.PIN=<"pin">	Enters the PIN number of the used SIM card.	*4.7.1.1.*
GSM.PUK=<"puk">,<"pin">	Enters the PUK and PIN numbers.	*4.7.1.2.*
GSM.IMEI	Reads the serial identification number of the product.	*4.7.1.3.*
GSM.IMSI	Returns the product serial number identification	*4.7.1.4.*
GSM.ICCID	Returns the integrated circuit card identification of the SIM card	*4.7.1.5.*
GSM.OwnNumber	Reads the caller's phone number.	*4.7.1.6.*
GSM.Balance	Reads the account information of the used SIM card.	*4.7.1.7.*
GSM.USSD	Performs an USSD call and return its answer.	*4.7.1.8.*
GSM.MCC	Reads out the current mobile country code information of the operator the device is registered to.	*4.7.1.9.*
GSM.Band	Specifies the GSM band used by the device.	*4.7.1.10.*
GSM.CBM.Add,<message_slot>,<cbm_id>	Adds a CBM message to an empty message slot	*4.7.2.1.*
GSM.CBM.Remove,<message_slot>	Removes a previously added CBM message from the message slot	*4.7.2.2.*
GSM.CBM.Info	Queries CBM information	*4.7.2.3.*
GSM.VoiceCall.Dial,<"p_number">	Performs a GSM Voice call.	*4.7.3.1.*
GSM.VoiceCall.Accept	Accepts an incoming voice call.	*4.7.3.2.*
GSM.VoiceCall.Hangup	Hangs-up an active voice call.	*4.7.3.3.*
GSM.VoiceCall.SendDTMF,<duration>,<"dtmf_tones">	Sends specified DTMF tones while inside a voice call	*4.7.3.4.*
GSM.Audio.ActiveProfile	Selects and activates an audio profile.	*4.7.4.1.*
GSM.Audio.ShowProfile	Shows all details of the specified audio profile.	*4.7.4.2.*
GSM.Audio.SaveProfileAs	Stores the currently used audio settings to a profile	*4.7.4.3.*
GSM.Audio.DeleteProfile	Erases a stored profile.	*4.7.4.4.*
GSM.Audio.EchoCancel	Activates or deactivates echo cancellation for a handsfree speaker/microphone	*4.7.4.5.*
GSM.Audio.SideTone	Activates or deactivates handset side tone	*4.7.4.6.*
GSM.Audio.SpeakerMute	Activates or deactivates speaker output	*4.7.4.7.*
GSM.Audio.SpeakerGain	Sets speaker gain (loudness)	*4.7.4.8.*
GSM.Audio.MicrophoneMute	Activates or deactivates microphone	*4.7.4.9.*

PFAL COMMANDS	MEANING	SECTION
GSM.Audio.HandsfreeMicroGain	Sets microphone gain (loudness) for handsfree microphone	*4.7.4.10.*
GSM.Audio.HandsetMicroGain	Sets microphone gain (loudness) for handset microphone.	*4.7.4.11.*
GSM.Audio.AudioRingPath	Selects the path to which ring signals (i.e. voice input/ output) are directed	*4.7.4.12.*
GSM.Audio.RingTone	Selects the used ring tone for incoming calls	*4.7.4.13.*
GSM.Audio.RingGain	Sets the gain (loudness) for ring tones	*4.7.4.14.*
GSM.Audio.AudioPath	Selects the path for regular audio signals (i.e. voice)	*4.7.4.15.*
GSM.Audio.SoundMode	Selects a global sound mode for the device	*4.7.4.16.*
GSM.SMS.Send,<“p_number”>,<protocols>,<“text”>	Sends a SMS to the defined phone number.	*4.7.5.1.*
GSM.SMS.SendRaw,<“p_number”>,<protocols>,<“text”>	Sends a SMS in raw format to the defined phone number.	*4.7.5.2.*
GSM.SMS.Inbox.Clear	Clears all inbox SMS messages (SMS memory for incoming messages).	*4.7.5.4.*
GSM.SMS.Inbox.State	Reads all inbox SMS messages.	*4.7.5.5.*
GSM.SMS.Outbox.Clear	Clears all outbox SMS messages (SMS memory for outgoing messages).	*4.7.5.7.*
GSM.SMS.Outbox.State	Reads all outbox SMS messages.	*4.7.5.8.*
GSM.DataCall.Send,<protocols>,<“text”>	Sends messages to a GSM modem via an established data call.	*4.7.6.1.*
GSM.DataCall.Accept	Accepts an incoming Data call.	*4.7.6.2.*
GSM.DataCall.Hangup	Hangs-up an active voice call.	*4.7.6.3.*
GSM.GPRS.Connect	Connects device to GPRS network.	*4.7.7.1.*
GSM.GPRS.Disconnect	Disconnects device from GPRS network.	*4.7.7.2.*
GSM.GPRS.State	Reads the GPRS state.	*4.7.7.3.*
GSM.GPRS.Traffic=<complete>,<incoming>,<outgoing>	Sets or reads the GPRS traffic counter.	*4.7.7.4.*
GSM.SetExternalAntenna	Switches to external antenna	*4.7.8.1.*
GSM.SetInternalAntenna	Switches to internal antenna	*4.7.8.2.*
TCP COMMANDS	Outputs the specified protocols and text in raw format to the selected serial port.	
TCP.Client.Connect	Performs a TCP connection to the remote server.	*4.8.1.1.*
TCP.Client.Disconnect	Disconnect the device from the remote server.	*4.8.1.2.*
TCP.Client.State	Reads the TCP connection state.	*4.8.1.3.*
TCP.Client.Send,<protocols>,<“text”>	Sends a TCP packet to the connected remote server.	*4.8.1.4.*
TCP.Client.ClearSendBuffer	Clears the outgoing TCP buffer	*4.8.1.5.*
TCP.Client.RxKey=<“key”>	AES128 Encryption for incoming packets using the key	*4.8.1.6.*
TCP.Client.TxKey=<“key”>	AES128 Encryption for outgoing packets using the key	*4.8.1.7.*
TCP.Client.FlushSendBuffer	Transfers buffered data immediately	*4.8.1.8.*

PFAL COMMANDS	MEANING	SECTION
TCP.Client.SetCertificate	Set certificate used by TLS library. The certificate must be sent after the command and the transmission is finished by "<CR><LF>"	*4.8.1.9.*
TCP.Client.ShowCertificate	Shows the used TLS certificate	*4.8.1.10.*
TCP.Storage.Dispatch	Moves the currently stored information inside the TCP storage to the outgoing TCP buffer.	*4.8.2.1.*
TCP.Storage.Clear	Clears the contents of the created TCP storage.	*4.8.2.2.*
TCP.Storage.AddProtocol,<protocol>,<"text”>	Writes the specified protocols and/or user text to the TCP storage.	*4.8.2.3.*
TCP.Storage.AddRecord,<protocol>,<“text”>	Appends a binary data frame to TCP storage.	*4.8.2.4.*
TCP.SMTP.Send,<email_address>,<protocols>,<“text”>	Sends an email to the specified SMTP server	*4.8.3.1.*
TCP.SMTP.SendRaw,<email_address>,<protocols>,<“text”>	Sends an email in raw format to the specified SMTP server	*4.8.3.2.*
TCP.Client2.SetCertificate	Set certificate used by TLS library for the second TCP connection The certificate must be sent after the command and the transmission is finished by "<CR><LF>"	*4.8.4.1.*
TCP.Client2.ShowCertificate	Shows the used TLS certificate for the second TCP connection	*4.8.4.2.*
TCP.Client2.Connect	Opens a second TCP connection	*4.8.4.4.*
TCP.Client2.Disconnect	Closes the second TCP connection	*4.8.4.5.*
TCP.Client2.State	Shows the state of the second TCP connection	*4.8.4.6.*
TCP.Client2.Send,<protocols>,<“text”>	Sends out a TCP packet from the second TCP connection	*4.8.4.7.*
TCP.Client2.ClearSendBuffer	Clears the outgoing TCP buffer for the second TCP connection	*4.8.4.8.*
TCP.Client2.FlushSendBuffer	Flushes the outgoing buffer for the second TCP connection	*4.8.4.9.*
TCP.Client2.TxKey=<"key">	AES128 Encrypts the outgoing TCP packet on the second TCP connection	*4.8.4.10.*
TCP.Client2.RxKey=<"key">	AES128 Decrypts the incoming TCP packet on the second TCP connection	*4.8.4.11.*
TCP.MQTT.Connect	Performs a TCP connection to a MQTT server / broker	*4.8.5.1.*
TCP.MQTT.Disconnect	Disconnects from the used MQTT server / broker	*4.8.5.2.*
TCP.MQTT.State	Returns MQTT connection state	*4.8.5.3.*
TCP.MQTT.Send,”<topic>@<message>”	Sends a message to the MQTT server / broker	*4.8.5.4.*
TCP.MQTT.Clear	Clears the outgoing MQTT buffer	*4.8.5.5.*
TCP.MQTT.SetRootCA	Set the server certificate for the MQTT connection	*4.8.5.6.*
TCP.MQTT.GetRootCA	Shows the server certificate for the MQTT connection	*4.8.5.7.*
TCP.MQTT.SetCertificate	Set the client certificate for the MQTT connection	*4.8.5.8.*
TCP.MQTT.GetCertificate	Shows the client certificate for the MQTT connection	*4.8.5.9.*

PFAL COMMANDS	MEANING	SECTION
TCP.MQTT.SetPrivateKey	Set the client private key for the MQTT connection	*4.8.5.10.*
TCP.MQTT.GetPrivateKey	Shows the client private key for the MQTT connection	*4.8.5.11.*
WLAN COMMANDS		
WLAN.Scan	Scans for new WLAN networks	*4.9.1.*
WLAN.Connect	Connects to a WLAN network profile	*4.9.2.*
WLAN.Send,<protocol>,<"text">	Sends protocols and user text via WLAN to the server	*4.9.3.*
WLAN.Client.Disconnect	Disconnects from a TCP server	*4.9.4.*
WLAN.Disconnect	Disconnects from the WLAN network	*4.9.5.*
WLAN.MAC	Shows the MAC address of the WIFI module.	*4.9.6.*
SEND COMMANDS		
MSG.Send.Serial<index>,<protocols>,<"text">	Outputs the selected protocols and text in the specified format to the specified serial port.	*4.10.1.1.*
MSG.Send.HexSerial<index>,<protocols>,<"text">	Outputs the selected protocols and text in hex format to the specified serial port.	*4.10.1.2.*
MSG.Send.USB,<protocols>,<"text">	Outputs the selected protocols and text in the specified format to the USB port.	*4.10.1.4.*
MSG.Send.RawUSB,<protocols>,<"text">	Outputs the specified protocols and text in raw format to the USB port.	*4.10.1.5.*
MSG.Send.RawFlashBuffer,<protocols>,<"text">	Outputs the specified protocols and text in raw format to the flash buffer.	*4.10.1.7.*
MSG.Send.CSD,<protocols>,<"text">	Transmits the selected protocols and text to the connected GSM modem via an established data call.	*4.10.1.8.*
MSG.Send.TCP,<protocols>,<"text">	Transmits the selected protocols and text to the connected server via TCP.	*4.10.1.9.*
MSG.Send.TCPBuffer,<protocols>,<"text">	Transmits the selected protocols and text to the TCP Buffer.	*4.10.1.12.*
MSG.Send.UDP,<protocols>,<"text">	Transmits the selected protocols and text to the connected server via UDP.	*4.10.1.14.*
MSG.Send.SMTP,<email_address><protocols>,<"text">	Sends the selected protocols and text via SMTP to an email address.	*4.10.1.15.*
MSG.Mode.<interface>=<out_sys_messages>,<mode>	Reads or forwards the in/out system messages from one interface to another.	*4.10.3.1.*
MSG.Version.Complete	Reads all version information of the target device.	*4.10.5.1.*
MSG.Version.Modules	Reads the modules versions of the target device.	*4.10.5.2.*
MSG.Version.BIOS	Reads the BIOS firmware version.	*4.10.5.3.*
MSG.Version.HardwareRev	Reads the hardware revision of the PCB.	*4.10.5.4.*
MSG.Version.Hardware	Reads the hardware version of the target device.	*4.10.5.5.*
MSG.Version.Software	Reads the software version of the target device.	*4.10.5.6.*
MSG.Info.ServerLogin	Identifies the device to the Lantronix Server.	*4.10.6.1.*
MSG.Info.Protocol,<protocols>,<"text">	Transmits the selected protocols to the sender.	*4.10.6.2.*
MSG.Info.Time	Displays the current system time.	*4.10.6.3.*

PFAL COMMANDS	MEANING	SECTION
MSG.Info.Alarm,<alarm_index>	Transmits the selected alarm to the sender.	*4.10.6.4.*
MSG.Feature	Reads all supported premium features	*4.10.7.1.*
MSG.Feature=<"code">	Activates using of premium features by codes	*4.10.7.2.*

4.1. Alarm

4.1.1. Alarm.Info – Displays all conditions defined in a specific alarm

Command Syntax	Alarm.Info,<index>
Examples	$PFAL,Alarm.Info,1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	🗝	🗝	🗝	🗝

Command description

This command allows you to display all conditions of the selected alarm and to show their current state *(whether they are true or false)*.

Parameter description

<index>

It specifies the alarm index, without leading "0", e.g. 0, 1, 8, 10, to be read. The index <index> is a number, which can be set to a value from:

Value	Meaning
0...99	Number of standard alarm indices
0...249[1]	Number of standard and extended alarm indices (premium feature).

Notes

This command can be used to check even most complex alarm configurations step by step to validate the desired behavior.

Events are always shown as “true” (even if there are several events inside an alarm - in reality such an alarm could never be executed).

4.1.2. Alarm.Clear – Clears and erases a specific alarm

Command syntax	Alarm.Clear,<index>
Examples	$PFAL,Alarm.Clear,1 $PFAL,Alarm.Clear,all

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
1	🗝	🗝	🗝	🗝

Command description

This command stops running the alarm of the specified index and erase all its settings inside.

Parameter description

<index>

It specifies the alarm index to be cleared or set it to “all” to clear all configured alarms available in the device.

Value	Meaning
0...99	Number of standard alarm indices
0...249[1]	Number of standard and extended alarm indices.
all	Clears and erases all alarm indices. When alarms are successfully erased, the device reports the number of the erased alarms.

4.1.3. Alarm.Reload – Reloads all alarms

Command Syntax	Alarm.Reload
Example	$PFAL,Alarm.Reload

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command can be used to update alarms after changing REPLACE settings.

Warning: ***Use of this command might result in inconsistent alarm behavior. While reloading alarms, no events will be generated. Therefore, correct behavior should be verified after using this command.***

Parameter description

None

4.1.4. Alarm.Call – Executes actions on the specified alarm index

Command Syntax	Alarm.Call,<index>
Example	$PFAL,Alarm.Call,1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	🗝	🗝	🗝	🗝

Command description

This command can be used to execute actions pre-configured in the specified alarm index. If the

alarm index is not available, the device returns an Error.

Parameter description

<index>

Specifies the index of the alarm to be executed.

Value	Meaning
0...99	Indices for standard alarms
100...249[1]	Indices for extended alarms (Premium-feature)

4.2. SYS

4.2.1. Sys.Security

The software inside the AVL devices has a built-in locking feature that prevents the unauthorized users from accessing the FOX3 as long as the **Unlock** command is not executed. The system lock is not released until the last locking password does not match exactly the unlock password on the same AVL device. An application may use this mechanism for the following purposes:

- To ensure that system does not complete any user request while the system lock is held.
- to prevent unauthorized users attempting to change the system configuration.

To remove permanently an applied lock, unlock the system first and then execute the **Sys.Security.RemoveLock** command.

Warning: ***There is no way to recover your device password if you forget it. If you have forgotten the password of your device, you won't be able to reset it. Resetting your device password can be done only by the device manufacturer (Lantronix).***

4.2.1.1. Sys.Security.Lock,"password" – Locks the device with a password

Command Syntax	Sys.Security.Lock,<"password">
Example	$PFAL,Sys.Security.Lock,"12345"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command allows you to define a password for execution of PFAL commands *(regardless of the message input – via TCP, SMS, CSD or Serial)*. It allows you to lock your device so that no other users may use your FOX3 device. None of PFAL commands is accepted by the device, if it is already locked.

Parameter description

<"password">

It consists of a string with a length of up to 50 characters to protect the AVL device from the unauthorized accesses.

4.2.1.2. Sys.Security.Unlock,"password" – Unlocks the device

Command Syntax	Sys.Security.Unlock,<"password">
Example	$PFAL,Sys.Security.Unlock,"12345"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command allows you to unlock a previously applied lock on a device. The password given must correspond with the existing user password specified for that device when the system has been locked. Unlocking the system enables the user to read/write the configuration and to execute PFAL commands.

Parameter description

<"password">

It consists of a string with a length of up to 50 characters to unlock the system. Use the last password specified with **Sys.Security.Lock,"password"** command.

4.2.1.3. Sys.Security.RemoveLock,"password" – Removes the lock and changes the password

Command Syntax	Sys.Security.RemoveLock,<"password">
Example	$PFAL,Sys.Security.RemoveLock,"12345"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is used to remove and change the current password to the new one. First unlock the system with **Sys.Security.Unlock,"password"** command and then remove it with **Sys.Security.RemoveLock,"password"**.

Parameter description

<"password">

Specifies the password (string type, max 50 characters) to hide the stored alarms in the device.

4.2.1.4. Sys.Security.HideAlarm,"password" – Hides alarms with a password from being read out

Command Syntax	Sys.Security.HideAlarm,<"password">
Example	$PFAL,Sys.Security.HideAlarm,"12345"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command can be used to hide the specified alarms starting with “AL" from being read out. Unless the specified password is known and entered within the command **Sys.Security.Unhide-Alarm, "password"**, no alarms can be read out by command.

Parameter description

<"password">

Specifies the password (string type, max 50 characters) to hide the stored alarms in the device.

4.2.1.5. Sys.Security.UnhideAlarm,"password" – Removes the read protection of alarms

Command Syntax	Sys.Security.UnhideAlarm,<"password">
Example	$PFAL,Sys.Security.UnhideAlarm,"12345"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command can be used to remove the alarm read protection. Alarms can be read from the configuration after this command was successful.

Parameter description

<"password">

Specifies the password (string type, max 50 characters) to unhide the stored alarms in the device. To remove the protection, the last password used for the Hide command must be specified.

4.2.2. Sys.RUpdate

All commands within this chapter enable to remotely upgrade the AVL devices to a new firmware version that is accessible over the Internet.

It is strongly recommended to use these commands with special care.

Notes

Except the information found in this document, Lantronix will not offer additional technical support for developing/implementing such web-based solutions.

4.2.2.1. Sys.RUpdate.Init – Initializes remote firmware update

Command Syntax	Sys.RUpdate.Init,<type>,<option>,<size>,<sectors>,<config>
Example	$PFAL,Sys.RUpdate.Init,FW_raw,new,890812,14,raw_cfg

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	×	×	×	×

Command description

Starts the firmware remote update or resumes a previous update process (also possible after a system restart). The command returns the number of sectors required for the update process.
To start transfer of firmware to the device you must define the channel from where the firmware data will be received.

Parameter description

<type>

It specifies the type of the remote firmware update. It can be set to the one of the following values:

Value	Meaning
FW_raw	Performs a remote update from uncompressed firmware data.
FW_cpr	Performs a remote update from compressed firmware data. The size of the compressed firmware is limited to 786 KB (maximal 12 sectors). A configuration can be stored within the compressed file, which will be unpacked then.
AID_raw	Performs a remote update for Aided GPS data (UBX only). The device does not perform a system reset.
file://path_to_file	Performs an update of a file in the internal file system of the device.

<option>

It specifies the option of the remote firmware update. It can have the following values:

Value	Meaning
new	Required for starting a new remote update. It erases previously transmitted data and erases e.g. possible history data stored in this region.
resume	Required for resuming a previous update. Erases previously transmitted device configuration, which must be transmitted again. Previously stored firmware data is not erased.

<size>

Specifies the exact length of the remote update firmware data. For **FW_raw**, this number specifies the length (in bytes) of the binary firmware data. The maximal length of a binary data is 65536 * 14 -1 bytes. For **FW_cpr**, this number specifies the length (in bytes) of the compressed firmware data. The file of the firmware 3.0.0 has a length of 631.908 Byte.

<sectors>

Specifies how large the new firmware is (how many sectors it uses in uncompressed format)- One sector is 64 KB.

For example: A firmware uses 14 sectors (e.g. 2.15.0), so 14 must be specified for <sectors>. Firmware 3.0.0 uses only 10 sectors, so 10 must be specified.

<config>

It specifies how to handle the configuration of the device during update process. See also

related documents in chapter *1.3.,* points [4] and [8]. The following values are allowed:

Value	Meaning
raw_cfg[1]	Erases the device configuration during the update process. A new (uncompressed) configuration can be transmitted via remote update. (see select sector for more details). **Config** can still be used until the remote update is finished; after finishing, the old configuration will be cleared
compressed_cfg[1]	A new configuration is stored within the compressed firmware. No separate **config** will be needed. It is not allowed to write uncompressed configuration data to the configuration sector if **compressed_cfg** is selected.
current_cfg	Uses the currently stored device configuration later. No separate configuration needs to be specified. An existing compressed configuration would be overwritten.

4.2.2.2. Sys.RUpdate.Abort – Aborts a started remote update and allows writing into the history again

Command Syntax	Sys.RUpdate.Abort
Example	$PFAL,Sys.RUpdate.Abort

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command aborts the remote update. The associated user interface is switched to command mode.

Parameter description

None

4.2.2.3. Sys.RUpdate.DataMode,<msg_input> – Defines the upgrade channel & continues upgrading

Command Syntax	Sys.RUpdate.DataMode,[<msg_input>]
Example	$PFAL,Sys.RUpdate.DataMode,TCP.Client

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✗
2	✓	✓	✓	✓

Command description

This command enters the target AVL device into the update data mode, which allows only entering of binary update commands. Inside this mode, no PFAL commands can be executed.

Parameter description

[<msg_input>]

This entry is optional. If specified, it defines in which channel the firmware-based packets will be received, otherwise the firmware-based packets are being received in that channel from where the command was sent. Now binary Update commands can be sent *(see next chapter)*.

Value	Meaning
Serial0	Received via serial port **0**
Serial1[1]	Received via serial port **0**
TCP.Client	Received via a TCP connection
CSD[2]	Received via a CSD call (GSM data call)

Notes

This command can be executed any time after a remote update has been initiated (**Rupdate.Init**).

- If no parameter after the comma "," is specified, the firmware-based packets are being received in that channel from where the command was sent.
- After this command has been sent successfully, binary Update commands can be sent (see next sub-section).

4.2.2.3.1. Binary update commands

Command Syntax	<sta><length><cmd_id><answer_id><datalength><data><sto>

Table 4-2 Binary update command syntax

Binary Command			Responses[1]	
Format	**Size**	**Description**	**Syntax**	**Example**[2]

Binary Command			Responses[1]	
<sta>	1 Byte	0xFC	$<txt_cmd> $answer lines $SUCCESS OR ERROR $<end:<txt_id>	$<03> $C06F $SUCCESS $<end:01>
<length>	1-n Byte	MSB bit (0x80) signalizes that another length byte follows.The length itself may range from '0x00 … 0x7F' for each byte (to a maximal length of 4096 – the current size of the internal buffer) Shows the number of bytes after <length> until **<sto> (<sto>** is included)		
<cmd_id>	1-n Byte	MSB bit (0x80) signalizes that another cmd byte follows. Specifies the command being sent. In the sub-section are listed all commands and their values		
<answer_id>	1 Byte	Here can be specified any value– the headline of the corresponding response will contain this value in order to identify the response. (Therefore, different values should be used for different commands)		
<datalength>	1-2 Bytes	Specifies the number of following bytes inside <**data**>, its value depends on the specified <**cmd_id**> (see below) MSB bit (0x80) signalizes that another length byte follows The length itself may range from '0x00 … 0x7F' for each byte (to a maximal length of 4096 – the current		
<data>	0-n Byte	(Specified with <**datalength**>) Contains formatted data which depends on the specified <**cmd_id**> (see below)		
<sto>	1 Byte	0xEC		

1. Although update commands are sent in binary, their answers are text messages, which match the PFAL answer format.
2. A response for a checksum command (its id is 0x01)
   <txt_id> - the textual decimal value of <id> (i.e. <id>=255)
   <txt_cmd> - the textual hexadecimal value of <cmd>

Table 4-3 List of binary commands

Syntax	Exit Data Mode	Select Sector	Write Data to Sector	Read Sector Checksum	Clear Sector
<cmd_id>	0x00	0x01	0x02	0x03	0x04
<datalength>	0x00	0x01	4 + number of bytes to be written.	2	0
<data>	-	Number of sector or 99 to select configuration sector.	-	-	-
<pos>	-	-	2 Bytes (position inside sector 0x00-0xFFFF)	2 Bytes (the length of sectors to be checked 0x00-0xFFFF)	-

Syntax		Exit Data Mode	Select Sector	Write Data to Sector	Read Sector Checksum	Clear Sector
	<data_to_write>	-	-	data for this position	-	-
	<cksum>	-	-	2 Bytes (16 bit checksum of <**data_to_write**>. <**pos**> is not included.)	-	-
		Switches back to normal Command mode (PFAL commands)	Selects one of the sectors (0... number returned from *RUpdate.Init* command). If '**cfg**' is selected, only a half sector can be accessed. the new device configuration can be stored there if desired. This selected sector is used for all further commands data can be written only to the currently selected sector The checksum command works only for the currently selected sector Erasing a sector can be performed only on the currently selected one	Writes data to the specified position inside a currently selected sector	Computes a 16 Bit Cksum of the currently selected sector from the first byte until <**pos**>. This Cksum must match with the expected value (i.e. of the new firmware sector). If this Cksum differs from expected results, data is corrupted, which can result in an unreachable device. in case a wrong cksum was reported, the whole sector must be erased (see next command). **Note:** If just a part of a sector needs to be written, the specified position should be the last byte written. If the maximum value (**0xFFFF**) is specified, trailing 0xFFs inside this sector would be also used for calculation. Do not specify **0xFFFF** for the very last sector containing a configuration – the maximum value for this sector is **0x7FF**.	Erases a currently selected sector. (i.e. if corrupted data was inside)

4.2.2.4. Sys.RUpdate.Finish – Finishes a remote update

Command Syntax	Sys.RUpdate.Finish[,<no_reset>]
Example	$PFAL,Sys.RUpdate.Finish // resets the device after the remote update has completed $PFAL,Sys.RUpdate.Finish,1 // doesn't reset of device after the remote update has completed

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

After a successful firmware transmission this command must be sent to the device. In case of a successful firmware update and no parameter is added to this command, the device will perform a reset and starts up with the updated firmware after approx. 30-40 seconds (depending on firmware size).

Parameter description

[<no_reset>]

Optional setting. Determines whether the device should perform a reset after completing the firmware update remotely.

Value	Meaning
1	The device makes no reset after completing the remote update.

Notes

All data required for the firmware update must be specified and verified using checksum commands before executing this command.

- For **"FW_raw"** type:
- If the finish command fails, note that after resuming the firmware update later, a previously transmitted configuration will always be erased.
- This doesn't matter in case the old firmware configuration is used (option **keep_cnf**), but in case it was transferred (option **clear_cnf**), it MUST be transferred again before finishing the update.
- Else the device will be unreachable (because it will start using default settings).
- In case the firmware update was resumed, note that a previously transmitted configuration will be always erased. This doesn't matter in case the old device configuration is used, but in case it was transferred, it MUST be transferred again before finishing the update procedure. Otherwise the device will be unreachable (because it will start with default settings).
- This command doesn't return any answer when successful executed (just in case of an error).

4.2.3. Sys.WebUpdate

The Firmware Web-Update is an advanced method to perform a remote firmware upgrade over the air. After being activated, the web update itself can be initiated via any available textual user interface using the following commands. For more details, please refer to ***App Note: Firmware Web Update for AVL Devices***. See *1.3. Related documents*.

Note: *Web-Update is currently under test and therefore, it is not activated per default. In order to activate it, send the command "PFAL,CNF.Set,DEVICE.TCP.WEBUPDATE=1"*

4.2.3.1. Sys.WebUpdate.Start – Starts an activated Web-Update

Command Syntax	Sys.WebUpdate.Start,"<>"[,<>]
Example	$PFAL,SYS.WebUpdate.Start,"www.test.com/avl_3.0.0_rc19-Zc0535461.zip",80

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command has the following functions:

- Starts an activated Web Update.
- Web Update is an autonomous feature and becomes active as soon as GPRS is connected.
- Interrupted downloads (i.e. when the connection drops) are resumed later.
- Device performs remote update reset when download is completed.
- You can also use an HTTPS connection for the web update. To identify the web server, additional commands are implemented to save the required certificates on the device.
  - ($PFAL,Sys.Webupdate.Start,"https://drive.google.com/test/avl_3.0.0_rc19-Zc0535461.zip")

Parameter description

<url_file>

It specifies the domain name and the path to the firmware file name residing on that server (the path may also include sub-directory separated with/). For example, the "www.test.de/AvlFW/avl_3.0.0_rc19-Zc0535461.zip" instructs the device to go to the **www.test.de** Web server (www.test.de is a dummy, please specify your web server address), open the folder **AvlFW** and access the file named **avl_3.0.0_rc19-Zc0535461.zip**.

[<port>]

It specifies the optional port number (by default, the port number for a Web server is 80) for HTTP protocol.

Notes

- For more details, please refer to ***App Note: Firmware Web Update for AVL Devices***. See *1.3. Related documents*.
- It is strongly NOT recommended to use another remote update when starting web update.
- It is not recommended to start web update again while a web update is in progress.
- Be sure that the Web-Update is activated via configuration setting before executing this command - if not, the following error may occur:

```
$<SYS.WebUpdate.Start>
$Web Update Test Mode is not activated (restart
device after activation)
$ERROR
$<end>
```

4.2.3.2. Sys.Webupdate.Stop – Stops a started Web-Update

Command Syntax	Sys.Webupdate.Stop
Example	$PFAL,Sys.Webupdate.Stop

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command stops a started Web-Update and current download data will be lost. A new download can be started. For more details, please refer to ***App Note: Firmware Web Update for AVL Devices***. See *1.3. Related documents*.

Parameter description

None.

4.2.3.3. Sys.Webupdate.State – Reads the status of the started Web-Update

Command Syntax	Sys.Webupdate.State
Example	$PFAL,Sys.Webupdate.State

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command returns in percent (%) the current state of a started WebUpdate.

Parameter description

None.

4.2.3.4. Sys.Webupdate.SetCertificate – Set certificate for the webupdate connection

Command Syntax	SYS.Webupdate.SetCertificate
Example	$PFAL,SYS.Webupdate.SetCertificate

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Set certificate used by the TLS library for the webupdate connection.
The certificate must be sent after the command and the transmission is finished by "<CR><LF>".

Parameter description

The used certificate chain for the TCP channel if using a TLS connection.

4.2.3.5. Sys.Webupdate.ClearCertificate - Clear certificate used by TLS library

Command Syntax	SYS.Webupdate.ClearCertificate
Example	$PFAL,SYS.Webupdate.ClearCertificate

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

No certificate is used by the TLS library. The device will not check the identity of the used web server.

Parameter description

None.

4.2.3.6. Sys.Webupdate.ShowCertificate - Show certificate used by TLS library

Command Syntax	SYS.Webupdate.ShowCertificate
Example	$PFAL,SYS.Webupdate.ShowCertificate

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Shows the certificate used by the TLS library on the webupdate connection.

Parameter description

None.

4.2.4. Sys.Device

Device commands allow you to reset the system, to set the system into a sleep mode or to shut down the system. Once one of these actions is performed the corresponding event raises in your application respectively. The event ***Sys.Device.eStart*** raises after the system restarts (after initialization), while the shutdown event raises once the shutdown command is executed. You can then handle these events *(see chapter 6.1.10.)* to execute alarms you need in such cases. See examples in chapters *11.6.1.1.* and *11.6.1.2.* The following actions can be executed:

- Reset,
- Update (firmware update, only locally via serial interface)
- Shutdown, FactoryReset
- Sleep, CfgUpdateMode

4.2.4.1. Sys.Device.Reset – Initiates a device restart with optional delay

Command Syntax	Sys.Device.Reset[,<user_reset_with_timeout>]
Example	$PFAL,Sys.Device.Reset $PFAL,Sys.Device.Reset,1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to perform a user device restart with a time delay.

Notes

In order to prevent the system instability, refresh the system initialization and ensure that the device is running at optimum performance. It is recommended to restart your device at least weekly. This can be done by starting a Timer when the device starts up and restarting it when a Timer expires: e.g.

```
$PFAL,Cnf.Set,AL1=Sys.Device.eStart:Sys.Timer0.Star
t=cyclic,604800000
$PFAL,Cnf.Set,AL2=Sys.Timer.e0:Sys.Device.Reset,10
```

This configuration would reset the device every week.

Parameter description

[<user_reset_with_timeout>]

Optional setting. This optional parameter performs a time delay in seconds before restarting the device and generates a wakeup event after device restarts. The default restart time delay is 5 sec. Following values are possible:

Value	Meaning
0..4	Performs a device restart after 5 seconds with an wakeup event “UserX” on startup The wakeup event **USER0..4** is generated after the system restarts.
1..5	Device will perform a restart after 5 seconds
>5	Device will perform a restart after the specified timeout in seconds

Notes

- This command will not respond any answer because the device either restarts immediately or after the time out expires.
- If the timeout is set to 0, the **eWakeupReason** event with the reason “**USER0**” will be generated/displayed after the device restarts.

4.2.4.2. Sys.Device.Shutdown – Initiates a device shutdown.

Command Syntax	Sys.Device.Shutdown
Example	$PFAL,Sys.Device.Shutdown

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command causes the device hardware to enter sleep-mode immediately.

- The internal software does not check other system states when receiving this command.
- This command is not **DEVICE.IGNTIMEOUT** configuration-dependent.
- No pending SMS, Email and TCP packets are executed, such information would get lost.
- For applications where safety against losing messages/data is required, this command should not be used, use ***4.2.4.5. Sys.Device.Sleep=<wakeup_condition> – Sends the device into sleep mode*** instead.
- FOX3 wakes up from the shutdown mode whenever detecting a level change on Ignition pin.

Parameter description

None.

4.2.4.3. Sys.Device.FactoryReset – Resets device to the factory defaults

Command Syntax	Sys.Device.FactoryReset
Example	$PFAL,Sys.Device.FactoryReset

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command resets the device to its factory-default state. All configuration parameters will be erased - only default settings are available after firmware restarts.

Parameter description

None.

Notes

No responses will be delivered from the FOX3 unit. Once this command is executed, all settings done by the user will be erased. The device will start up and run with factory default configuration, which are listed in chapter *7:*

4.2.4.4. Sys.Device.ChargeSleep=<wakeup_condition> – Send the device into sleep and activate charging

Command Syntax	Sys.Device.ChargeSleep=<wakeup_condition>
Example	$PFAL,Sys.Device.ChargeSleep=Ign

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	×
2	×	×	×	×
3	?	?	?	✓
4	✓	✓	✓	×
5	×	×	×	×

Command description

This command sends the AVL device into sleep mode and activates charging of the internal battery (if available) while the device is sleeping.

Parameter description

<wakeup_condition>

It defines the conditions to wake up the AVL device from the doze mode. At least one <wakeup_condition> must be specified. Use a ‚+'(plus-sign) between the condition names, if several **<**wakeup_condition**>** are needed to be specified. The supported **<**wakeup_condition**>** are listed in the command *4.2.4.5.*

4.2.4.5. Sys.Device.Sleep=<wakeup_condition> – Sends the device into sleep mode

Command Syntax	Sys.Device.Sleep=<wakeup_condition>
Example	$PFAL,Sys.Device.Sleep=Ign $PFAL,Sys.Device.Sleep=Ign+GPS $PFAL,Sys.Device.Sleep=Ign+Ring+Timer=1:20:00 $PFAL,Sys.Device.Sleep=Ign+Motion=200 $PFAL,Sys.Device.Sleep=AiWu=3,10 $PFAL,Sys.Device.Sleep=Ign+LowBat=3.6

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓
2	×	×	×	×
3	?	?	?	✓
4	✓	✓	✓	×
5	×	×	×	×

Command description

This command sends the device into sleep mode which means in detail:

- Event "Sys.eShutdown" is generated
- A sleep timeout is started that has already been configured by the user with the configuration parameter **DEVICE.IGNTIMEOUT**
- No other alarm Events can be generated at this point
- Active alarms will be executed
- Existing connections like TCP and GPRS will be closed when pending messages are sent via TCP.

If all these steps have been performed (or the started sleep timeout is expired), the system enters the sleep mode.

Notes

If an AVL device has an internal battery and you are going to use applications where the device will spend most of the time into the sleep mode, it is recommended to use the PFAL-Command **$PFAL,Sys.Device.ChargeSleep=<wakeup_condition>** instead of **$PFAL,Sys.Device.Sleep=<wakeup_condition>**. The command **Sys.Device.ChargeSleep** enables charging of the internal battery while the device is sleeping.

Parameter description

<wakeup_condition>

It defines the wakeup condition. At least one **<wakeup_condition>** must be specified. Use a ‚+'(plus-sign) between the condition names, if several **<wakeup_condition>** are needed to be specified. The following listed wakeup conditions are supported by Lantronix AVL devices for the following PFAL commands:

```
Sys.Device.ChargeSleep=<wakeup_condition>,
Sys.Device.Sleep=<wakeup_condition>
Sys.Device.Doze=<wakeup_condition
```

Value	Meaning
Ign	System starts when the IGN (IN7) gets a Low to High signal level (rising edge) and event "IO.e8=redge" occurs. ***Note:*** *In this sleep mode, the device offers the lowest power consumption.*
Ring [1]	Like a mobile phone the FOX3 series has its own telephone number. The device is activated when it receives a voice call or SMS message. The device rejects that call and then wakes up. This wake up condition should only be used when the device is powered externally, otherwise no alarms are triggered after the device wakes up and no GSM engine is detected by the firmware if no connection to external power. ***Note:*** *This mode requires most of power while being asleep due to the GSM engine stays always on.*
ExtPwrDetect	System starts when external power is connected to the device (in detail: when external voltage rises above 9V). This mode allows an average power consumption (more than Timer, less than Ring).
ExtPwrDrop	System starts when external power is disconnected from the device (in detail: when external voltage drops below 8V). This mode allows an average power consumption (more than Timer, less than Ring).
GPS	Sets the GPS receiver into the Hibernate state. This state reduces to the lowest possible power consumption without switching of the GPS receiver. This mode allows an average power consumption more than **Timer** and less than **Ring**.

Value	Meaning
Motion=<<*motion_param*>>	System will be started when a change of attitude (i.e. a change of motion - only for devices with motion sensor) is detected. This mode allows an average power consumption (more than **Timer**, less than **Ring**). The Motion wakeup parameter should not be used alone. It is recommended to combine it with the IGN-SLEEP parameter. There are two different motion sensors implemented in Lantronix devices: 1) Digital motion sensor (FOX3/-3G/-4G + BOLERO40 Series). The syntax is: <motion_param> <sensitivity>e.g. 200(tolerance in mG) Example: Motion=200 => the device wakes up, if 200mg tolerance on one axis is exceeded <sensitivity> Minimal tolerance can be 0, but should be 142 (otherwise the device may wake up almost immediately). The tolerance has an internal resolution of 72, so values will be rounded internally to match the resolution. Only For BOLERO-LT2 2) Analogue motion sensor (BOLERO-LT2). The syntax is: motion_parameter <cnt_a><cnt_n><sensitivity> e.g. 5,20,30 <cnt_a> Hexadecimal value from 0x00 to 0xFF (without 0x) for the counter A <cnt_n> Hexadecimal value from 0x00 to 0xFF (without 0x) for the counter A Hexadecimal value (without 0x) The value of the <cnt_a> must be smaller than the value of the <cnt_n> These counters can be used to modify inertia tolerance to changes of attitude. Small counter values can be used to filter out high frequent (quick) changes of attitude. In contrast, high counter values increase the inertia and therefore filter out low frequent (slow) changes of attitude. Mathematical description: The inertia tolerance of detection is inversely proportional to the difference between both counters. This implies that if both counters are using the same value, no motion can be detected (maximum inertia tolerance). Basically the larger the difference between <cnt_a> and <cnt_n>, the higher sensitivity to high and low frequent changes of attitude can be achieved. (The sensitivity level itself is configured within <sensitivity>) Recommendation (hexadecimal value without leading 0x): <cnt_a> 0x5 <cnt_n> 0x20 <sensitivity> (value 0x00 – 0xFFFF) This value specifies sensitivity level of detection. The smaller this level, the higher the sensitivity. (a level of 0 would always detect a motion, a level of 0xFFFF never). Recommendation: <sensitivity> 0x20 ----------END---------- It is not recommended to enter values above 600mg, as the device may hardly wake up then. Maximum tolerance which can be entered is 1000mg. Recommendation when using "Motion" as wakeup parameter: It is STRONGLY recommended to install the device parallel to one of the motion sensor axis (so that when the device is not moving, the reported G values shall be almost zero on 2 axes and should showing a high value (positive or negative) of approximately 1000mg on the 3rd axis. To assure an alternative way for waking up the device, the wakeup parameter "Motion" shall be never used as alone wakeup condition.

Value	Meaning
Timer=<timeout>	Device wakes up after the specified time. The sleep time can be specified with an accuracy of 10 minutes. This mode allows low power consumption (higher than IGN). Do not use the TIMER wakeup parameter alone. It is recommended to combine it with the IGN-SLEEP parameter. The format of the timer timeout is <hhh>:<mm>:<ss> <timeout> <hh> Number from 0 – 180. If value exceeded, it sets to 180. <mm> Number from 0 – 59. If value exceeded, it sets to 59. <ss> Number from 0 – 50. If value exceeded, it sets to 50. Its accuracy is 10 seconds. The specified value will be rounded to 10,20,30,40 or 50.
Wakeup=<time_hh_mm_ss>	System will be started at the pre-specified time. The wake up time is system time dependent. This mode allows low power consumption (higher than IGN). Do not use the WAKEUP parameter alone or in combination with the TIMER parameter. It is recommended to combine it with the IGN-SLEEP parameter. The format of the wakeup timeout is <hhh>:<mm>:<ss> <time_hh_mm_ss> <hh> Number ranging from 0 – 24. <mm> Number ranging from 0 – 59. <ss> Number ranging from 0 – 59.
CAN[2]	Device wakes up when activity on CAN bus is detected. This state allows lowest power consumption than IGN-SLEEP state. This wake up condition is supported only with **$PFAL,Sys.Device.Doze**.
LowBat[3]<min_voltage>	=<>System wakes up when battery voltage drops below the specified threshold. This mode allows an average power consumption (more than Timer, less than Ring). **Warning: Invalid or impossible thresholds can cause the device to wake up immediately or keep sleeping "forever" - therefore it is suggested to:** **- test the values before deploying devices in the field.** **- use additional wakeup conditions so that the device can be woken up when desired.** <min_voltage> It specifies the voltage value, up 3.5V, to wake up the device when the specified voltage is reached.

Value	Meaning
AiWu[4]=<analog thresholds>	System wakes up when voltage on IO0 exceeds the defined upper or lower threshold. Alternatively: if the device provides an additional detection possibility (i.e. GPS antenna disconnects or other extensions, the wakeup condition depends no longer on IO0 – instead the specific additional wakeup condition causes a wakeup. This state allows average power consumption more than **TIMER-SLEEP**, but less than **RING-SLEEP**. <analog_thresholds> <a_min_voltage >,<a_max_voltage > **for IO0** or <d_min_level >,<d_max_level >**if IO0 is not used** <a_min_voltage> Specifies the minimum allowed voltage threshold. <a_max_voltage> It specifies the maximal allowed voltage threshold. True input voltages (e.g. between 0 and 40V) have to be specified as thresholds – (regardless of voltages specified in IO0 configuration/calibration). If incorrect voltages are specified, this command will return in error (no sleep state is entered then). It specifies the lowest voltage (for offset command) or highest voltage (for gain command) to be measured on this IO. For more detailed information (about voltage ranges etc.), refer to chapter *4.4.8.* Examples: -5.1 = -5.001 V 12 = 12.000 V 1.123 = 1.123 V ***Note:*** *If invalid voltage levels are entered, the currently configured offset voltage and gain voltage of IO0 is used.* <d_min_level> Hexadecimal value without 0x (0 – 3FF) <d_max_level> Hexadecimal value without 0x (0 - 3FF) As an alternative of entering voltages, it is possible to specify the detection levels directly. As this way requires to transform voltages into corresponding detection levels, this alternative should be used only for special extensions when level of **IO0** isn't used for **AiWu**.
Serial1[5]	System wakes up when when receiving data on the serial port. This wake up condition is only for Doze mode available.

Notes

- Wakeup conditions are not case sensitive (as usual).
- The order of several submitted wakeup conditions doesn't matter.
- Always use these sleep modes in combination with IGN-Sleep.

4.2.4.6. Sys.Device.Doze=<wakeup_condition> – Sends the device into doze mode

Command Syntax	Sys.Device.Doze=<wakeup_condition>
Example	$PFAL,Sys.Device.Doze=Ign $PFAL,Sys.Device.Doze=Ign+Wakeup=23:20:00 $PFAL,Sys.Device.Doze=Ign+Motion=5,20,20 // for all other AVL devices, except FOX3 & BOLERO40 $PFAL,Sys.Device.Doze=Ign+Motion=200 // FOX3 & BOLERO40 only $PFAL,Sys.Device.Doze=Ign+LowBat=3.6 $PFAL,Sys.Device.Doze=Ign,Serial1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓*
1	✓	✓	✓	✖
2	🗝	🗝	🗝	✖
3	?	?	?	✖
4	✓	✓	✓	✖
5	✓	✓	✓	✖

* Function Not Supported

Command description

This command provides an alternative method to reduce power consumption and sets the AVL device into the doze mode. In this mode the processor goes to sleep, GSM keeps running in stand-by state and GPS enters hibernate state. When the device wakes up from the doze mode, the event *SYS.Device.eStart=Doze will be* occurred.

Parameter description

<wakeup_condition>

It defines the conditions to wake up the AVL device from the doze mode. At least one <wakeup_condition> must be specified. Use a ‚+'(plus-sign) between the condition names, if several **<**wakeup_condition**>** are needed to be specified. The supported **<**wakeup_condition**>** are listed in the *4.2.4.5.*command. The numbers 1, 2, 3 and 4 in table above refer to the **<**wakeup_condition> in chapter *4.2.4.5.*

Notes

- Wakeup conditions are not case sensitive (as usual).
- The order of several submitted wakeup conditions doesn't matter, but always use these sleep modes in combination with **IGN-Sleep**.

4.2.4.7. Sys.Device.CfgUpdateMode – Sets the configuration settings into update mode

Command Syntax	Sys.Device.CfgUpdateMode,[<timeout>]
Example	$PFAL,Sys.Device.CfgUpdateMode,120

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command enters the device into a special configuration update mode, which is recommended for changing alarm configurations. This mode prevents the interferences between the old and new alarms.

When being inside this mode:

- All configured alarms become inactive.
- The configured Macros can be directly executed via PFAL commands.

Parameter description

[<timeout>]

Number of seconds after which the device resets itself in order to exit this mode.

Notes

- Reset the system to exit this update mode.
- This mode can also be used to test the alarm conditions and to simulate step-by-step usual alarm behavior:
  - The events will be displayed,
  - The alarm states can be checked using the MSG.Info.Alarm command,
  - The alarm actions will be simulated using the PFAL commands (because no actions are being executed - i.e. the user must start timers, increment counters or change the trigger's states etc.).
- However, it is not possible to easily change other system states such as the device speed, connection states and so on.

4.2.4.8. Sys.Device.ClearConfig – Erases the current configuration settings

Command Syntax	Sys.Device.ClearConfig
Example	$PFAL,Sys.Device.ClearConfig

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command erases the current configuration settings and reset the device to default settings. All settings made by the user together with settings made by the factory (e.g. all alarms inside the device delivered with the promotion kit) will be erased.

Parameter description

None

Notes

All configuration parameters will be erased – only default settings are available after rebooting the

firmware.

4.2.4.9. Sys.Device.BackupReset – Factory reset and restore backup

Command Syntax	Sys.Device.BackupReset
Example	$PFAL,Sys.Device.BackupReset

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command performs a factory reset that resets the device to factory default settings and restores the backup configuration settings at once. After rebooting the device loads the configuration settings from backup and if there are no user settings available it loads the factory settings.

Parameter description

None.

4.2.4.10. Sys.Device.RestoreBios – Upgrades or downgrades the on-board BIOS

Command Syntax	Sys.Device.RestoreBios
Example	$PFAL,Sys.Device.RestoreBios

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Upgrades or downgrades the on-board BIOS to the software version which is contained within each firmware. This command is usually used to keep the internal BIOS software up to date. EVAL samples (BIOS 3.0) can also be updated, but will lose the stored information like IO calibration data etc.

Notes

After executing this command, it must be assured that the device keeps powered until the BIOS update completes (approx. 5-10 seconds). Otherwise the device may be damaged, and it is no longer functional.

Parameter description

None.

4.2.4.11. Sys.Device.ClearAGPS – Erases an existing AGPS file from device

Command Syntax	Sys.Device.ClearAGPS
Example	$PFAL,Sys.Device.ClearAGPS

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command erases an existing AGPS file from the device. Note that this command always results in success regardless of an existing AGPS file.

Parameter description

None.

4.2.4.12. Sys.Device.PwrManagement.Set,<Options>,<Off_Time>,<On_Time> – Sets the power management options for deep sleep mode - for ROCK device only

Command Syntax	Sys.Device.PwrManagement.Set,<Options>,<full_cycle>,<Ontime>
Example	$PFAL,Sys.Device.PwrManagement.Set,0,6,5

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Command description

This command is used to set the power management options for using the deep sleep functionality on the Rock device.

Parameter description

<Options>

Reserved. Currently should be set to zero "0".

<full_cycle>

Periodical time duration in minutes after which the device wakes up:

- This value must be always greater than the value of <OnTime>.
- <full_cycle_time> works independent from <OnTime> – i.e. the device will wake up each i.e. 6 hours regardless of specified <OnTime>.
- Depending on the minimal <OnTime>, the minimum value is 6.
- To disable power management when the device is always running, set <full_cycle_time> to 0.

[<Ontime>]

Time duration in minutes. Within this timespan the device will stay powered on (a minimal value of 5 minutes is automatically set). The <OnTime> setting may be set to 0 only if the full cycle time is also 0 (→ device is always powered).

Notes

- Settings will be stored into non-volatile memory and are automatically synchronized at each system start.
- Currently, when reconfiguring an activated DeepSleep mode on a ROCK device you have to disconnect for min. 1 minute the power and reconnect it after sending the DeepSleep command with new settings to the ROCK device. The DeepSleep mode with new settings will be executed after removing and reconnecting the power (battery pack) only.

4.2.4.13. Sys.Device.PwrManagement.EnterSleep – Enters the ROCK device into deep sleep mode

Command Syntax	Sys.Device.PwrManagement.EnterSleep
Example	$PFAL,Sys.Device.PwrManagement.EnterSleep

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✗	✗	✗	✗

Command description

This command enters the ROCK device into the deep sleep mode using the power management parameters specified with the command "*$PFAL,Sys.Device.PwrManagement.Set*".

Parameter description

None.

Notes

Deep sleep mode enters immediately after executing the sleep command and the pending SMS or TCP packets will not be sent.

4.2.4.14. SYS.Device.SetSerialID,"SerialID" - Sets the serial id of the device

Command Syntax	SYS.Device.SetSerialID,"<SerialID>"
Example	$PFAL,SYS.Device.SetSerialID,"0080A3CD6D54"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command Description

Specifies a serial number for the device. This may be required and used for authentication purposes for some remote services.

Parameter Description

Parameter	Value	Meaning
"<SerialID>"	The used serial ID for the device. It can consist of any printable characters and can be up to 16 digits in length.	

4.2.5. Sys.Power

4.2.5.1. Sys.Power.Mode - Sets the device into a low power mode

Command Syntax	Sys.Power.Mode=<mode>,<on_time>,<off_time>
Example	$PFAL,Sys.Power.Mode=auto,30,30 $PFAL,Sys.Power.Mode=doze,20,60 $PFAL,Sys.Power.Mode=disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is used to put the device periodically into the power save mode.

Parameter description

<mode>

Specifies the low power mode to be set.

Value	Meaning
disable	Device remains always in full power operation.
auto	Device switches on power save mode periodically every user-specified time if no event occurs within this time.
doze	System enters doze mode whenever the defined on-time expires. This mode means that process and GPS go to sleep and GSM keeps running in standby state.

<on_time>

Defines the amount of time, in seconds, that elapses before the system enters the auto or doze mode.

<off_time>

Defines the amount of time, in seconds, the system will stay sleeping before it wakes up from the auto or doze mode.

4.2.6. Sys.Set/GetTime

4.2.6.1. Sys.SetTime – Sets the system time

Command Syntax	Sys.SetTime,<date>,<time>
Example	$PFAL,Sys.SetTime,03.06.2013,10:00:00 $<SYS.SetTime> $SUCCESS $<end>

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command sets the system date and time.

Parameter description

<date>

Set the current system date in the format "dd.mm.yy" or "yy/mm/dd". However, if the date was synchronized via GPS, the command results in an error.

<time>

Set the current system time in the format "hh:mm:ss". However, if the time was synchronized via GPS, the command results in an error.

4.2.6.2. Sys.GetTime – Gets the system time

Command Syntax	Sys.GetTime
Example	$PFAL,SYS.GetTime $<SYS.GetTime> $time is 03.06.2013,10:03:21 (monday) $SUCCESS $<end>

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command gets the system date and time.

Parameter description

None.

4.2.7. Sys.1-Wire

FOX3/-3G/-4G series provide a 1-Wire interface on the 6-pins accessory port. The commands *SYS.1Wire.Enable, SYS.1Wire.Devices* are used to activate this interface and communicate with externally connected 1wire sensors. In order to report to a server the current temperature values of the external sensors, use the dynamic variable Sys.1Wire[=<sensor_id>]. For more information about the 1-Wire interface, please refer to ***App Note: Getting Started with 1-Wire Devices.***See *Chapter 1: Related documents*.

4.2.7.1. Sys.1Wire.Enable – Enables the 1-wire bus on the device

Command Syntax	Sys.1Wire.Enable
Example	$PFAL,Sys.1Wire.Enable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command enables requests on the 1-Wire bus and thus the 1-Wire devices connected to this bus.

Parameter description

None.

4.2.7.2. Sys.1Wire.Disable – Disables the 1-wire bus on the device

Command Syntax	Sys.1Wire.Disable
Example	$PFAL,Sys.1Wire.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command disables requests on the 1-Wire bus and thus the 1-Wire devices connected to this bus.

Parameter description

None.

4.2.7.3. Sys.1Wire.Devices – Lists the IDs of the 1-Wire devices connected to the bus

Command Syntax	Sys.1Wire.Devices
Example	$<Sys.1Wire.Devices> $devices 21c2272e000000EF,21c2272e000001EE $SUCCESS $<end>

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command lists the IDs of the 1-Wire devices connected to the 1-Wire bus.

Parameter description

None.

4.2.7.4. Sys.1Wire.Temperature – Lists the temperatures of the 1-Wire sensors

Command Syntax	Sys.1Wire.Temperature
Example	$PFAL,Sys.1Wire.Temperature $<Sys.1Wire.Temperature> $device 10c2272e000000E1 value 20.0‘C $device 10c2272e000001E0 value 21.0‘C $SUCCESS $<end>

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command lists the temperature values of the 1-Wire Temperature sensors that are connected to the 1-Wire bus.

Parameter description

None.

4.2.8. Sys.GSM

4.2.8.1. Sys.GSM.<mode> – Powers on/off the GSM engine

Command Syntax	Sys.GSM.<mode>
Example	$PFAL,Sys.GSM.Enable $PFAL,Sys.GSM.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to power ***ON*** (or OFF) the GSM module.

Parameter description

<mode>

Defines whether to enable or disable the GSM module on the AVL device.

Value	Meaning
enable	Powers on the GSM module on the AVL device (default).
disable	Powers off the GSM module on the AVL device. The purpose of this command is to power off the GSM module, when use of GSM/GPRS services is not required for a long period of time. To reset the GSM module (generally not required) perform a full system reset, using Sys.Device.Reset. It is NOT recommended to use this command, when the system is already GPRS attached.

Notes

- This setting will be stored within non-volatile memory and is restored during start up.
- In case the device has a deeply discharged battery, system waits until the battery is recharged above 2.2V in order to safely start the GSM engine. This also happens during regular start up – when GSM is configured to start automatically.
- This command should be used to start GSM for a longer period (if a **Sys.GSM.Disable** was issued before). It may not be used to reset the GSM core. (Resetting the GSM core isn't required. If it is still desired, perform a complete system reset).

4.2.8.2. Sys.GSM.Reset – Initiates a GSM reset

Command Syntax	Sys.GSM.Reset
Example	$PFAL,Sys.GSM.Reset

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command resets the build in GSM engine (implemented for test purposes only).

Parameter description

None.

Notes

It is not recommended to use this command within alarms. This command is not to be used for regular operation.

4.2.9. Sys.GPS

4.2.9.1. Sys.GPS.<mode> – Powers on/off the GPS engine

Command Syntax	Sys.GPS.<mode>
Example	$PFAL,Sys.GPS.Enable $PFAL,Sys.GPS.Disable
Responses	$GPS started $SUCCESS

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is used to power on (or off) the GPS engine. The event **GPS.Nav.eFix=invalid** occurs when this command is executed, and when the GPS gets a valid fix the event **GPS.Nav.eFix=valid** will also occur.

Parameter description

Parameter	Value	Meaning
<mode>	Defines whether to enable or disable the GPS engine on the AVL device.	
	enable	Powers on the GPS engine on the AVL device *(default)*
	disable	Powers off the GPS engine on the AVL device

4.2.9.2. Sys.GPS.Reset – Initiates a GPS reset

Command Syntax	Sys.GPS.Reset
Example	$PFAL,Sys.GPS.Reset

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command resets the GPS engine of the target device.

Parameter description

None.

Notes

- It is not recommended to use this command within alarms.
- This command is not to be used for regular operation.
- This command resets immediately GPS engine without responses from the target device.

4.2.10. Sys.Timer

Timers are used for alarm configuration only. A timer can be used to trigger periodically an event or a single event. Their purpose is to launch periodical or delayed actions.

Warning: ***Only the armed timers raise events when they expire (by default, all times are armed when system starts up).***

4.2.10.1. Sys.Timer<index>.Configure<mode>,<timeout> – Configures a specific timer

Command Syntax	Sys.Timer<index>.Configure=<mode>,<timeout>
Example	$PFAL,Sys.Timer0.Configure=cyclic,5000 $PFAL,Sys.Timer1.Configure=single,2000 $PFAL,Sys.Timer0.Configure=single,20:00:30 $PFAL,Sys.Timer0.Configure=single,30:30 $PFAL,Sys.Timer39.Configure=……

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Configures a timer and stops it *(use the start command to activate it)*. Reconfiguring a timer is possible with this command.

Parameter description

Parameter	Value	Meaning
<index>	Identifies the index of the timer to be stopped. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

Parameter	Value	Meaning
<mode>	It specifies the task of the timer to be executed. It can be set to:	
	single	Once the timer is expired, it is not restarted again.
	cyclic	Every time the timer expires, it is automatically restarted, which allows setting up periodical alarm actions.
<timeout>	It specifies either a 32-bit integer value from 0 to 2147483647 or the time in hours, minutes, seconds to generate a single or cyclic time event. This event can then be used to execute alarms.	
	0 - 2147483647	Specifies the time in milliseconds (32-bit integer value) to create a single or cyclic timer event.
	mm:ss	Specifies the countdown time (in minutes and seconds) to create a single or cyclic timer event. The event occurs when this timer reaches 0.
	hh:mm:ss	Specifies the countdown time (in hours, minutes and seconds) to create a single or cyclic timer event. The event occurs when this timer reaches 0.

Notes

Take caution when using cyclic timers in combination with a very low timeout value. Always keep an eye for the execution time of alarms which are executed upon this timer event (i.e. periodical SMS cannot be sent faster than each 10 seconds. So, timers with very short time intervals will only slow down the system performances)

- The accuracy of system timers is approx. 500 ms. Therefore, a 0 value of <timeout> is valid, however the timer will be called every 500 milliseconds.
- If a **Timer**<index> is currently running, and this command is executed, the running timer will be stopped. To activate/start it use the "**SYS.Timer0.Start**" command.

4.2.10.2. Sys.Timer<index>.Start=<timer_settings> – Starts/restarts a specific timer

Command Syntax	Sys.Timer<index>.Start=[<timer_settings>]
Example	$PFAL,Sys.Timer0.Start $PFAL,Sys.Timer1.Start $PFAL,Sys.Timer0.Start=cyclic,5000 $PFAL,Sys.Timer1.Start=single,2000 $PFAL,Sys.Timer0.Start=single,20:00:30 $PFAL,Sys.Timer1.Start=cyclic,20:00:30

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Starts a configured Timer. The Timer can also be configured with the Start command. Any time this command is used for a configured timer, this Timer will be set to its initial timeout. Resetting or reconfiguring a timer is possible in this way.

Parameter description

Parameter	Value	Meaning
<index>	Identifies the index of the timer to be stopped. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.
[<timer_settings>]	**Optional**. It can be used to start an un-configured timer or to overwrite the existing settings by the new one. It consists of the <mode> and <timeout> parameters which can be set the same as by the **Sys.Timer<index>.Configure**. If this parameter is used, the syntax of this command looks like this: Sys.Timer<index>.Start=<mode>,<timeout>	
<mode>	It specifies the task of the timer to be performed. It can be set to:	
	single	Once the timer is expired, it is not restarted again. The timer event is triggered just one time.
	cyclic	Every time the timer expires, it is automatically restarted allowing to trigger periodically a timer event.
<timeout>	It specifies either a 32-bit integer value from 0 to 2147483647 or the time in hours, minutes, seconds to generate a single or cyclic time event. This event can then be used to execute alarms.	
	0 - 2147483647	Once the timer is expired, it is not restarted again. The timer event is triggered just one time.
	mm:ss	Specifies the countdown time (in minutes and seconds) to create a single or cyclic timer event. The event occurs when this timer reaches 0.
	hh:mm:ss	Specifies the countdown time (in hours, minutes and seconds) to create a single or cyclic timer event. The event occurs when this timer reaches 0.

Notes

- Take caution when using cyclic timers in combination with a very small timeout. Always keep an eye for the execution time of alarms which are executed upon this timer event (i.e. periodical SMS cannot be send faster than each 10 seconds, so specifying fast timers will only slow down system performance in this case).
- The "**$PFAL,Sys.Timer**<index>**.Start**" command without value can be used only by configured timers.
- The accuracy of system timers is approx. 500 ms. Therefore, a 0 value of <timeout> is valid, however the timer will be called every 500 milliseconds.

4.2.10.3. Sys.Timer<index>.Stop – Stops a running timer

Command Syntax	Sys.Timer<index>.Stop
Example	$PFAL,Sys.Timer0.Stop

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Stops a configured and running timer immediately. Once stopped a start command is needed to restart this timer. Note that a stopped timer cannot be resumed anymore.

Parameter description

Parameter	Value	Meaning
<index>	Identifies the index of the timer to be stopped. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

4.2.10.4. Sys.Timer<index>.Pause– Pauses (suspends) a running timer

Command Syntax	Sys.Timer<index>>.Pause
Example	$PFAL,Sys.Timer0.Pause

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Pauses a running timer. While paused, the remaining time until expiration is memorized. To continue the timer, use the resume function. Also, a start command can be used to force a restart of this timer.

Parameter description

Parameter	Value	Meaning
<index	Determines the index of the timer to be paused. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

4.2.10.5. Sys.Timer<index>.Resume– Restarts a paused timer

Command Syntax	Sys.Timer<index>.Resume
Example	$PFAL,Sys.Timer0.Resume

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Continues a paused timer. Note that the timer must be paused first before resuming it.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be resumed. Up to 40 Timers are available. It can be set to a value as below. Please, specify a paused timer.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

4.2.10.6. Sys.Timer<index>.Arm– Arms an initialized and disarmed timer

Command Syntax	Sys.Timer<index>.Arm
Example	$PFAL,Sys.Timer0.Arm

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Arms an initialized, disarmed timer. Only an armed timer will generate events when it expires. Each Timer is armed per default when configuring it (using the configure command). The start command however doesn't change the armed/disarmed state, allowing to restart disarmed timers too. Note that the timer must be disarmed first before arming it.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be armed. Up to 40 Timers are available. It can be set to a value as below. Please, specify the index of a disarmed system timer.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

4.2.10.7. Sys.Timer<index>.Disarm– Disarms an initialized and armed timer

Command Syntax	Sys.Timer<index>.Disarm
Example	$PFAL,Sys.Timer0.Disarm

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Disarms an initialized, armed timer. Disarmed timer will not generate any events when they expire. Each Timer is armed per default when configuring it (using the configure command). Note that the timer must be armed first before disarming it.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

4.2.10.8. Sys.Timer<index>.Erase - Erases the configuration of a timer

Command Syntax	Sys.Timer<index>.Erase
Example	$PFAL,Sys.Timer0.Erase

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Erases the configuration of a timer. If the timer was running, it is stopped immediately. To restart this timer, it must be configured first (you may use the start command plus initialization parameters).

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

<index>

4.2.10.9. Sys.Timer<index>.Save<slot_id>– Saves a timer state to a storage slot

Command Syntax	Sys.Timer<index>.Save<slot_id>
Example	$PFAL,Sys.Timer0.Save0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Saves the current timer state to a storage slot.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.
<slot_id>	Saves the current timer state to a storage slot. The ID of the slot which is used to store the state. Only 5 storage slots (from 0 to 4) are available in the device for all Timer, Counter, Trigger and GPS.Nav.Position.	

Notes

Alias names can be defined for all storage indices by using ALIAS.STORAGE<storage_index>=<alias_name>.

4.2.10.10. Sys.Timer<index>.Load<slot_id> – Loads the saved timer state from a storage slot

Command Syntax	Sys.Timer<index>.Load<slot_id>
Example	$PFAL,Sys.Timer0.Load0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Loads the current timer with the previously saved timer state of a storage slot.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.
<slot_id>	Saves the current timer state to a storage slot. The ID of the slot which is used to store the state. Only 5 storage slots (from 0 to 4) are available in the device for all Timer, Counter, Trigger and GPS.Nav.Position.	

Notes

- Alias names can be defined for all storage indices by using ALIAS.STORAGE<storage_index>=<alias_name>.
- This operation is successful only if a timer state is saved inside the chosen storage slot.

4.2.10.11. Sys.Timer<index>.State – Reads the state of a used timer

Command Syntax	Sys.Timer<index>.State
Example	$PFAL,Sys.Timer0.State

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Returns the current state of the specified timer. All returned states are separated by comma.

```
Example1: state of timer 17: erased, inactive, armed.
Example2:state of timer 18: initialized, active,
running, armed
Example3:state of timer 19: initialized, active,
paused, disarmed
```

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

4.2.11. Sys.Trigger

Triggers are used for alarm configuration only. Their purpose is to act as additional conditions (so actions can be launched depending on the value of a Trigger).

Any Trigger value can be High or Low. The corresponding event/state will be generated.

4.2.11.1. Sys.Trigger<index>=<state_type> – Sets a Trigger to high or low

Command Syntax	Sys.Trigger<index>=<state_type>
Example	$PFAL,Sys.Trigger0=high

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Sets a Trigger to high or low.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.
<state_type>	Determines the state to be set. Following states are available:	
	High	Sets **Trigger<index>** to high level (active). The corresponding SYS.Trigger.e0=high event is generated and its state SYS.Trigger.s0=high is set to true.
	Low	Sets **Trigger<index>** to low level (inactive). The corresponding SYS.Trigger.e0=low event is generated and its state SYS.Trigger.s0=low is set to true.

Notes

Each trigger is initially low. Their purpose is to act as additional conditions to the alarms (AL). So actions can be launched depending on the value of a Trigger.

4.2.11.2. Sys.Trigger<index>– Reads the current trigger state

Command Syntax	Sys.Trigger<index>
Example	$PFAL,Sys.Trigger0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Returns the current state of the specified trigger.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

4.2.11.3. Sys.Trigger<index>.Save<slot-id> – Saves the state of trigger to a storage slot

Command Syntax	Sys.Trigger<index>.Save<slot_id>
Example	$PFAL,Sys.Trigger0.Save0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Saves the current trigger state to a storage slot.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.
<slot_id>	Saves the current timer state to a storage slot. The ID of the slot which is used to store the state. Only 5 storage slots (from 0 to 4) are available in the device for all Timer, Counter, Trigger and GPS.Nav.Position.	

Notes

Alias names can be defined for all storage indices by using ALIAS.STORAGE<storage_index>=<alias_name>.

4.2.11.4. Sys.Trigger<index>.Load<storage_slot>– Loads a saved trigger from a storage slot

Command Syntax	Sys.Trigger<index>.Load<slot_id>
Example	$PFAL,Sys.Trigger0.Load0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Loads the current trigger with the previously saved trigger state of a storage slot.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.
<slot_id>	Loads the current trigger state to a storage slot. Only 5 storage slots (from 0 to 4) are available in the device for all Timer, Counter, Trigger and GPS.Nav.Position.	

Notes

Alias names can be defined for all storage index by using ALIAS.STORAGE<storage_index>=<alias_name>. This operation is successful only if a Trigger state is saved inside the chosen storage slot.

4.2.12. Sys.Counter

Counters are used for alarm configuration only. Their purpose is to count certain events or combinations of various states. Depending on the counter value other actions can be performed then. Sets, changes or reads system counters which are used as alarms states. Once a counter reaches 0 *(while decrementing or if set to 0)*, an event will be launched. If a counter remains at 0, no events will be generated. Beside the event also the current value of a counter can be used inside alarms.

4.2.12.1. Sys.Counter<index>.Set=<value> – Assigns a value to the counter

Command Syntax	Sys.Counter<index>.Set=<value>
Example	$PFAL,Sys.Counter0.Set=55

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Sets the counter to the specified value.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.
<value>	32-bit integer value from 0 to 2147483647. Sets the value of the specified Counter<index>.	

4.2.12.2. Sys.Counter<index>.Increment=<inc_value> – Increments an existing value of the counter

Command Syntax	Sys.Counter<index>.Increment=<inc_value>
Example	$PFAL,Sys.Counter0.Increment=11

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Adds the specified number to the current value of this counter. Once the counter value reaches maximum, further increments have no effect.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.
<inc_value>	32-bit integer value from 0 to 2147483647. It increments (it counts up from the initial set value toward 2^{32} - 1) the value of the specified **Counter**<index> by a given number <inc_value>.	

4.2.12.3. Sys.Counter<index>.Decrement=<dec_value> – Subtracts an existing value of the counter

Command Syntax	Sys.Counter<index>.Decrement=<dec_value>
Example	$PFAL,Sys.Counter0.Decrement=11

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Subtracts the specified number from the current value of this counter. Once the counter value reaches its minimum (0), further decrements have no effect

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.
<dec_value>	32-bit integer value from 0 to 2147483647. Decrements (it counts down from the initial set value toward 0) the value of the specified **Counter**<index> by a given number <dec_value>.	

4.2.12.4. Sys.Counter<index>.Add – Adds a value to a counter

Command Syntax	Sys.Counter<index>.Add=<value>
Example	$PFAL,Sys.Counter0.Add=20

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
1	✓	✓	✓	✓

Command description

This command adds a value to the counter<index>.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.
<value>	32-bit integer value from 0 to 2147483647. Sets the value of the specified Counter<index>.	

4.2.12.5. Sys.Counter<index>.Sub – Subtracts a value from a counter

Command Syntax	Sys.Counter<index>.Sub=<value>
Example	$PFAL,Sys.Counter0.Sub=20

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

This command subtracts a value from the counter<index>.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.
<value>	32-bit integer value from 0 to 2147483647. Sets the value of the specified **Counter**<index>.	

4.2.12.6. Sys.Counter<index>.Save<slot_id>– Saves the state of a counter to a storage slot

Command Syntax	Sys.Counter<index>.Save<slot_id>
Example	$PFAL,Sys.Counter0.Save0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Saves the current counter state to a storage slot. This operation is successful only if a counter state is saved inside the chosen storage slot.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.
<slot_id>	The ID of the slot which is used to store the state. Only 5 storage slots (from 0 to 4) are available in the device for all Timer, Counter, Trigger and GPS.Nav.Position.	

The ID of the slot which is used to store the state. Only 5 storage slots (from 0 to 4) are available in the device for all Timer, Counter, Trigger and GPS.Nav.Position.

Notes

Alias names can be defined for all storage indices by using ALIAS.STORAGE<storage_index>=<alias_name>.

4.2.12.7. Sys.Counter<index>.Load<slot_id>– Loads a saved counter from the storage slot

Command Syntax	Sys.Counter<index>.Load<slot_id>
Example	$PFAL,Sys.Counter0.Load0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Loads the current counter with the previously saved counter state of a storage slot.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.
<slot_id>	The ID of the slot which is used to store the state. Only 5 storage slots (from 0 to 4) are available in the device for all Timer, Counter, Trigger and GPS.Nav.Position.	

Notes

Alias names can be defined for all storage indices by using ALIAS STORAGE <storage_index> =<alias_name>.

4.2.12.8. Sys.Counter<index>.Clear – Sets the value of a counter to 0

Command Syntax	Sys.Counter<<index>.Clear
Example	$PFAL,Sys.Counter0.Clear

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

Sets the specified counter to **0**. This might cause the generation of a Counter Event if this counter wasn't **0** before.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

4.2.13. Sys.nvCounter

Counters are used for alarm configuration only. Their purpose is to count certain events or combinations of various states. Depending on the counter value other actions can be performed then. Non-volatile counters keep their value during system resets, power on/off. Therefore no load/ save operations are required.

In details:

- Non-volatile counters store their current value within non-volatile memory.
- During system start, these counter values are read
  - If no valid counter setting is found (i.e. if a counter was never saved before),
  - this counter is initially set to 0.

- Although there is a slim theoretical chance of a counter not being stored correctly (if power is removed exactly in the moment of writing the values, memory corruption might occur), the used architecture should reduce or ideally prevent such occasions.

- A special architecture reduces impacts of write/erase restrictions of this non-volatile memory. This allows a huge number (>10 million) write cycles instead of several 100000, before the lifetime of the flash itself is exceeded.
- In case a counter

```
Example:PFAL,Sys.nvCounter0.State
PFAL,Sys.nvCounter0.Set=55
```

Sets, changes or reads system counters which are used as alarms states. Once a counter reaches 0 (while decrementing or if set to 0), an event will be launched.

If a counter remains at 0, no events will be generated.

Beside the event also the current value of a counter can be used inside alarms. Up to 20 nvCounters are available.

The Counter Index is directly appended after the group name.

Each Counter is initially 0.

4.2.13.1. Sys.nvCounter<index>.State – Reports the status of a nvCounter

Command Syntax	Sys.nvCounter<index>.State
Example	$PFAL,Sys.nvCounter0.State

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command returns the current state of the specified counter.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	The range of the counter index.

4.2.13.2. Sys.nvCounter<index>.Clear - Clears the status of a nvCounter

Command Syntax	Sys.nvCounter<index>.Clear
Example	$PFAL,Sys.nvCounter0.Clear

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command sets the specified counter to 0. This might cause the generation of a Counter Event if this counter was not 0 before.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	The range of the counter index.

4.2.13.3. Sys.nvCounter<index>.Set=<value> – Assigns a value to the nvCounter

Command Syntax	Sys.nvCounter<index>.Set=<value>
Example	$PFAL,Sys.nvCounter0.Set=33

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command sets the specified counter to the value you define.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	The range of the counter index.
<value>	Specifies the value for the counter. Up to 20 Counters are available.	
	0 to 2147483647	32-bit integer value for the counter value.

4.2.13.4. Sys.nvCounter<index>.Increment=<inc_value> – Increments the existing value of a nvCounter

Command Syntax	Sys.nvCounter<index>.Increment=<inc_value>
Example	$PFAL,Sys.nvCounter0.Increment=11

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command adds the specified number to the current value of this counter. Once the counter value reaches maximum, further increments have no effect.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the counter to be incremented, Up to 20 counters are available	
	0 to 19	The range of the counter index.
<inc_value>	. It increments (it counts up from the initial set value toward 2^{32} - 1) the value of the specified Counter<index> by a given number <inc_value>.	
	0 to 2147483647	32-bit integer value for the counter value.

4.2.13.5. Sys.nvCounter<index>.Decrement=<dec_value> – Subtracts the existing value of a nvCounter

Command Syntax	Sys.nvCounter<index>.Decrement=<dec_value>
Example	$PFAL,Sys.nvCounter0.Decrement=11

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Subtracts the specified number from the current value of this counter. Once the counter value reaches its minimum (0), further decrements have no effect.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the counter to be subtracted. Up to 20 Counters are available.	
	0 to 19	The range of the counter index.
<dec_value>	Decrements (it counts down from the initial set value toward 0) the value of the specified **Counter**<index>> by a given number <dec_value>.	
	0 to 2147483647	32-bit integer value for the counter value.

4.2.14. Sys.Macro

4.2.14.1. Sys.Macro<index>– Activates an already configured macro

Command Syntax	Sys.Macro<index>
Example	$PFAL,Sys.Macro0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

Command description

This command is intended to activate a configured macro. To configure a macro, please refer to

the corresponding specification of the parameter *( )*.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index of the timer to be disarmed. Up to 40 Timers are available. It can be set to.	
	0 to 19	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 39[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

Notes

- This command does not directly execute several times a macro, even if the macro is activated e.g. by an event, it will execute the macro just once.
- Of course, a macro can be executed several times, but the time between 2 activations depends on how much time the macro needs to execute all set commands within it before it will be reactivated again.
- If the time-span between 2 activations is too short, then only a part of set commands within a macro might be executed twice.

4.2.15. Sys.CAN - 1st CAN bus

CAN interface on AVL devices is available as:

`High speed CAN interface` `(CAN interface supporting a baudrate up to 1M, but is not fault tolerant.`

This group contains all necessary commands for using this CAN interface. For more detailed information about the CAN bus refer ***App Note: CAN Applications with AVL Devices***. *See 1.3. Related documents.*

If the CAN Bus option is ordered, the following pins will be used for connecting a AVL device to the in-vehicle CANBus interface:

FOX3 Series uses (with CA31b, CA39b or CA68 installation cable) :		
Color	**PIN**	**Meaning**
Green	6	CAN_H (dominant HIGH)
Yellow	5	CAN_L (dominant LOW)
Brown	2	Ground
Red	1	Input voltage (+10.8.....32.0V DC)

Examples:

Let's assume that a FOX3 device is connected to a high speed CAN bus with 250K baud rate and you need to know the Vehicle/Wheel speed then the connected device only needs to listen out for a message with an identifier of "0x123" and extract the 2nd and 3rd bytes. About the identifiers and the data bytes attached to them contact your vehicle manufacturer.

To do it, follow the steps below:

1. Enable the CANBus interface on the FOX3 device using the command:

```
$PFAL, Sys.Can.Enable,250K, RO
```

2. Add the hex value of identifier (message ID) that contains the data you are interested in (e.g.

Vehicle/Wheel speed has the ID=0x123) using the command below:

```
$PFAL,Sys.Can.Msg.Add,std,123,FFFFFFFF (std: standard
message, FFFFFFFF is a mask that allow only that
message ID to be read)
```

Execute the command "PFAL,sys.can.msg.info" to display the data (highlighted in red) attached to the identifier "0x123" - for example:

```
$<SYS.Can.Msg.Info>
$Msg0/0: type:std id:0x123 mask:0xFFFFFFFF, vars
assigned: 0, data: 01 12 34 56 78 9A BC DE
$SUCCESS
$<end>
```

3. Add a CAN variable (highlighted in red) and the data bytes (highlighted in blue) that are used to represent the Vehicle/Wheel speed. In the example below, the 2nd and 3rd bytes (highlighted in blue) contain the value of the Vehicle/Wheel speed:

```
$PFAL,Sys.Can.Var.Add,0,number,state,std,123,1,0,2,7,
LSB
Variable 0 stores the value of the message std 123
that is added in step 2. The value in variable 0 is
extracted out of the 2nd and 3rd Byte (Byte 1 and 2 ,
because Byte0 would be the first byte). LSB specifies
that the last byte is the most significant one, the
first one contains the low byte.)
```

Execute the command **PFAL,sys.can.msg.info** to show the data (highlighted in red) attached to the identifier "0x123" - for example:

```
$<SYS.Can.Msg.Info>
$Msg0/0: type:std id:0x123 mask:0xFFFFFFFF, vars
assigned: 1, data: 01 12 34 56 78 9A BC DE
$SUCCESS
$<end>
```

4. The speed value stored in the variable 0 is the hex value extracted from Byte1/bit0 to Byte2/bit7 of the identifier "0x123":

```
data: 01 12 34 56 78 9A BC DE
Byte 0 1 2 3 4 5 6 7
```

5. The bytes/bits that contain the speed data are extracted as follows:

```
Byte 1; Bit 0..7: 0x12
Byte 2; Bit 0..7: 0x34
```

The value stored in the CAN variable 0 = 0x3412. (Using MSB instead of LSB at the end of the command above, the value would be read 0x1234 instead of 0x3412)

4.2.15.1. Sys.CAN.Enable – Enables the CAN interface

Command Syntax	Sys.CAN.Enable,<baudrate>,<mode>
Example	$PFAL,Sys.Can.Enable,250K,RO

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

This command activates the **CAN** interface. The setting will be stored in non-volatile memory and **CAN** will be enabled after rebooting the system.

Parameter description

Parameter	Value	Meaning
<baudrate>	Defines the baudrate settings of the CAN Bus. Note that, after changing the baudrate with activated CAN, to activate the new user-specified settings a device reset is required. Following values are available:	
	10K	CAN interface operates at 10 Kbits/s
	20K	CAN interface operates at 20 Kbits/s
	33K3	CAN interface operates at 33.3 Kbits/s
	50K	CAN interface operates at 50 Kbits/s
	83K3	CAN interface operates at 83.3 Kbits/s
	95K2	CAN interface operates at 95.2 Kbits/s
	100K	CAN interface operates at 100 Kbits/s
	125K	CAN interface operates at 125 Kbits/s
	250K	CAN interface operates at 250 Kbits/s
	500K	CAN interface operates at 500 Kbits/s
	666K6	CAN interface operates at 666.6 Kbits/s.
	800K	CAN interface operates at 800 Kbits/s.
	1M	CAN interface operates at 1024 Kbits/s.
<mode>		
	RO	Read Only mode (Silent mode). CAN interface only listens incoming CAN packets. It does not accept any packets or sends any data over the bus. This is the recommended setting, as it does not interfere with other communication at the bus.
	RW	Read Write mode (Running mode). CAN interface accepts received packets and can send CAN messages if requested. Note that the can message must be acknowledged by the CAN bus, otherwise the device keeps repeating this message until an acknowledgement is received. **This setting should be used with caution**, as it influences and interferes with the communication at the connected CAN bus.
	LB	Loop back mode, for self-test function.
	SLB	Loop back combined with silent mode, for self test function.

Notes

This command must be used only if CAN interface has been disabled.

4.2.15.2. Sys.CAN.Disable – Disables the CAN interface

Command Syntax	Sys.CAN.Disable
Example	$PFAL, Sys.CAN.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

This command deactivates the **CAN** interface and all **CAN** dependent commands (except ***Sys.Can.Enable***). The setting will be stored in non-volatile memory and **CAN** will be inactive after the system starts.

Note: *It is not recommended to use this command as alarm action.*

Parameter description

None.

Notes

This command can be used to deactivate the CAN Interface.

4.2.15.3. Sys.CAN.Msg.Add –Adds a CAN message for reading

Command Syntax	Sys.CAN.Msg.Add,<msg_type>,<msg_identifier>[<mask>]
Example	$PFAL,Sys.CAN.Msg.Add,std,12DF $PFAL,Sys.CAN.Msg.Add,std,12DF,FFFFFFF0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

This command adds and saves a **CAN** message into the device. The list of all active **CAN** messages can be retrieved by sending the command ***Sys.CAN.Msg.Info***. Each added **CAN** message will update its data whenever a corresponding **CAN** message is received via the **CAN** Bus. This message data can then be used to retrieve **CAN** variables of it *(i.e. Door open, Engine temperature, speed etc.)*. The message settings will be stored in non-volatile memory and restored after the system starts up. Up to **31** CAN messages (with FW 2.x.x) and up to **28** CAN messages (with FW 3.x.x) can be stored in the device. The saved CAN messages can be reported using the dynamic variable <index>.

Note: *It is not recommended to use this command as alarm action.*

Parameter description

Parameter	Value	Meaning
<msg_type>	Specifies the CAN message type for the 1st CAN interface on the FOX3 devices. It can be set to:	
	std	Standard CAN message (11-bits identifier).
	ext	Extended CAN message (29-bits identifier).
	err	Error message (reserved)
<msg_identifier>	Specifies the **CAN** message identifier, in hexadecimal *(max. 8 digits, usually 3 digits)*, which is used to filter the desired messages out of the **CAN** message stream. Within a <msg_identifier><msg_identifier> more than one variable can be provided. So you have the possibility to save up to **31** CAN messages (with firmware 2.x.x) and up to **28** CAN messages (with firmware 3.x.x) and use up to 50 variable slots for the data (see command **Sys.CAN.Var.Add**,*<slot>,….* for more details). *About the identifiers and the data bytes attached to them contact your vehicle manufacturer.*	
<mask>	***(Recommended for advanced users only !).*** This optional message ID mask can be used to mask out specific identifier bits and allowing to receive messages from a group of messages. It comes in handy if *i.e.* priority bits are part of the message identifier. In this case, the priority bits could be masked out, allowing to receive data from messages with different priorities. The mask itself is a 32-bit value. The High ('1')-bits are used to specify a required message ID bit, while the Low ('0')-bits are used as don't care ID bits – therefore, the more '0' bits within the mask, the more message ID's will match.	

Example: Let's assume that following message identifiers flows on the CAN Bus data steam:

```
- Incoming message identifiers on the CAN interface: A: id=7F  B: id= 30
- Setting with $PFAL,Sys.CANB.Msg.Add,std,70,FFFFFFF0" ("id=70" and
"mask=FFFFFFF0") means:
All messages IDs with matching bits from 0111 0000 (0x70) to 0111 1111
(0x7F) will be received as valid due to the set mask, and all other
messages will be filtered out as invalid. Here below an example.
```

```
Convert the set message id in binary: 0x70=0111 0000
Convert mask (last 2 hex) in binary: 0xF0=1111 xxxx //x= 0 or 1
doesn't care
Search pattern on the CAN interface= 0111 xxxx
Receiving message A with id 0x7F= 0111 1111
```

Which messages will be received as valid ?

```
Valid messages are all IDs with matching bits from:0111 0000 to
0111 1111
Message 0x7F is within the set mask= 0111 1111 //means valid
```

What about the message B ?

```
Convert message B in binary: 0x30 = 0011 0000
Search pattern: = 0111 xxxx //due to the set id "0x70" and mask
"0xF0"
Receiving message with ID 0x30 = 0011 0000
Due to the set id "0x70", the id 0x30 is invalid = 0111 xxxx //
means filtered out
```

Notes

This command must be sent before any CAN variable can be defined for it.

4.2.15.4. Sys.CAN.Msg.Remove – Removes an added CAN message

Command Syntax	Sys.CAN.Msg.Remove,<msg_type>,<msg_identifier>
Example	$PFAL,Sys.CAN.Msg.Remove,std,12DF

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

This command removes a **CAN** message from the system *(and from non-volatile memory)*. Removing a **CAN** message causes all variables of this message to be deleted as well *(as there is no need for them any longer)*.

Note: *It is not recommended to use this command as alarm action.*

Parameter description

Parameter	Value	Meaning
<msg_type>	Specifies the CAN message type for the 1st CAN interface on the FOX3 devices. It can be set to:	
	std	Standard CAN message (11-bits identifier).
	ext	Extended CAN message (29-bits identif
	err	Error message (reserved)
<msg_identifier>	Hexadecimal number *(max. 8 digits, usually 3 digits)* - a CAN message identifier, which is used to filter the desired messages out of the **CAN** message stream.	

Notes

No CAN variables can be created after the corresponding message has been removed.

4.2.15.5. Sys.CAN.Msg.Info – Shows a list of all active CAN & CANB messages

Command Syntax	Sys.CAN.Msg.Info
Example	$PFAL,Sys.CAN.Msg.Info

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

This command shows a list of all active **CAN** & CANB messages.

Parameter description

None.

4.2.15.6. Sys.CAN.Var.Add – Adds a CAN variable to a slot for reading

Command Syntax	Sys.CAN.Var.Add,<slot>,<variable_type>,<notification>,<msg_type><msg_identifier>,<start_byte>,<start_bit>,<stop_byte>,<stop_bit>,<byte_order>[,<src_offset>, <multiplier>, <divider>, <dst_offset>][,<filter>,<f_start_byte>,<f_start_bit>,<f_stop_byte>,<f_stop_bit>]
Example	$PFAL,Sys.Can.Var.Add,0,number,state,std,12DF,0,0,1,7,MSB $PFAL,Sys.Can.Var.Add,0,number,state,std,12DF,0,0,1,7,MSB,0,1,1,0 $PFAL,Sys.Can.Var.Add,0,number,state,std,12DF,0,0,1,7,MSB,0,1,1,0,A,0,0,0,7

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖
1	🗝	🗝	🗝	✖

Command description

This command adds a **CAN** variable to one of 50 **CAN** variable slots available in the 1st CAN and second CANB interface. Each of these slots may contain a **CAN** variable, which is linked to a **CAN** identifier/message. The **CAN** variable retrieves its value (i.e. a number or a string) from its dedicated **CAN** message. At maximum, 4 Bytes of information may be specified to a CAN variable, which represents a 32bit value. A **CAN** variable can never exist or configured without having an active **CAN** message. Whenever a **CAN** message is removed, all dedicated **CAN** variables are removed as well. The variable settings will be stored in non-volatile memory and restored after system start.

Note: *It is not recommended to use this command as an alarm.*

Example of a CAN Variable at slot 0 (see example in table above), which causes no events on change. The variable is an integral number which is extracted from CAN Message 12DF. Its value is stored at position Byte0, Bit0 – Byte1, Bit7 (its length is: 2 Bytes= 16 bit).

Parameter description

Parameter	Value	Meaning
<slot>	Decimal number. Specifies the slot index at which the variable will be stored. This index is also used by dynamic variable &(CAN<slot>) to access and report its value inside. Please note that, the already assigned variable slots in the 2nd CANB cannot be used in the 1st CAN interface	
	0 to 24	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 49[1]	Firmware version 2.16x or 3.1.0_rc12 and higher.
<variable_type>	This variable, which can be either a number or a string, defines the type of data to be stored in the variable slot. It can be set to:	
	Number	In alarm conditions, this variable can be compared to other (integral) numbers, allowing to execute alarms depending on the value of a **CAN** variable.
	String	**(Currently not supported)** In alarm conditions, this variable can be compared with a fixed string (i.e. like operator name) - if the variable content starts with this fixed string, an alarm action can be executed.
<notification>	This variable, which can be either a state or an event, defines how to notify when device detects changes on this slot content. It can be set to:	
	State	No event is occurred. This setting is recommended for all variables which change often, or which might change many times in a short period of time.
	Event	An event is launched whenever the variable changes. It is **strongly NOT recommended** to use it for quickly changing variables, as this will slow down the system drastically.
<msg_type>	Specifies the CAN message type for the 1st CAN interface on the FOX3 devices. It can be set to:	
	std	Standard CAN message (11-bits identifier).
	ext	Extended CAN message (29-bits identifier).
	err	Error message (reserved).
<msg_identifier>	Hexadecimal number *(max. 8 digits, usually 3 digits)*. This identifier specifies the **CAN** message, to which this variable is added. A message must have been configured before (*see Sys.CANB.Msg.Add,<msg_type> <msg_identifier>,[<mask>]* adding a **CAN** variable with this command. *About the identifiers and the data bytes attached to them contact your vehicle manufacturer.*	
<start_byte>	Defines a decimal number from **0 ... 7**. Specifies the data byte of the **CAN** message at which the **CAN** variable value starts.	
<start_bit>	Defines the bit position, a decimal number from **0 ... 7,** within the start/stop byte of the **CAN** message at which the **CAN** variable value starts.	
<stop_byte>	Defines the data byte, a decimal number from **0** ... **7,** of the **CAN** message at which the **CAN** variable value ends.	
<stop_bit>	Defines the bit position, a decimal number from **0** ... **7,** within the start/stop byte of the **CAN** message at which the **CAN** variable value ends.	
<byte_order>	Defines the direction of how to read the 8 byte data added to the specified identifier.	

Parameter	Value	Meaning
	MSB	Most significant byte comes first (i.e. the bytes 0x**AA** 0x**BB** (in this order) is transformed as a variable value **AABB**.
	LSB	Least significant byte comes first (i.e. the bytes 0x**AA** 0x**BB** (in this order) is transformed as a variable value **BBAA**.
Here below are listed some optional settings that can be used to transform the value of a CAN variable into the original unit. For example, the value of a CAN variable is 50000 (millilitre) and you will like to have it in litre, then you must use the formula below for specifying the correct settings in this command: `new_value (in litre) = ( (old_value (in millilitre) + <src_offset>) * <multiplier> / <divider>) + <dst_offset> <dst_offset>` Based on the example described above and that formula, the command settings will look like as follow: `$PFAL,Sys.Can.Var.Add,0,number,state,std,12DF,0,0,1,7,MSB,0,1,1000,0` **`Calculation:`** `new_value (in litre)=((50000ml+0)*1/1000)+0=5 litre`		
<src_offset>	*Optional entry*. Signed decimal number ranging from *-32768* ... *32767*. Specifies a constant value which is added to the value of the CAN variable before applying multiplier and divider. Default value is 0.	
<multiplier>	*Optional entry*. Signed decimal number ranging from *-32768* ...*32767*. Specifies a constant value which the value of the CAN variable is multiplied with. Default value is 1.	
<divider>	*Optional entry*. Signed decimal number ranging from *-32768* ... *32767*. Specifies a constant value which the value of the can variable is divided with. Default value is 1.	
<dst_offset>	*Optional entry*. Signed decimal number ranging from *-32768* ... *32767*. Specifies a constant value which is added to the value of the CAN variable after applying multiplier and divider. Default value is 0.	
<filter>	*Optional entry*. Hexadecimal value which is compared to incoming messages (start/stop positions are defined with following parameters). If the comparison succeeds, the variable value is updated.	
<f_start_byte>	*Optional entry*. Decimal number from 0 to 7 (0-7). Specifies the data byte of the CAN message at which the CAN filter value starts.	
<f_stop_byte>	*Optional entries.* Decimal number from 0 to 7 (0-7). Specifies the data byte of the CAN message at which the CAN filter value ends.	
<f_start_bit>	*Optional entry*. Decimal number from 0 to 7 (0-7). Specifies the bit position within the start byte of the CAN message at which the CAN filter value starts.	
<f_stop_bit>	*Optional entry*. Decimal number from 0 to 7 (0-7). Specifies the bit position within the stop byte of the CAN message at which the CAN filter value ends.	

Notes

- CAN variables can be used in dynamic variables to display values which are retrieved through the CAN Bus.
- CAN variables are updated every second, so an update of filtered values cannot be guaranteed if several messages with the same ID are sent quickly in a row over the bus.

4.2.15.7. Sys.CAN.Var.Remove – Removes an added CAN variable from the slot

Command Syntax	Sys.CAN.Var.Remove,<slot>
Example	$PFAL,Sys.Can.Var.Remove,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖
1	🗝	🗝	🗝	✖

Command description

This command removes the **CAN** variable from the given slot and from non-volatile memory. This variable is no longer present and will cause no more events. Its state cannot be used to trigger alarms anymore.

Parameter description

Parameter	Value	Meaning
<slot>	Decimal number. The slot index, from which the variable will be removed. This index is used to access the variable.	
	0 to 24	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 49[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

4.2.15.8. Sys.CAN.Var.Info – Shows settings and current CAN value of the given slot

Command Syntax	Sys.Can.Var.Info,<slot>
Example	$PFAL,Sys.Can.Var.Info,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖
1	🗝	🗝	🗝	✖

Command description

This command shows settings and current value of **CAN** variable within given slot.

Parameter description

Parameter	Value	Meaning
<slot>	Decimal number. The slot index, from which the variable will be removed. This index is used to access the variable.	
	0 to 24	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 30[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

4.2.15.9. Sys.CAN.GetTimings – Shows the hardware timing of the CAN bus

Command Syntax	Sys.Can.GetTimings
Example	$PFAL,Sys.Can.GetTimings

Command Syntax	Sys.Can.GetTimings
Responses	$<SYS.CAN.GetTimings> $baud=100000 bps $clk=32000000 $BPR=16 $nQ=20 $TSEG1=13 $TSEG2=6 $SJW=4 $SUCCESS $<end>

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

This command shows the timing of **CAN** hardware. This function is only for diagnostic purposes and should be only used by experts. It is not relevant for using FMS functionalities.

Parameter description

None.

4.2.15.10. Sys.CAN.FMS.<mode> – Enables/Disables the CAN-FSM functionalities

Command Syntax	Sys.Can.FMS.<mode>
Example	$PFAL,Sys.Can.FMS.Enable $PFAL,Sys.Can.FMS.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

This command enables or disables the FMS interface in the AVL device. This command deletes all already existing CAN filters and CAN variables when executed. For more detailed information about the FMS parameters (dynamic variable) that are supported by AVL devices and how to connect and request such parameters (dynamic variable) using an AVL device, refer to ***App Note: How to Collect CAN FMS/J1939/OBD-II Data with FOX3 Series.*** *See* *1.3. Related documents*

Parameter description

Parameter	Value	Meaning
<mode>	Defines whether to enable or disable the FMS interface on the AVL device. The CAN-Bus can be disabled while the FMS is enabled.	
	enable	Enables CAN FMS functionality. This command deletes all pre-existing CAN-filters and CAN-variables.
	disable	Disables CAN FMS functionality. This command deletes all CAN-filters and CAN-variables.

4.2.15.11. Sys.CAN.OBDII.Enable – Enables OBD-II port on 1st CAN interface (on main port)

Command Syntax	Sys.CAN.OBDII.Enable[,<format>]
Example	$PFAL,Sys.CAN.OBDII.Enable,std

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

This command enables the OBDII functionality and also the frame format (identifier) on the 1st CAN interface.

Parameter description

Parameter	Value	Meaning
<format>	Defines the identifier to be enabled for reading on 1st CAN interface (on main port).	
	Std	Enables the 11-bit identifier (CAN2.0A).
	Ext	Enables the 29-bit identifier (CAN2.0B).

4.2.15.12. Sys.CAN.OBDII.Disable – Disables the CAN-OBDII functionalities

Command Syntax	Sys.CAN.OBDII.Disable
Example	$PFAL,Sys.CAN.OBDII.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

This command disables the OBDII functionalities in the 1st CAN interface. It deletes all already existing CAN OBDII filters and variables when executed. For more detailed information about the FMS parameters (dynamic variable) that are supported by AVL devices and how to connect and request such parameters (dynamic variable) in the 1st CAN interface, refer to ***App Note: How to Collect CAN FMS/J1939/OBD-II Data with FOX3 Series.*** See *1.3. Related documents*.

Parameter description

None.

4.2.15.13. Sys.CAN.Timeout – Defines CAN timeout for Idle/active events

Command Syntax	Sys.CAN.Timeout,<timeout>
Example	$PFAL,Sys.CAN.Timeout,10

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

This command enables to generate Can.eStat=idle and Can.eStat=active events on the 1st CAN interface.

Parameter description

Parameter	Value	Meaning
<timeout>	Defines in seconds the timeout.	

4.2.15.14. Sys.CAN.OBDII.DTCrq – Requests a diagnostic trouble code message (DTC)

Command Syntax	Sys.CAN.OBDII.DTCrq
Example	$PFAL,Sys.CAN.OBDII.DTCrq

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

This command requests a Diagnostic Trouble Code message (DTC) on the CAN-Bus interface. Note that the CAN interface must be configured with RW option in order to request packets.

Parameter description

None.

4.2.15.15. Sys.Can.DTCO.FMS.<mode> – Enables/Disables tachograph communication via FMS

Command Syntax	Sys.Can.DTCO.FMS.<mode>
Example	$PFAL,Sys.Can.DTCO.FMS.Enable $PFAL,Sys.Can.DTCO.FMS.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

This command is intended to enable/disable tachograph reading via FMS on the 1st CAN port.

Parameter description

Parameter	Value	Meaning
<mode>	Defines whether to enable or disable the DTCO for reading.	
	enable	Enables DTCO for reading on the 1st CAN port.
	disable	Enables DTCO for reading on the 1st CAN port.

4.2.15.16. Sys.Can.DTCO.SendAPDU – Transfer messages(requests) to a tachograph

Command Syntax	Sys.Can.DTCO.SendAPDU,<TA>,"<“codes”>"
Example	$PFAL,Sys.Can.DTCO.SendAPDU,0xee,"0x10 0x7e"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

This command is intended to transfer messages (requests) to a tachograph.

Parameter description

Parameter	Value	Meaning
<TA>	Defines the address where to send the request <codes>.	
<codes>	Defines the bytes (requests) to be sent to the <address> specified above.	

4.2.15.17. SYS.CAN.CANopen.enable – Enable CANopen on the main interface (8-pin connector)

Command Syntax	SYS.CAN.CANopen.enable[,<hex_node_id>]
Example	$pfal,sys.can.CANopen.enable $pfal,sys.can.CANopen.enable,0A

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

Enable CANopen on the main interface (8-pin connector). The Node ID is configurable with <hex-_node_id>- Hex number 0.. 7f

Parameter description

Parameter	Value	Meaning
<hex_node_id>	Hexadecimal from 0 to 7f.	

4.2.15.18. SYS.CAN.CANopen.disable – Disable CANopen on the main interface

Command Syntax	SYS.CAN.CANopen.disable
Example	SYS.CAN.CANopen.disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

Command description

Disable CANopen on the main interface (8-pin connector).

Parameter description

None.

4.2.15.19. SYS.CAN.CANopen.cmd,"<CIA309-3 gateway command>"

Command Syntax	SYS.CAN.CANopen.cmd,"<command string>" The syntax is specified in CiA 309-3. Refer to "AppNote_CANopen_gateway_functions.pdf" for a description of this command and the implemented CANopen gateway commands.
Example	$PFAL,SYS.CAN.CANopen.cmd, "[100] 4 r 0x1008 0 vs"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	🗝

Command description

Execute the specified CANopen gateway command on the main interface.

Refer to "AppNote_CANopen_gateway_functions.pdf" for a description of this command and the implemented CANopen gateway commands.

Parameter description

Parameter	Value	Meaning
<command string>	CANopen gateway command string according to CiA 309-3 (enclosed in double quotation marks)	

4.2.16. Sys.CANB - Second CAN bus

The commands listed within this chapter are available for FOX3 series devices with connected IOBOX-CAN accessory device. The symbol (?) means option (ordering of IOBOX-CAN).

Following CAN interface is available:

`High speed CAN interface` `(CAN interface supporting a baudrate up to 1M, but is not fault tolerant).`

This group contains all necessary commands for using the second CAN interface on the FOX3 device. For more detailed information about the CAN bus, ***App Note: How to use IOBOX-WLAN with a FOX3 Series Device.*** *See 1.3. Related documents.*

Table below shows the position of the CAN_High and CAN-Low on the 16 pin connector of the IOBOX-CAN/WLAN. The pin out of the IOBOX-CAN/WLAN is available on chapter *4.4.*

IOBOX-CAN/WLAN uses (with CA38 installation cable) :		
Color	**PIN**	**Meaning**
Brown	Pin10	CAN_High (dominant HIGH)
Black	Pin12	CAN_Low (dominant LOW)
Orange	Pin16	Ground

4.2.16.1. Sys.CANB.Enable –Enables the second CAN interface on IOBOX-CAN

Command Syntax	Sys.CANB.Enable,<baudrate>,<mode>
Example	$PFAL,Sys.CANB.Enable,100K,RO

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command activates the **CAN** interface on the IOBOX-CAN device. The setting will be stored in the non-volatile memory and **CAN** will be enabled after rebooting the system.

Parameter description

Parameter	Value	Meaning
<baudrate>	Defines the baudrate settings of the CAN Bus. Note that, after changing the baudrate with activated CAN, to activate the new user-specified settings a device reset is required. Following values are available:	
	10K	CAN interface operates at 10 Kbits/s
	20K	CAN interface operates at 20 Kbits/s
	33K3	CAN interface operates at 33.3 Kbits/s
	50K	CAN interface operates at 50 Kbits/s
	83K3	CAN interface operates at 83.3 Kbits/s
	95K2	CAN interface operates at 95.2 Kbits/s
	100K	CAN interface operates at 100 Kbits/s
	125K	CAN interface operates at 125 Kbits/s
	250K	CAN interface operates at 250 Kbits/s
	500K	CAN interface operates at 500 Kbits/s
	666K6	CAN interface operates at 666.6 Kbits/s.
	800K	CAN interface operates at 800 Kbits/s.
	1M	CAN interface operates at 1024 Kbits/s.

Parameter	Value	Meaning
<mode>		
	RO	Read Only mode (Silent mode). CAN interface only listens incoming CAN packets. It does not accept any packets or sends any data over the bus. This is the recommended setting, as it does not interfere with other communication at the bus.
	RW	Read Write mode (Running mode). CAN interface accepts received packets and can send CAN messages if requested. Note that the can message must be acknowledged by the CAN bus, otherwise the device keeps repeating this message until an acknowledgement is received. **This setting should be used with caution**, as it influences and interferes with the communication at the connected CAN bus.
	LB	Loop back mode, for self-test function.
	SLB	Loop back combined with silent mode, for self test function.

Notes

This command must be used only if CANB interface has been disabled.

4.2.16.2. Sys.CANB.Disable – Disables the second CAN interface on IOBOX-CAN

Command Syntax	Sys.CANB.Disable
Example	$PFAL,Sys.CANB.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command deactivates the second **CAN** interface on the IOBOX-CAN and all **CAN**-dependent commands (except ***Sys.CANB.Enable***). The setting will be stored in the non-volatile memory and **CAN** will be inactive after the system starts.

Note: *It is not recommended to use this command as alarm action.*

Parameter description

None.

Notes

This command can be used to stop the second CAN Interface.

4.2.16.3. Sys.CANB.Msg.Add –Adds a CANB message to the IOBOX-CAN

Command Syntax	Sys.CANB.Msg.Add,<msg_type>,<msg_identifier>[,<mask>]
Example	$PFAL,Sys.CANB.Msg.Add,**std**,**12DF** $PFAL,Sys.CANB.Msg.Add,**std**,**12DF**,***FFFFFFF0***

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command adds and saves a **CAN** message from the second CAN interface on the IOBOX-CAN into the device. The list of all active **CAN** messages can be retrieved by sending the command ***Sys.CANB.Msg.Info***. Each added **CAN** message will update its data whenever a corresponding **CAN** message is received via the **CAN** Bus. This message data can then be used to retrieve **CAN** variables of it *(i.e. Door open, Engine temperature, speed etc.)*. The message settings will be stored in non-volatile memory and restored after the system starts up. Up to **14** CAN messages can be stored in the device. The saved CAN messages can be reported using the dynamic variable <index>.

Note: *It is not recommended to use this command as alarm action.*

Parameter description

Parameter	Value	Meaning
<msg_type>	Specifies the CAN message type for the 1st CAN interface on the FOX3 devices. It can be set to:	
	std	Standard CAN message (11-bits identifier).
	ext	Extended CAN message (29-bits identifier)
	err	Error message (reserved)
<msg_identifier>	Specifies the **CAN** message identifier, in hexadecimal *(max. 8 digits, usually 3 digits)*, which is used to filter the desired messages out of the **CAN** message stream. Within a <msg_identifier><msg_identifier> more than one variable can be provided. So you have the possibility to save up to **14** CAN messages and use up to 50 variable slots for the data (see command *Sys.CAN.Var.Add,<slot>,….* for more details). *About the identifiers and the data bytes attached to them contact your vehicle manufacturer.*	
<mask>	***(Recommended for advanced users only!).*** This optional message ID mask can be used to mask out specific identifier bits and allowing to receive messages from a group of messages. It comes in handy if *i.e.* priority bits are part of the message identifier. In this case, the priority bits could be masked out, allowing to receive data from messages with different priorities. The mask itself is a 32-bit value. The High ('1')-bits are used to specify a required message ID bit, while the Low ('0')-bits are used as don't care ID-bits – therefore, the more '0' bits within the mask, the more message ID's will match.	

Example 1:

Let's assume that following message identifiers flows on the CAN Bus data steam:

- Incoming message identifiers on the CAN interface: A: id=7F B: id= 30
- Setting with $PFAL,Sys.CANB.Msg.Add,std,70,FFFFFFF0" ("id=70" and "mask=FFFFFFF0") means:

All messages IDs with matching bits from 0111 0000 (0x70) **to** 0111 1111 (0x7F) will be received as valid due to the set mask, and all other messages will be filtered out as invalid. Here below is an example.

```
Convert the set message id in binary: 0x70=0111 0000
Convert mask (last 2 hex) in binary:0xF0=1111 xxxx //x=0 or 1
doesn't care
Search pattern on the CAN interface= 0111 xxxx
Receiving message A with id 0x7F=0111 1111
```

Which messages will be received as valid?

```
Valid messages are all IDs with matching bits from:0111 0000 to
0111 1111
Message 0x7F is within the set mask= 0111 1111 //means valid
```

What about the message B?

```
Convert message B in binary: 0x30 = 0011 0000
Search pattern: = 0111 xxxx //due to the set id "0x70" and mask
"0xF0"
Receiving message with ID 0x30 = 0011 0000
Due to the set id "0x70", the id 0x30 is invalid= 0111 xxxx //
means filtered out
```

Notes

This command must be sent before any CAN variable can be defined for it.

4.2.16.4. Sys.CANB.Msg.Remove – Removes a CANB message from IOBOX-CAN

Command Syntax	Sys.CANB.Msg.Remove,<msg_type>,<msg_identifier>
Example	$PFAL,Sys.CANB.Msg.Remove,**std,12DF**

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command removes a **CAN** message from the system *(and from non-volatile memory)*. Removing a **CAN** message causes all variables of this message to be deleted as well *(as there is no need for them any longer)*.

Note: It is not recommended to use this command as alarm action.

Parameter description

Parameter	Value	Meaning
<msg_type>	Specifies the CAN message type for the 1st CAN interface on the FOX3 devices. It can be set to:	
	std	Standard CAN message (11-bits identifier).
	ext	Extended CAN message (29-bits identif
	err	Error message (reserved)

Parameter	Value	Meaning
<msg_identifier>	Hexadecimal number *(max. 8 digits, usually 3 digits)* - a CAN message identifier, which is used to filter the desired messages out of the **CAN** message stream	

Notes

No CAN variables can be created after the corresponding message has been removed.

4.2.16.5. Sys.CANB.Var.Add – Adds a CAN variable to the slot

Command Syntax	Sys.CANB.Var.Add,<slot>,<variable_type>,<notification>,<msg_type><msg_identifier>,<start_byte>,<start_bit>,<stop_byte>,<stop_bit>,<byte_order>[,<src_offset>,<multiplier>,<divider>,<dst_offset>][,<filter>,<f_start_byte>,<f_start_bit>,<f_stop_byte>,<f_stop_bit>]
Example	$PFAL,Sys.CANB.Var.Add,0,number,state,std,12DF,0,0,1,7,MSB $PFAL,Sys.CANB.Var.Add,0,number,state,std,12DF,0,0,1,7,MSB,0,1,1,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	✗
1	🗝	🗝	🗝	✗

Command description

This command adds a **CAN** variable to one of 50 **CAN** variable slots available for the 1st CAN and second CANB interface. Each of these slots may contain a **CAN** variable, which is linked to a **CAN** identifier/message. The **CAN** variable retrieves its value (i.e. a number or a string) from its dedicated **CAN** message. At maximum, 4 Bytes of information may be specified to a CAN variable, which represents a 32bit value. A **CAN** variable can never exist or configured without having an active **CAN** message. Whenever a **CAN** message is removed, all dedicated **CAN** variables are removed as well. The variable settings will be stored in non-volatile memory and restored after system start.

Note: It is not recommended to use this command as an alarm.

Example of a CAN Variable at slot 0 (see example in table above), which causes no events on change. The variable is an integral number which is extracted from CAN Message 12DF. Its value is stored at position Byte0, Bit0 – Byte1, Bit7 (its length is: 2 Bytes= 16 bit).

Parameter description

Parameter	Value	Meaning
<slot>	Decimal number. Specifies the slot index at which the variable will be stored. This index is also used by dynamic variable &(CAN<slot>) to access and report its value inside. Please note that, the already assigned variable slots in the 2nd CANB cannot be used in the 1st CAN interface	
	0 to **24**	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to **49**[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

Parameter	Value	Meaning
<variable_type>	This variable, which can be either a number or a string, defines the type of data to be stored in the variable slot. It can be set to:	
	Number	In alarm conditions, this variable can be compared to other (integral) numbers, allowing to execute alarms depending on the value of a **CAN** variable.
	String	(Currently not supported) In alarm conditions, this variable can be compared with a fixed string (i.e. like operator name) - if the variable content starts with this fixed string, an alarm action can be executed.
<notification>	This variable, which can be either a state or an event, defines how to notify when device detects changes on this slot content. It can be set to:	
	State	No event is occurred. This setting is recommended for all variables which change often, or which might change many times in a short period of time.
	Event	An event is launched whenever the variable changes. It is **strongly NOT recommended** to use it for quickly changing variables, as this will slow down the system drastically.
<msg_type>	Specifies the CAN message type for the 1st CAN interface on the FOX3 devices. It can be set to:	
	std	Standard CAN message (11-bits identifier).
	ext	Extended CAN message (29-bits identifier).
	err	Error message (reserved).
<msg_identifier>	Hexadecimal number *(max. 8 digits, usually 3 digits)*. This identifier specifies the **CAN** message, to which this variable is added. A message must have been configured before (*see Sys.CANB.Msg.Add,<msg_type>,<msg_identifier>[<mask>]* adding a **CAN** variable with this command. About the identifiers and the data bytes attached to them contact your vehicle manufacturer.	
<start_byte>	Defines a decimal number from **0** ... **7**. Specifies the data byte of the **CAN** message at which the **CAN** variable value starts.	
<start_bit>	Defines the bit position, a decimal number from **0** ... **7,** within the start/stop byte of the **CAN** message at which the **CAN** variable value starts.	
<stop_byte>	Defines the data byte, a decimal number from **0** ... **7,** of the **CAN** message at which the **CAN** variable value ends.	
<stop_bit>	Defines the bit position, a decimal number from **0** ... **7,** within the start/stop byte of the **CAN** message at which the **CAN** variable value ends.	
<byte_order>	Defines the direction of how to read the 8 byte data added to the specified identifier.	
	MSB	Most significant byte comes first (i.e. the bytes 0x**AA** 0x**BB** (in this order) is transformed as a variable value **AABB**.
	LSB	Least significant byte comes first (i.e. the bytes 0x**AA** 0x**BB** (in this order) is transformed as a variable value **BBAA**.

Parameter	Value	Meaning
Here below are listed some optional settings that can be used to transform the value of a CAN variable into the original unit. For example, the value of a CAN variable is 50000 (millilitre) and you will like to have it in litre, then you must use the formula below for specifying the correct settings in this command: `new_value (in litre) = ( (old_value (in millilitre) + <src_offset>) * <multiplier> / <divider>) + <dst_offset> <dst_offset>` Based on the example described above and that formula, the command settings will look like as follow: `$PFAL,Sys.Can.Var.Add,0,number,state,std,12DF,0,0,1,7,MSB,0,1,1000,0` **`Calculation:`** `new_value (in litre)=((50000ml+0)*1/1000)+0=5 litre`		
<src_offset>	*Optional entry*. Signed decimal number ranging from *-32768* ... *32767*. Specifies a constant value which is added to the value of the CAN variable before applying multiplier and divider. Default value is 0.	
<multiplier>	*Optional entry*. Signed decimal number ranging from *-32768* ... *32767*. Specifies a constant value which the value of the CAN variable is multiplied with. Default value is 1.	
<divider>	*Optional entry*. Signed decimal number ranging from *-32768* ... *32767*. Specifies a constant value which the value of the can variable is divided with. Default value is 1.	
<dst_offset>	*Optional entry*. Signed decimal number ranging from *-32768* ... *32767*. Specifies a constant value which is added to the value of the CAN variable after applying multiplier and divider. Default value is 0.	
<filter>	*Optional entry*. Hexadecimal value which is compared to incoming messages (start/stop positions are defined with following parameters). If the comparison succeeds, the variable value is updated.	
<f_start_byte>	*Optional entry.* Decimal number from 0 to 7 (0-7). Specifies the data byte of the CAN message at which the CAN filter value starts.	
<f_stop_byte>	*Optional entries.* Decimal number from 0 to 7 (0-7). Specifies the data byte of the CAN message at which the CAN filter value ends.	
<f_start_bit>	*Optional entry*. Decimal number from 0 to 7 (0-7). Specifies the bit position within the start byte of the CAN message at which the CAN filter value starts.	
<f_stop_bit>	*Optional entry*. Decimal number from 0 to 7 (0-7). Specifies the bit position within the stop byte of the CAN message at which the CAN filter value ends.	

Notes

- CAN variables can be used in dynamic variables to display values which are retrieved through the second CAN interface.
- CAN variables are updated every second, so an update of filtered values cannot be guaranteed if several messages with the same ID are sent quickly in a row over the bus.

4.2.16.6. Sys.CANB.Var.Remove – Removes a CAN variable from the slot

Command Syntax	Sys.CANB.Var.Remove,<variable_slot>
Example	$PFAL,Sys.CANB.Var.Remove,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×
1	🗝	🗝	🗝	×

Command description

This command removes the CAN variable from the given slot in the second CAN interface and from non-volatile memory. This variable is no longer present and will cause no more events. Its state cannot be used to trigger alarms anymore.

Parameter description

Parameter	Value	Meaning
<variable_slot>	Decimal number. Slot index, from which the variable will be removed. This index is used to access the variable.	
	0 to 24	Firmware version 2.15.x and lower as well as firmware version 3.0.0_rc20.
	0 to 49[1]	Firmware version 2.16.x or 3.1.0_rc12 and higher.

4.2.16.7. Sys.CANB.GetTimings – Shows the hardware timing of CANB

Command Syntax	Sys.CANB.GetTimings
Example	$PFAL,Sys.CANB.GetTimings
Possible Responses	$<SYS.CAN.GetTimings> $baud=100000 bps $clk=32000000 $BPR=16 $nQ=20 $TSEG1=13 $TSEG2=6 $SJW=4 $SUCCESS $<end>

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command shows the timing of **CAN** hardware in the second CAN interface. This function is only for diagnostic purposes and should be only used by experts. It is not relevant for using FMS functionalities.

Parameter description

None

4.2.16.8. Sys.CANB.FMS.<enable> – Enables/Disables the CAN-FSM functionalities

Command Syntax	Sys.CANB.FMS.<enable>
Example	$PFAL,Sys.CANB.FMS.Enable $PFAL,Sys.CANB.FMS.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command enables or disables the FMS interface in the second CAN interface. This command deletes all already existing CAN filters and CAN variables when executed. For more detailed information about the FMS parameters (dynamic variable) that are supported on the second CAN interface and how to connect and request such parameters (dynamic variable) on the second CAN interface, refer to ***App Note: How to Collect CAN FMS/J1939/OBD-II Data with FOX3 Series.*** See *1.3. Related documents*.

Hint: The firmware running on the IOBOX-CAN device supports currently up to 14 CAN message.

Parameter description

Parameter	Value	Meaning
<enable>	Defines whether to enable or disable the FMS interface on the IOBOX-CAN. The second CAN-interface can be disabled while the FMS is enabled.	
	enable	Enables CAN FMS functionality on the second CAN interface. This command deletes all pre-existing CAN-filters and CAN-variables.
	disable	Disables CAN FMS functionality on the second CAN interface. This command deletes all CAN-filters and CAN-variables.

4.2.16.9. Sys.CANB.OBDII.Enable – Enables OBD-II port on second CAN interface (on main port)

Command Syntax	Sys.CANB.OBDII.Enable[,<format>]
Example	$PFAL,Sys.CANB.OBDII.Enable,std

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×
1	?	?	?	×

Command description

This command enables the OBDII functionality and also the base frame format (identifier) on the second CAN interface (IOBOX-CAN).

Parameter description

Parameter	Value	Meaning
<format>	Defines the identifier to be enabled for reading messages on second CAN interface.	
	Std[1]	Enables the 11-bit identifier (CAN2.0A).
	Ext[1]	Enables the 29-bit identifier (CAN2.0B)

4.2.16.10. Sys.CANB.OBDII.Disable – Disables the CAN-OBDII functionalities

Command Syntax	Sys.CANB.OBDII.Disable
Example	$PFAL,Sys.CANB.OBDII.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command disables the OBDII functionalities in the second CAN interface. It deletes all already existing CAN OBDII filters and variables when executed. For more detailed information about the FMS parameters (dynamic variable) that are supported by AVL devices and how to connect and request such parameters (dynamic variable) in the second CAN interface, refer to ***App Note: How to Collect CAN FMS/J1939/OBD-II Data with FOX3 Series.*** See *1.3. Related documents*.

Parameter description

None

4.2.16.11. Sys.CANB.Timeout – Defines CANB timeout for Idle/active events

Command Syntax	Sys.CANB.Timeout,<timeout>
Example	$PFAL,Sys.CANB.Timeout,10

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command enables to generate *Can.eStat=idle and Can.eStat=active* events on the second CAN interface (IOBOX-CAN).

Parameter description

Parameter	Value	Meaning
<timeout>	Defines the timeout in seconds.	

4.2.16.12. Sys.CANB.OBDII.DTCrq – Requests a Diagnostic Trouble Code message (DTC)

Command Syntax	Sys.CANB.OBDII.DTCrq
Example	$PFAL,Sys.CANB.OBDII.DTCrq

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command requests a Diagnostic Trouble Code message (DTC) on the CAN-Bus interface. Note that the CAN interface must be configured with RW option in order to request packets.

Parameter description

None

4.2.16.13. Sys.CanB.DTCO.FMS.<mode> – Enables/Disables tachograph communication via FMS

Command Syntax	Sys.CanB.DTCO.FMS.<mode>
Example	$PFAL,Sys.CanB.DTCO.FMS.Enable $PFAL,Sys.CanB.DTCO.FMS.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command is intended to enable/disable tachograph reading via FMS on the second CAN port (IOBOX-CAN).

Parameter description

Parameter	Value	Meaning
<mode>	Defines whether to enable or disable the DTCO for reading.	
	enable	Enables DTCO for reading on the second CAN port
	disable	Enables DTCO for reading on the second CAN port.

4.2.16.14. SYS.CANB.CANopen.enable – Enable CANopen on the second interface (8-pin connector)

Command Syntax	SYS.CANB.CANopen.enable[<hex_node_id>]
Example	$pfal,sys.canb.CANopen.enable $pfal,sys.canb.CANopen.enable,0A

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

Enable CANopen on the second interface (8-pin connector). The Node ID is configurable with <hex_node_id>- Hex number 0 .. 7f

Parameter description

Parameter	Value	Meaning
<hex_node_id>	Hexadecimal from 0 to 7f.	

4.2.16.15. SYS.CANB.CANopen.disable - Disable CANopen on the second interface

Command Syntax	SYS.CANB.CANopen.disable
Example	SYS.CANB.CANopen.disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

Disable CANopen on the second interface (8-pin connector).

Parameter description

None

4.2.16.16. SYS.CANB.CANopen.cmd ,"<CIA309-3 gateway command>"

Command Syntax	SYS.CAN.CANopen.cmd,"<command string>" The syntax is specified in CiA 309-3. Refer to ***App Note: CANOpen Gateway Functions for AVL Devices*** for a description of this command and the implemented CANopen gateway commands. See *1.3. Related documents*.
Example	$PFAL,SYS.CANB.CANopen.cmd, "[100] 4 r 0x1008 0 vs"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

Execute the specified CANopen gateway command on the second interface.

Refer to ***App Note: CANOpen Gateway Functions for AVL Devices*** for a description of this command and the implemented CANopen gateway commands. See *1.3. Related documents*.

Parameter description

Parameter	Value	Meaning
<command string>	CANopen gateway command string according to CiA 309-3 (enclosed in double quotation marks)	

4.2.17. Sys.WLAN

The commands listed within this chapter are available for FOX3/-3G/-4G Series with connected IOBOX-WLAN (Wireless Local Area Network) accessory device. The symbol (?) means option (you need to order the IOBOX-WLAN).

The IOBOX-WLAN is made for anyone who wants to promptly add WLAN connectivity to a FOX3/-3G/-4G Series as fast as possible. It is a WLAN embedded accessory device with a 2.4GHz ISM band wireless radio for enabling WLAN connectivity into FOX3/-3G/-4G Series. The IOBOX-WLAN is suitable for wireless network systems based on IEEE 802.11 b/g/n 2.4GHz and features the capability to operate as WLAN client and connects to routers and other Aps (Access Point) allowing connectivity to Internet instead of using mobile networks. All that is needed is a FOX3/-3G/-4G Series and an IOBOX-WLAN. It plugs into the mini-USB port on one of the FOX3/-3G/-4G Series. Eight IOs are available for digital outputs or digital inputs for almost every application within the automobile industry. To know more about and control these IOs, please refer to the chapter *4.4.*

This command group contains all necessary commands for using the IOBOX-WLAN on the FOX3/-3G/-4G Series device.

4.2.17.1. Sys.WLAN.<mode> – Enables or disables the WLAN module on IOBOX-WLAN

Command Syntax	Sys.WLAN<mode>
Example	$PFAL,Sys.WLAN.Enable $PFAL,Sys.WLAN.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command enables or disables the WLAN module on the IOBOX-WLAN. It should be already connected to the FOX3/-3G-4G Series when executing one of these commands. To start scanning for WLAN networks, refer to the command group in chapter *4.9.*

Parameter description

Parameter	Value	Meaning
<mode>	Defines whether to enable or disable the WLAN module. It can be set to:	
	enable	Enables and boots the WLAN module.
	disable	Disables the WLAN module

4.2.18. Sys.LUA

The commands listed within this chapter are available for FOX3/-3G/-4G Series with activated LUA as premium-Feature. For more details about commands events and state provided for Lua scripts, refer to ***App Note: Using Lua Scripts for FOX3 and BOLERO40 Series.*** *See 1.3. Related documents*.

4.2.18.1. Sys.Lua.Start – Starts a Lua script loaded to device

Command Syntax	Sys.Lua.Start
Example	$PFAL,Sys.Lua.Start

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command starts and runs a Lua script that is available in the device. If the Lua script is already running this command responds an error.

Parameter description

None

4.2.18.2. Sys.Lua.Stop – Stops an already running Lua script

Command Syntax	Sys.Lua.Stop
Example	$PFAL,Sys.Lua.Stop

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command stops an already running Lua script. If the Lua script is already stopped this command responds an error.

Parameter description

None.

4.2.18.3. Sys.Lua.Dump – Reads Lua script source code available on device

Command Syntax	Sys.Lua.Dump
Example	$PFAL,Sys.Lua.Dump

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	?	🗝	🗝

Command description

This command reads the source code of that Lua script available on the device.

Parameter description

None

4.2.18.4. Sys.Lua.Lock,<"password"> – Protects Lua script from reading

Command Syntax	SYS.Lua.Lock,<"password">
Example	$PFAL,SYS.Lua.Lock,"112233"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command protects the source code of that Lua script from reading using a password.

Parameter description

Parameter	Value	Meaning
<password>	Defines the password to protect the Lua script from reading.	

4.2.18.5. Sys.Lua.UnLock,<"password"> – Unlocks a password-protected Lua script

Command Syntax	SYS.Lua.UnLock,<"password">
Example	$PFAL,SYS.Lua.UnLock,"112233"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command unlocks a password-protected Lua script for reading.

Parameter description

<"password">

Defines the password to protect the Lua script from reading.

4.2.18.6. Sys.Lua.Dump,<"password"> – Reads a password-protected Lua script

Command Syntax	SYS.Lua.Dump,<"password">
Example	$PFAL,SYS.Lua.Dump,"112233"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command reads a password-protected Lua script.

Parameter description

Parameter	Value	Meaning
<password>	Defines the password to protect the Lua script from reading.	

4.2.18.7. Sys.Lua.Clear – Clears Lua script available on device

Command Syntax	SYS.Lua.Clear
Example	$PFAL,SYS.Lua.Clear

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command clears the Lua script that is available on the device.

Parameter description

None

4.2.18.8. SYS.LUA.Event,<id>,<"text"> – Generates custom events for Lua

Command Syntax	SYS.LUA.Event,<id>,"<"text">
Example	$PFAL,SYS.LUA.Event,23,"do something"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command generates a Lua event when it is executed.. For more details about commands events and state provided for Lua scripts, see LUA Commands, Event and States for LUA Scripts.

Parameter description

Parameter	Value	Meaning
<id>	Defines an id to be called on the Lua script.	
<"text">	Defines the text.	

4.2.18.9. SYS.Lua.Start[,<"script.lua">] – Load specific Lua script

Command Syntax	Sys.Lua.Start[,<"script.lua">]
Example	$PFAL,SYS.Lua.Start,"example.lua"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command starts and runs a specific Lua script that is available in the device. If the Lua script is already running, this command responds an error.

Parameter description

Parameter	Value	Meaning
<"script.lua">	The file name of the Lua script.	

4.2.18.10. SYS.Lua.Clear[,<"script.lua">] – Delete specific Lua script

Command Syntax	Sys.Lua.Clear[,<"script.lua">]
Example	$PFAL,SYS.Lua.Clear,"example.lua"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command deletes a specific Lua script that is available in the device.

Parameter description

Parameter	Value	Meaning
<"script.lua">	The file name of the Lua script.	

4.2.18.11. SYS.Lua.Info[,<"script.lua">] – Comment of specific Lua script

Command Syntax	Sys.Lua.Info[,<"script.lua">]
Example	$PFAL,SYS.Lua.Info,"example.lua"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command returns a comment of a specific Lua script that is available in the device.

Parameter description

Parameter	Value	Meaning
<"script.lua">	The file name of the Lua script.	

4.2.18.12. SYS.Lua.Write[,<"script.lua">] – Write specific Lua script

Command Syntax	Sys.Lua.Write[,<"script.lua">]
Example	$PFAL,SYS.Lua.Write,"example.lua"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command writes a specific Lua script that is available in the device.
The data must send after the command. The transmission is finished by <CR><LF>.

Parameter description

Parameter	Value	Meaning
<"script.lua">	The file name of the Lua script.	

4.2.19. Sys.BLE

Note: *The commands listed within this section are available for FOX3-3G-BLE device only.*

Bluetooth® Low Energy (BLE) (also called Bluetooth® Smart or Version 4.0+ of the Bluetooth® specification) is the power- and application-friendly version of Bluetooth® that was built for the Internet-of-Things (IoT). FOX3-3G-BLE uses BLE 4.1 and it is designed to detect ibeacons within its field and collects their data or to connect to BLE-peripheries such as BLE smartphones and tablets for data exchange.

Note:

- *The FOX3-3G-BLE does not run with the firmware version 2.xx or lower.*
- *The BLE on the FOX3-3G-BLE supports serial data communication (SPP-like protocol) only.*
- *No audio support over Bluetooth®*
- *The maximum length for the FOX3-3G-BLE advertised name is 10 characters.*

Restrictions:

- *FOX3-3G-BLE with the firmware version 3.x.x or higher.*
- *Smartphone or tablet with Android 4.4 or higher to install the demo application which Lantronix provides for test purposes.*
- *BLE ibeacons (if needed)*

This command group contains all necessary commands for using FOX3-3G-BLE device.

For more details, refer to ***App Note: How to Use BLE with the FOX3-3G-BLE Device***. See *1.3. Related documents*.

4.2.19.1. Sys.BLE.Enable – Enables BLE in device

Command Syntax	Sys.BLE.Enable
Example	$PFAL,Sys.BLE.Enable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to enable BLE in FOX3-3G-BLE.

Parameter description

None

4.2.19.2. Sys.BLE.Disable – Disables BLE in device

Command Syntax	Sys.BLE.Disable
Example	Sys.BLE.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to disable BLE in FOX3-3G-BLE.

Parameter description

None

4.2.19.3. Sys.BLE.Scan – Scans for new BLE devices

Command Syntax	Sys.BLE.Scan
Example	$PFAL,Sys.BLE.Scan

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to start scanning for new BLE devices in your near environment. After the scan is finished, the BLE module will store a list of BLE devices. To show the list of available BLE devices use the command *$PFAL,Sys.BLE.List.*

Parameter description

None

4.2.19.4. Sys.BLE.List – Lists available BLE devices

Command Syntax	Sys.BLE.List
Example	$PFAL,Sys.BLE.List

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to show the available BLE devices listed by names.

Parameter description

None

4.2.19.5. Sys.BLE.ClearList – Clears the list of BLE sensors saved from last scan

Command Syntax	Sys.BLE.ClearList
Example	$PFAL,Sys.BLE.ClearList

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to clear the list of BLE beacon already saved into the device from the last BLE scan.

Parameter description

None

4.2.19.6. Sys.BLE.ListDev – Lists of active BLE devices

Command Syntax	Sys.BLE.ListDev=<<content>
Example	$PFAL,Sys.BLE.ListDev=MAC

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to show the list of BLE devices with defined user <content>.

Parameter description

Parameter	Value	Meaning
<content>	Defines the content of the list to be shown. It can be set to:	
	Name	Show the list with names.
	MAC	Show the list with MACs.
	UUID	Show the list with UUIDs.
	RSSI	Show the list with RSSIs

4.2.19.7. Sys.BLE.ListAdd – Lists of added BLE devices

Command Syntax	Sys.BLE.ListAdd=<content>
Example	$PFAL,Sys.BLE.ListAdd=MAC

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to show the list of added BLE devices with defined user <content>..

Parameter description

Parameter	Value	Meaning
<content>	Defines the content of the list to be shown. It can be set to:	
	Name	Show the list with names.
	MAC	Show the list with MACs.
	UUID	Show the list with UUIDs.
	RSSI	Show the list with RSSIs

4.2.19.8. Sys.BLE.ListRel – Lists of released BLE devices

Command Syntax	Sys.BLE.ListRel=<content>>
Example	$PFAL,Sys.BLE.ListRel=MAC

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Command description

This command is used to show the list of lost BLE devices with defined user <content>.

Parameter description

Parameter	Value	Meaning
<content>	Defines the content of the list to be shown. It can be set to:	
	Name	Show the list with names.
	MAC	Show the list with MACs.
	UUID	Show the list with UUIDs.
	RSSI	Show the list with RSSIs

4.2.19.9. Sys.BLE.Select,<index> – Selects a BLE beacon by index in the list

Command Syntax	Sys.BLE.Select,<index>
Example	$PFAL,Sys.BLE.Select,1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to select the user specified index from the list saved during the last device scan. After selecting an index, you can use the supported dynamic variables to report to your server the Name, RSSI, MAC, UUID, Minor, Major values of that ibeacon.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index in the list. Depending on the length of the advertised friendly name of the BLE Beacons, the device can store up to 28 BLE beacons, if name has a length of 24 bytes.	

4.2.19.10. Sys.BLE.Show,<index>,<"text"> – Shows one of the entries in the list

Command Syntax	Sys.BLE.Show,<index>,<”text”>
Example	$PFAL,Sys.BLE.Show,<index>,<"text">

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to select and show one of the list entries.

Parameter description

Parameter	Value	Meaning
<index>	Determines the index in the list. Depending on the length of the advertised friendly name of the BLE Beacons, the device can store up to 28 BLE beacons, if name has a length of 24 bytes.	
<”text”>	Specifies the text of the specified index to be shown. The <”text”> can include values transmitted from beacon.	

4.2.20. Sys.BlueID

Note: *The commands listed within this chapter are available for FOX3-3G-BLE-BlueID device only.*

BlueID is a technology that makes possible to receive and execute sensitive authorizations in a secure manner via smartphone apps. The FOX3-3G-BLE-BID uses BlueID technology to create a system allowing car sharing and rental car companies to identify, access, open and close vehicles with a smartphone or NFC in a simple and smart way using the highest level of secure offline communication. Please contact your regional Lantronix Sales representative for more details about using BlueID.

4.2.20.1. Sys.BlueID.SetTicket – Sets booking for car sharing

Command Syntax	SYS.BlueID.SetTicket,<"<"NFCCardUID">">,<start_date>,<start_time>,<end_date >,<end _time>
Example	$PFAL,SYS.BlueID.SetTicket,"014874198FAC0F23",06.02.2018,07:30,06.02.2018,10:00

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to create a single internal test ticket for one NFC card in FOX3-3G-BLE-BlueID.

Parameter description

Parameter	Meaning
<"NFCCardUID">	Specifies the UUID of the NFC card.
<start_date>	Specifies the start date of the registration for the ticket booking in the format dd.mm.yy (day.month.year)
<end_date >	Specifies the stop date of the registration for the ticket booking in the format dd.mm.yy (day.month.year)
<end _time>	Specifies the stop time of the registration for the ticket booking in the format hh:mm (hour:minute)

4.2.20.2. Sys.BlueID.ListTickets – Lists available tickets

Command Syntax	SYS.BlueID.ListTicket
Example	$PFAL,SYS.BlueID.ListTicket

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to report all tickets stored in internal ticket store.

Output format BLE ticket:

```
<"TicketID"> <"MobileDeviceID">
valid from <start_date>,<start_ time> and <end_date >,<end _time>
```

Parameter description

None.

4.2.20.3. SYS.BlueID.ClearTicket – Clear all available tickets

Command Syntax	SYS.BlueID.ClearTickets
Example	$PFAL,SYS.BlueID.ClearTickets

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to remove all tickets from internal ticket store.

Parameter description

None

4.2.20.4. SYS.BlueID.ClearLastTicket – Clear last available ticket

Command Syntax	SYS.BlueID.ClearLastTicket
Example	$PFAL,SYS.BlueID.ClearLastTicket

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to clear the last ticket ID and last mobile device ID from device internal RAM.

Parameter description

None

4.2.20.5. SYS.BlueID.SetState – Clear last available ticket

Command Syntax	SYS.BlueID.SetState,<<state>
Example	$PFAL, SYS.BlueID.SetState

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only by Fox3-3G BLE.

Command description

This command is used to set the return state which informs the calling mobile app about the execution result of a command. Without setting this the answered standard value is 7 which originally means “timeout after not successful command execution at a maximum time range of 5 seconds”.

Inside this time range you can overwrite the return value to another value [0..6,8.. ?].

Parameter description

Parameter	Meaning
<state>	Sets the return state to inform the calling mobile app about the execution result of a command.

4.2.21. Sys.NFC

The commands listed within this chapter applies for FOX3 device connected to Lantronix NFC (Near field communication) readers only.

4.2.21.1. Sys.NFC.Enable – Enables NFC reader

Command Syntax	Sys.NFC.Enable=<port>
Example	$PFAL,Sys.NFC.Enable=Serial0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	?
1	?	?	?	×

Command description

This command is used to enable communication with a Lantronix NFC reader at a serial port.

Parameter description

Parameter	Value	Meaning
<port>	Determines the port where the NFC reader is connected to.	
	Serial0	The serial port on the 8pin connector.
	Serial1[1]	The serial port on the 6pin connector (Not available on LITE models)

4.2.21.2. Sys.NFC.Disable – Disables NFC reader

Command Syntax	Sys.NFC.Disable
Example	$PFAL,Sys.NFC.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is used to disable communication with Lantronix NFC readers.

Parameter description

None.

4.2.21.3. Sys.NFC.reset – Resets NFC reader

Command Syntax	Sys.NFC.Reset
Example	$PFAL,Sys.NFC.Reset

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is used to reset the Lantronix NFC readers.

Parameter description

None.

4.2.21.4. Sys.NFC.Play – Plays specific tones on buzzer on NFC reader,

Command Syntax	Sys.NFC.play,<tone>$_1$>,<tone>$_2$>…<tone>$_n$>
Example	$PFAL,SYS.NFC.Play,"cdefgahc4" $PFAL,SYS.NFC.Play,"c4 d4 e4 f4 g4 a4 h4 c5" $PFAL,SYS.NFC.Play,"c4,d4,e4,f4,g4,a4,h4,c5" $PFAL,SYS.NFC.Play,"c4/8 d4/8 e4/8 f4/8 g4/8 a4/8 h4/8 c5/8" $PFAL,SYS.NFC.Play,"c4/8ss d4/8ss e4/8ss f4/8ss g4/8ss a4/8ss h4/8ss c5/8ss" $PFAL,SYS.NFC.Play,"c5d5e5f5g5a5h5c6" $PFAL,SYS.NFC.Play,"a4/8l f#5/l a4/8l d5/l a4/8l f#5/l a4/8l d5/l" $PFAL,SYS.NFC.Play,"a4/16l d5/4.l a4/16l d5/16l a4/16l d5/16l a4/16l d5/4.l"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is used to turn on and play specific tones on the buzzer on the Lantronix NFC reader.

Parameter description

Parameter	Value	Meaning
<tone>	<note>|<pause>[<shift>][<oktave>][/[<duration>[.][<art>] The following tones can be played.	
	<note>	It can be set to: a|b|h|c|d|e|f|g
	<pause>	It can be set to: _
	<oktave>	It can be set to: #|-
	<duration>	It can be set to: 0|1|2|3|4|5|6|7|8
	<art>	It can be set to: 1|2|4|8|16
	#	Increase

Parameter	Value	Meaning
	-	Decrease (b)
	.	Dotted duration
	l	Legato: tone duration until the next tone without pause
	s	Staccato: shortened tone duration
	ss	Staccatissiomo: strongly shortened tone duration

4.2.21.5. Sys.NFC.LED – Sets LED behaviours on NFC reader

Command Syntax	Sys.NFC.LED<index>,<config_type>
Example	$PFAL,Sys.NFC.LED

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is used to set or change the LED behaviors on the Lantronix NFC reader.

Parameter description

Parameter	Value	Meaning
<index>	It specifies the index of LED indicator to be configured.	
<config_type>	It specifies the configuration type of the LED. It can be set to:	
	high	Sets LED<index> to the High level.
	low	Sets LED<index> to the Low level.
	hpulse, <high_time>	Performs a single high pulse for the specified time. The LED<index> immediately turns on, waits <high_time> ms and then turns off if it has been off.
	lpulse, <high_time>	Performs a single low pulse for the specified time. The LED<index> immediately turns off, waits <high_time> ms and then on if it has been on.

Parameter	Value	Meaning
	cyclic<high_time><low_time>[<impulse_cnt>]	This setting changes periodically the level of the LED<index> for the specified times. The LED<index> immediately turns on, waits <high_time> ms, then turns off for <low_time> ms. This procedure is repeated until another functionality is send to the device. <high_time> Time in ms at which the port is set to high level. Min. time currently 125 ms. smaller values will be accepted but won't show a different behavior. <low_time> Time in ms at which the port is set to low level. Min. time currently 125 ms. smaller values will be accepted but won't show a different behavior. [<impulse_cnt>] The default value is 0, which results in infinite operation. The value may range between 0 and 255. This is an optional setting for the option "cyclic" setting. It specifies the number of cycles (one cycle is considered as ON pulse and following the OFF pulse) after all pulses have been set, the cyclic functionality stops, and it changes to “OFF" function.

4.2.22. Sys.UserEvent

4.2.22.1. Sys.UserEvent<index> – Creates an user-event

Command Syntax	Sys.UserEvent<index>
Example	$PFAL,Sys.UserEvent0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to create a user-event that can be used to execute different alarms.

Parameter description

Parameter	Meaning
<index>	Determines the index of the *UserEvent* to be created. Up to 10 UserEvents are available. It can be set to a value from **0** to **9**.

Note: *This event can be used to combine several alarms (i.e. for optimizing larger configuration or simply if more than 5 conditions are needed).*

Warning: ***The UserEvent is not recommended to be used as it may produce "endless loops" that can slow down the system performances or may affect the stability of other functions. Use the UserEvents at your own risk. When using the UserEvents, think about all consequences of (maybe recursively) launching alarms, especially in combination with various states, which may influence itself by actions. System behaviour***

can be very unpredictable and complex. Therefore no support will be provided for configurations that contain UserEvents.

4.2.23. Sys.WhiteList

A Whitelist is a user configurable list which contains text. Its main purpose is for authentication purposes – whenever incoming data should to be compared against a large list of text, it is possible now to do so in an elegant way – using 1 alarm only instead of many. Currently Whitelist can be used for SerialData events and 1-Wire Register/Release events

> **Example**: If a security tag being read by a barcode or RFID reader attached to a serial port of the device, its transmitted data can be automatically checked against contents stored within this list. If comparison succeeds, authentication i.e. to a car's engine can be granted.

4.2.23.1. Sys.Whitelist.Info – Counts and shows the number of entries in the whitelist

Command Syntax	Sys.Whitelist.Info
Example	$PFAL,Sys.Whitelist.Info

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to count the number of entries defined in the list.

Parameter description

None.

4.2.23.2. Sys.Whitelist.Show – Shows the contents of the entries defined in the whitelist

Command Syntax	Sys.Whitelist.Show
Example	$PFAL,Sys.Whitelist.Show

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to show the list of the entries defined within the list.

Parameter description

None.

Notes

Please note that, depending on the stored text length of the entries, the list may not show all entries already stored in the whitelist.

4.2.23.3. Sys.Whitelist.Clear – Erases all whitelist entries

Command Syntax	Sys.Whitelist.Clear
Example	$PFAL,Sys.Whitelist.Clear

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to erase all entries in the whitelist.

Parameter description

None.

4.2.23.4. Sys.Whitelist.Set<index>=<"text"> – Adds or edits an entry to/from the whitelist

Command Syntax	Sys.Whitelist.Set<index>=<"your_text">
Example	$PFAL,Sys.WhiteList.Set0="Example text 0" $PFAL,Sys.WhiteList.Set1="Example text 1" ... ... $PFAL,Sys.WhiteList.Set4999="Example text 4999"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to write one entry inside the whitelist. If an entry exists on the specified index, it will be overwritten with the new text.

Parameter description

Parameter	Value	Meaning
<index>	Sets the index of the specified text.	
	0 4999	The index in which the specified text will be saved.
"<your_text>"	Enclosed in quotation marks, it specifies the alphanumerical text (may contain letters, numbers) to be assigned to the defined index. The specified text is converted internally in to the upper-case characters before comparing with incoming data. Maximum text length for one entry is 20 characters/bytes. You can store 5000 entries with a max. text length of 15 byte or 2480 entries with a max. text length of 20 bytes.	

4.2.23.5. Sys.Whitelist.Get,<index> – Reports the assigned text of the specific index

Command Syntax	Sys.Whitelist.Get,<index>
Example	$PFAL,Sys.WhiteList.Get,0 $PFAL,Sys.WhiteList.Get,1 ... ... $PFAL,Sys.WhiteList.Get,4999

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to read the text assigned to the specified index.

Parameter description

Parameter	Value	Meaning
<index>	Sets the index of the specified text.	
	0 4999	The index in which the specified text will be saved.

4.2.24. Sys.Bat

4.2.24.1. Sys.Bat.Voltage – Queries the current battery voltage

Command Syntax	Sys.Bat.Voltage
Example	$PFAL,Sys.Bat.Voltage
Responses	$<SYS.BAT.Voltage> $battery voltage: 3.541V $SUCCESS $<end>

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	✓

Command description

This command requests the current voltage of the internal battery. The returned value is in Volt (V). If battery is charging at the request time, this command reports incorrect voltage. Therefore, this command returns a warning if battery is being charged.

Parameter description

None.

4.2.24.2. Sys.Bat.ChargeMode – Enables and disables the battery charging

Command Syntax	Sys.Bat.ChargeMode[=<mode>]
Example	$PFAL,Sys.Bat.ChargeMode=auto

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	✓

Command description

This command either reads the current state of the battery mode (if no parameter is set) or changes the mode of charger operation from enable to disable and vice versa (if one of the following supported parameters is used). This setting will be stored within non-volatile memory and is restored during start up - the device memorizes this setting. Emergency case: If the internal battery is completely discharged, the battery charger enables automatically in order to provide a minimal charge, which is used to start GSM safely. However, the configured chargemode setting is not affected by this – so after next wakeup/restart or system reset, the old chargemode setting becomes active again.

Parameter description

Parameter	Value	Meaning
<mode>	Sets the battery charging mode.	
	disabled	The internal battery is never charged – *regardless if enough external power is detected or not.* This mode should be used only if power consumption of the device must be reduced. **Example:** If a car observes power consumption and prohibits normal power consumption when switched off. It is not recommended to use this option if e.g. the device is connected to an external power supply always. Even in this case, it might be desirable to have an additional power supply in case the external one drops (emergency cases).
	auto	The internal battery is recharged if external power is present (**≥ 9V**) and charging temperature is not exceeded. Charging stops when the internal battery is fully charged.
	eco	(Default) It operates as auto mode, but internal battery is charged only if: ◆ Device has been started and the battery is not fully charged ◆ Device is running and the battery is below 3.9V. ◆ During regular device operation, battery is not charged in short cycles (which happens in auto mode when battery slightly drops below 4,15V). ◆ Charging is stopped as soon as the internal battery is fully charged, and it will be resumed when battery voltage drops below 3.9V.

Note: *If the internal battery is completely discharged, the battery charger enables automatically. A GPSTATE message will appear if the battery is discharged completely.*

Note: *Battery charge mode can also be changed by using the configuration setting DEVICE.BAT.CHARGEMODE=<function>.*

4.2.24.3. Sys.Bat.ChargeState – Gets the current battery charge status

Command Syntax	Sys.Bat.ChargeState
Example	$PFAL,Sys.Bat.ChargeState
Responses	$<SYS.BAT.ChargeState> $chargemode: eco $battery voltage: 4.034V $battery is fully charged $SUCCESS $<end

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	✓

Command description

Returns a text indicating the current battery status as follow:

Command	Description
charging	Indicates that the battery is currently charging.
not charging	Indicates that the battery is not charging (no external power is available).
full	Indicates that the battery is full charged.

Additionally the above text the selected ChargeMode is also displayed.

Parameter description

None.

4.2.24.4. Sys.Bat.Mode – Set the battery power mode

Command Syntax	Sys.Bat.Mode=<mode>
Example	$PFAL,Sys.Bat.Mode=auto

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	✓

Command description

This command selects the operating mode of the internal battery. This setting will be stored within non-volatile memory and is restored during startup.

Parameter description

Parameter	Value	Meaning
<mode>	Specifies operating mode of the internal battery.	
	disabled	(default) The internal battery is not used for normal operations, so the device can start only if a sufficient external power supply is detected.As soon as external power becomes unavailable (*i.e. disconnected or voltage drops below limit*), the device shuts down immediately. This mode should be used only if sufficient external power supply can be assured.
	auto	The device starts with or without external power. Whenever sufficient external power (**≥ 8 V**) is detected, it will be used by the device. As soon as external power drops below a limit of **8 V**, the device switches to battery usage. This allows to continue operation of the device even if external power is disconnected. In contrast to mode **always**, the device is able to fully charge its internal battery, if external power is available. This mode is best used for consuming as less power as possible from the internal battery (*without having the risk of an immediate shutdown like in "**disable**" mode*). In this case it is recommended to use **ChargeMode=auto.** If battery power gets too low to assure correct functionality of the device and NO external power is detected, a battery power critical event is created. In this case it is possible to execute a few important actions before going asleep (*i.e. sending an SMS/TCP message etc.*). Special care must be taken if this mode should be used with **ChargeMode=disabled**. This combination is desirable in situations where the device may not consume power – especially not charging its battery. If external power is available and there are no restrictions in using it, battery **ChargeMode** should be switched to "**auto**" whenever possible to allow the internal battery being charged. As soon as power usage is restricted (*i.e. vehicle is turned off*), battery **ChargeMode** should be disabled. In order to increase the operating time of the device using only its internal battery, sleep modes should be entered as often as possible (i.*e. **IGN** could be used to wake up the device by an external digital signal, which provides lowest power consumption - **RING** as wakeup condition can be used to wake up the device remotely, but requires more power when sleeping*).
	always	The device uses the internal battery as power source regardless of external power. Disadvantages: Even when charging mode is set to **AUTO**, the battery can be never fully charged. Therefore operation time without external power is reduced using this feature. Advantages: The device uses less power from external power source until its internal battery is discharged to approx. **3.7** V. This mode is best used if the internal battery is charged completely and the device comes in a situation in which it should use as less external power as possible.

4.2.25. SYS.ModBus

The commands in this group define functions that can be used to interact with devices that support the ModBus/RTU protocol over a serial connection (*https://en.wikipedia.org/wiki/Modbus*). The most important terms are listed here for the later description of specific commands.

<station> - The station or slave address of the connected device.

Each ModBus device must have a unique address. The ModBus devices can then be addressed

via their address and an addressed client responds with the requested data.

Addresses in the range from 1 to 247 are used for addressing the devices and the corresponding device is selected with this setting.

<Endianness> - The endianness format used to represent the register values.

Only the representation of 16-bit register values is defined in the ModBus protocol. If floating point or 32/64-bit register values have to be read from a connected ModBus device, the representation of the register values depends on the respective manufacturer.

The implemented ModBus functions therefore offers the possibility of interpreting this value as little-endian formated (LE) or as big-endian formated (BE). If a value appears unlogical when reading a register, changing this parameter can ensure that the value is represented correctly.

<register> - The register address to read from.

The ModBus implementation uses a de facto standard that extends the readable register to 65536 by adding one digit to the address.

- coil numbers span from 000001 to 065536
- discrete input numbers span from 100001 to 165536
- input register numbers span from 300001 to 365536
- holding register numbers span from 400001 to 465536

Another way to note the data addresses is to use the hexadecimal value, which clarifies the use of the four digits in the traditional convention mentioned previously.

- coil numbers span from 0x0000 to 0xFFFF
- discrete input numbers span from 0x10000 to 0x1FFFF
- input register numbers span from 0x30000 to 0x3FFFF
- holding register numbers span from 0x40000 to 0x4FFFF

<format> - The data format which is used to display the register content.

- BIT displays a single bit value
- S16 displays a signed 16-bit value
- U16 displays an unsigned 16-bit value
- H16 displays a hexadecimal 16-bit value
- S32 displays a signed 32-bit value
- U32 displays an unsigned 32-bit value
- H32 displays a hexadecimal 32-bit value
- F32 displays a 32-bit floating point value
- S64 displays a signed 64-bit value
- H64 displays a hexadecimal 64-bit value

4.2.25.1. SYS.ModBus.Enable - Activates ModBus connection over the given interface

Command Syntax	SYS.ModBus.Enable,<Serial>[,<baudrate>]
Example	$PFAL,SYS.ModBus.Enable,Serial0,9600

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command Description

This command defines an interface for communication with an external ModBus device. It is necessary for communicating with a ModBus device (e.g., reading registers). The command only

needs to be specified once and saves the specified parameters in the DEVICE.MODBUS setting. After each device reset, this setting is recalled and applied automatically.
To guarantee the security of the connected device, only read access is implemented and the implementation cannot be used to make changes to the device.

Parameter Description

Parameter	Value	Meaning
<Serial>	The used serial interface for communication with the connected ModBus device. In the case of the FOX3 units you can specify Serial0 (on Main Port) or Serial1 (on Accessory Port) here.	
<baudrate>	Specifies the baudrate used for the ModBus communication. The baudrate can be specified from 1200 to 115200 baud. However, this parameter is optional. If this parameter is not specified, the interface's nominal baud rate is used (from the DEVICE.SERIAL0.BAUDRATE or DEVICE.SERIAL1.BAUDRATE setting).	

4.2.25.2. SYS.ModBus.Disable - Deactivates a ModBus connection

Command Syntax	SYS.ModBus.Disable
Example	$PFAL,SYS.ModBus.Disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command Description

Stops a ModBus connection over a serial interface. The FOX3 no longer attempts to query registers from a connected ModBus device. However, PFAL communication via the interface is only possible after a device reset (see *4.2.4.1.* SYS.Device.Reset).

Parameter Description

none

4.2.25.3. SYS.ModBus.ReadRegister - Reads a register from the connected ModBus device

Command Syntax	SYS.ModBus.ReadRegister,<"station:endianness,register:format">
Example	$PFAL,SYS.ModBus.ReadRegister="1:LE,400100:U32"

Command Description

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

The command reads the specified register from a connected ModBus device. The ModBus device is controlled under the station/slave address. The command returns the value of the addressed register in the specified format.

Parameter Description

Parameter	Value	Meaning
<station>	1 .. 247	Station/slave address of the ModBus device.
<endianness>	Representation of register content for 32/64-bit register values (LE - LittleEndian or BE - BigEndian representation).	
<register>	The register address to be read (see *4.2.25.* for the possible register addresses in the description).	
<format>	The representation format of the data of the read register (see *4.2.25.* for the possible representation formats).	

4.2.25.4. SYS.ModBus.Poll - Read periodically register values from the connected ModBus device

Command Syntax	SYS.ModBus.Poll,<"station:endianness,register:format:period[,register:format:period,...]"> SYS.ModBus.Poll,<"None"> SYS.ModBus.Poll,<"Query">
Example	$PFAL,SYS.ModBus.Poll="1:LE,400100:U32:10,400102:U32:10"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command Description

Th command is used to specify which registers are to be queried periodically from a connected device. The ModBus implementation reads the programmed registers in the background and updates the values of these registers. The register values are then available for display and can be used for a wide variety of outputs. See *Chapter 7: Dynamic Entries/Variables* on displaying the ModBus registers &(ModBus:register). The polling of the register values can be terminated with the "None" parameter. With the parameter "Query", the non-periodic register values can be read (the registers defined with period equal to 0).

Parameter Description

Parameter	Value	Meaning
<station>	1 .. 247	Station/slave address of the ModBus device.
<endianness>	Representation of register content for 32/64-bit register values (LE - LittleEndian or BE - BigEndian representation).	
<register>	The register address to be read (see *4.2.25.* for the possible register addresses in the description).	
<format>	The representation format of the data of the read register (see *4.2.25.* for the possible representation formats).	
<period>	The cycle time in seconds used to read the register.	

4.2.26. Sys.MSG

4.2.26.1. MSG.Event – Transmit and execute user defined commands

Command Syntax	MSG.Event[,<nterface>],<"text">
Example	$PFAL, MSG.Event, "door.open" $PFAL, MSG.Event,TCP,"door.open" $PFAL, MSG.Event,Serial0,"door.open" $PFAL, MSG.Event,Serial1,"door.open" $PFAL, MSG.Event,User,"door.open"
Responses	$<$PFAL,MSG.Event> $SUCCESS $<end>

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command allows you to send user defined commands via PFAL command to the FOX3 and receive answers on the server side about the transmission and execution of these user commands.

Example: **Example:**

Example:Sending from server:

```
$PFAL,MSG.Event,TCP,"door.open"
```

Device answers:

```
$<$PFAL,MSG.Event>
$SUCCESS
$<end>
```

Alarm configuration:

```
$PFAL,CNF.Set,AL10=TCP.Client.eReceived="door.open":msg.send.RawSerial0,0,"IO5.Set=hpulse,2000"&IO6.set=lpulse,500
```

Inform server about the execution status of user command "door.open":

```
$PFAL,CNF.Set,AL11=IO.e2=fedge:TCP.Client.Send,8,"event=door.open,OK"
```

Parameter description

Parameter	Value	Meaning
[,<interface>]	Optional. specifies the interface on which the event for the user defined command must be generated. If the interface parameter is omitted, then the device will answer/respond on the channel receiving the command/text.	
	Serial0	Generates the event (*Sys.eSerialdata0[="user_command_text"]*) on the serial port 0.
	Serial1	Generates the event (*Sys.eSerialdata1[="user_command_text"]*) on the serial port 1

Parameter	Value	Meaning
	USB	Generates the event (*Sys.eUSBData[="user_command_text"]*) on the USB port
	TCP	Generates the event (*TCP.Client.eReceived[="user_command_text"]*) for the first TCP channel
	TCP2	Generates the event (*TCP.Client2.eReceived[="user_command_text"]*) for the second TCP channel
	User	Generates the event (*SYS.eUserText=<"user_command_text">*) on the user channel
<text>	Defines the text or user defined command e.g. “door.open”, “door.close”.	

4.3. CNF

CNF commands are used to set and read all device configuration settings. They can be used to affect almost any device functionality and are very powerful commands - but this implies a huge number of different parameters. In order to increase a clearly arranged command set, configuration functions will be moved one by one to corresponding command groups in future software versions. Configuration parameters are stored within non-volatile system memory, so they remain active after a system reset/sleep mode or if power is removed. Each parameter can be set, modified, cleared and read out at any time. Note that changes might not become active immediately *(example: changing TCP/GPRS connection settings while being inside a running GPRS or TCP connection)*. A few parameters also require a system reset to become active. If a configuration setting is cleared, its default value becomes active again.

4.3.1. Cnf.Set,<parameter_name=value> - Checks/Sets/Stores the parameter settings into the device

Command Syntax	Cnf.Set,<parameter_name>
Example	$PFAL,Cnf.Set,GSM.PIN=1234 $PFAL,Cnf.Set,DEVICE.NAME=myFOX3 $PFAL,Cnf.Set,DEVICE.NAME $PFAL,Cnf.Set,TCP.CLIENT.CONNECT=1,212.119.014,0005 $PFAL,Cnf.Set,AL0=IO.e0=redge:IO4.Set=cyclic,1000,2000 ….

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command checks, sets and stores the parameters settings into the device. When setting the configuration settings for the parameters of external accessory devices e.g. IOBOX-MINI, IOBOX-CAN, IOBOX-WLAN that are not yet connected to the AVL device, this command will respond with error instead of success. This command checks first for the availability of the corresponding accessory device on the AVL device, before storing its settings. In order to save the parameter settings without checking the availability of such accessory devices, use **Cnf.Write** (see chapter 4.3.11.) instead of **Cnf.Set**.

Parameter description

Parameter	Value	Meaning
<parameter_name>	This parameter consists of two entries (<name>=<value>) that are by an equal (=) separated.	
	<name>	Specifies the parameter name which is a part of the configuration. Note that <parameter_name> always must be specified in CAPITAL letters. If small letters are inside - or the spelling is not exactly the same, this configuration setting would also be stored, but would never be used as it differs from the configuration. Specifying a <parameter_name> which is unknown for the system allows to store user information permanently, which might be desired for some applications.
	<value>	Defines the setting of that parameter. Usually this information is case insensitive, but special care should be taken to match the required syntax for the corresponding parameter name. Else the setting might not be stored or lead to unexpected system behavior.

Chapter 5: Configuration Settings shows the configuration parameters that are available in the AVL firmware.

Notes

- All parameter names must be written in capital letters, otherwise no configuration can be stored.
- The string format (the parameter name enclosed in double quotes " ") may not be used. (The text in double quotes "" would be stored as parameter value, which can cause syntax errors).
- The "$PFAL,Cnf.Set" command can be used to "overwrite" most default parameter values. When a configuration parameter is sent without settings to the AVL device (e.g $PFAL,Cnf.Set,DEVICE.NAME, $PFAL,Cnf.Set,AL0) its current configuration will be reset to default and remains active until overwritten by the new settings.

4.3.2. Cnf.Get,<parameter_name> - Gets parameter settings from device

Command Syntax	Cnf.Get,<parameter_name>
Example	$PFAL,Cnf.Get,DEVICE.NAME $PFAL,Cnf.Get,TCP.CLIENT.CONNECT ….

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Using this command the AVL device responds the current configuration settings corresponding to the specified <parameter_name>. The configuration settings are stored in the internal FLASH memory.

Parameter description

Parameter	Value	Meaning
<parameter_name>	It specifies the parameter name to get its settings. The list of parameter names can be found in chapter 4.3.1. Do not specify any value to the parameter names, otherwise the device responds with error.	

4.3.3. Cnf.Clear,<parameter_name> - Clears settings of the parameter name

Command Syntax	Cnf.Clear,<"parameter_name">"
Example	$PFAL,Cnf.Clear,"AL"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command can be used to clear single or groups of configuration parameters. Each configuration parameter will be reset to its default value if it is cleared. If the specified parameter name matches to several parameters (i.e. AL - matches to **AL0 ... 99 or 0 ...249**), all found parameters will be cleared *(i.e. all alarms)*.

Parameter description

Parameter	Value	Meaning
<parameter_name>	It specifies the name of the parameter, in capital letters, that is intended to erase its present settings. Set one of the parameters listed in the chapter 4.3.1. The parameter must be wrapped in quotation marks (" ").	

Notes

The name of the parameter (or the start characters) must be written in capital letters.

- No semicolon may be inside the specified parameter name.
- If several user configuration settings would match to the specified parameter name, a list of all deleted parameters is given out.
- The factory default settings cannot be erased.
- If many configuration settings are to be erased, this command might need several seconds to complete the process.

4.3.4. Cnf.ShowUser - Shows the configuration settings defined by user

Command Syntax	Cnf.ShowUser
Example	$PFAL,Cnf.ShowUser

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command delivers the whole parameter names stored in the internal FLASH memory of the AVL device, which are modified/added by the user.

Parameter description

None.

Notes

AVL device will deliver all parameter settings, except default configuration.

4.3.5. Cnf.ShowDefault - Returns default configuration settings

Command Syntax	Cnf.ShowDefault
Example	$PFAL,Cnf.ShowDefault

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command delivers all default settings stored in the internal FLASH memory of the AVL device.

Parameter description

None.

4.3.6. Cnf.Show - Returns all used parameter settings

Command Syntax	Cnf.Show
Example	$PFAL,Cnf.Show

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command delivers all currently used parameter settings stored in the internal FLASH memory of the AVL device.

Parameter description

None.

Notes

- Default parameter settings are displayed, if the user does not have modified them. For example, if the user changes the device name from default "unnamed FOX3" to the user name "myFOX3", the AVL device does not deliver that parameter value.
- All parameters will be displayed with the **Cnf.Show** if they result cleared by the user.

4.3.7. Cnf.Search,<parameter_name> – Searches for a parameter name

Command Syntax	Cnf.Search,<parameter>
Example	$PFAL,Cnf.Search,"DEVICE" $PFAL,Cnf.Search,"TCP"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command searches all parameter names, which match to the specified text. The found parameters are returned together with their current values.

Parameter description

Parameter	Value	Meaning
<parameter>	Determines the text, in capital letters, to be searched. The specified text must be wrapped in double quotation marks (""). The text to be searched can be one of the parameter names listed in chapter *4.3.1.*	

4.3.8. Cnf.Backup - Backup current user configuration

Command Syntax	Cnf.Backup
Example	$PFAL,Cnf.Backup

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command can be used to back up the current user configuration. The user configuration can be anytime restored either manually by using the command *$PFAL,Cnf.Restore*.

Parameter description

None

Notes

It is strongly recommended to execute this command after configuring the device to back up your configuration settings.

- A backup configuration should contain all required configuration settings used to get remote access of the device (i.e. SIM PIN, GPRS settings etc.). Also important Alarms can also be backed up.
- It is not recommended to save last valid position, almanac, counter, trigger or timer values to the configuration backup. If a restore command is executed, all backup settings are restored, which could mean i.e. wrong last valid position and even more important – an invalid startup time.

4.3.9. Cnf.EraseBackup – Erases the backed up user configuration settings

Command Syntax	Cnf.EraseBackup
Example	$PFAL,Cnf.EraseBackup

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command can be used to erase the backed up user configuration.

Parameter description

None

4.3.10. Cnf.Restore – Restores the backed up user configuration settings

Command Syntax	Cnf.Restore
Example	$PFAL,Cnf.Restore

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

If a configuration was backed up before, this command can restore all backup settings to the current configuration. The device also attempts to restore a backup configuration automatically if:

- a corrupted configuration is detected (new precaution feature),
- the current user configuration is empty when the device starts up,

Within the restore command, a system hot start is performed. This causes an update of most configuration settings (i.e. alarms, protocols, geofences etc.).

The GSM engine is also reinitialized, which aborts existing GPRS/TCP/SMTP sessions as well as voice / data calls. Start events will be shown after the completion.

Parameter description

None

4.3.11. Cnf.Write,<parameter_name=value> - Saves parameter settings to device

Command Syntax	Cnf.Write,<parameter_name>
Example	$PFAL,Cnf.Write,WLAN.CLIENT.LOGIN=1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is commonly used only to save the settings of the specified parameters. It does not check the availability of external accessory devices e.g. IOBOX-MINI, IOBOX-CAN, IOBOX-WLAN if they are connected to the AVL device or not.

Parameter Description

Parameter	Value	Meaning
<parameter_name>	It specifies the parameter name with the values to be stored. The syntax is (<name>=<value>), that are separated by an equal (=). The list of parameter name can be found in chapter 4.3.1.	

4.3.12. Cnf.Write,Device.CAN.Transceiver=<on> - Ensures proper operation of IO2 & IO3

Command Syntax	Cnf.Write,Device.CAN.Transceiver=<mode>
Example	$PFAL,CNF.Write,Device.CAN.Transceiver=on

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command can be used to reduce the influence on the IO2 on the IO3 (FOX3 series hardware revision 09 and older revisions) by dropping the voltage of these IOs when they are not used, so that no false events are generated in the unused state. Additionally to this description, refer to ***App Note: Use of I/Os on AVL Devices.*** See 1.3. Related documents.

Parameter Description

Parameter	Value	Meaning
<mode>	Defines whether or not to use corrections on IO2/ IO3 of FOX3 with **Hardware revision 09** and older:	
	on	Makes correction on the IO2 and IO3.
	off	No correction is made on the IO2 and IO3 (default).

Notes

This command may only be relevant for the Hardware Revision 09 and older revisions of FOX3 Series. Call this command “$PFAL,MSG.Version.HardwareRev” to get known the hardware revision of your FOX3 series you are using.

4.3.13. CNF.Load – Overwrites the existing config with a file from flash

Command Syntax	CNF.Load, <"/sys/file.txt">
Example	$PFAL,CNF.Load,"/sys/config.txt"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command allows the existing config to be overwritten with a file from flash.

Parameter Description

<"/sys/file.txt">

The file name

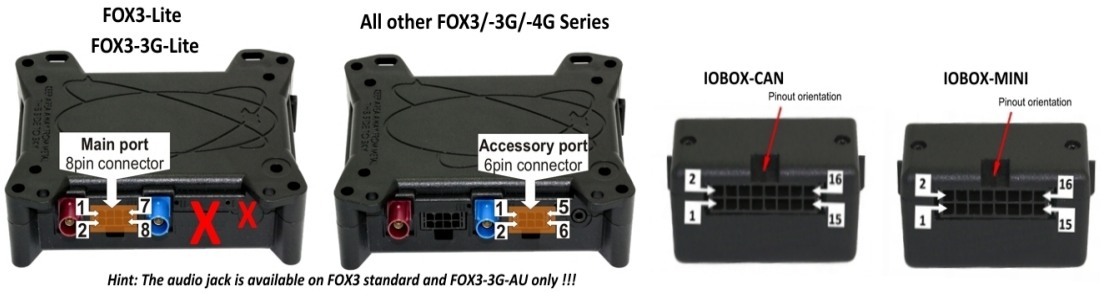
4.4. IO COMMANDS

4.4.1. Firmware-indexed IOs

The following commands can be used to read, set or modify existing inputs and outputs of the AVL devices. It is possible to define alias names for each IO index as well as for the group **IO** itself.

The figure below shows the pin out of Lantronix AVL devices and accessories on their connectors respectively.

Figure 4-1 Pin outs and accessories



This table lists the existing inputs and outputs of the AVL devices:

Table 4-4 Inputs and Outputs of AVL devices

IO indices	Default function	Hardware Assignment	Alternative function	Notes
FOX3/-3G/-4G Series				
0	DI	IO1 (Pin 4, sleep option AiWu)	AI* | OUT	
1	DI	IO2 (Pin 5)	AI | OUT	CanBus (CAN_L)**
2	DI	IO3 (Pin 6)	AI | OUT	CanBus (CAN_H)**
3	Not available			
4	DO	IO1 (Pin 4, if IO1 = Output)		
5	DO	IO2 (Pin 5, if IO2 = Output)		
6	DO	IO3 (Pin 6, if IO3 = Output)		
7	Not available			
8*	DI	IGN (PIN 3, sleep option IGN)		
9, 10	Not available			
11	DO	LED Orange		
12	DO	LED Green		
13	DO	LED Red		
BOLERO40 Series				
0	Not available			
1	DI	IO1 (Yellow)	AI* | OUT	

IO indices	Default function	Hardware Assignment	Alternative function	Notes
2	DI	IO2 (Green)	AI | OUT	
3,4	Not available			
5	DO	IO1 (Yellow, if IO1 = Output)		
6	DO	IO2 (Green, if IO2 = Output)		
7	Not available			
8*	DI	IGN (Blue, sleep option IGN)		
9-11	Not available			
12	DO	LED Green		
13	DO	LED Red		
0 … 13	Not available			
14	ANA	IOBOX Analogue input (PIN 7)		Don't need calibration
15	Not available			
16	DO	IOBOX OUT0 (PIN 9)		
17	DO	IOBOX OUT1 (PIN 11)		
18	DO	IOBOX OUT2 (PIN 15)		
19	DO	IOBOX OUT3 (PIN 13)		
20	DI	IOBOX DI0 (PIN 8)		
21	DI	IOBOX DI1 (PIN 10)		
22	DI	IOBOX DI2 (PIN 12)		
23	DI	IOBOX DI3 (PIN 14)		
24	DI	IOBOX DI4 (PIN 16)		
25	DI	IOBOX DI5 (PIN 2)		
26	DI	IOBOX DI6 (PIN 4)		
27	DI	IOBOX DI7 (PIN 6)		
IOBOX-CAN/WLAN				
0...15	Not available (PIN 10 = CAN-Hig; PIN12 = CAN-Low; PIN16 = GND)			
16	DO	IOBOX OUT0 (PIN 5)		
17	DO	IOBOX OUT1 (PIN 7)		
18	DO	IOBOX OUT2 (PIN 9)		
19	DO	IOBOX OUT3 (PIN 11)		
20	AI	IOBOX DI1 (PIN 2)	DI	
21	AI	IOBOX DI2 (PIN 4)	DI	
22	AI	IOBOX DI3 (PIN 6)	DI	
23	AI	IOBOX DI4 (PIN 8) – D8 feature	DI	
24...27	Not available			

AI = analog input; **DI** = digital input; **DO** = digital output;
*__IOs__ *can be used as wakeup condition (they can wake up the device when sleeping).*
***If CAN bus feature is available and used, both digital inputs are automatically disabled.*

Notes

Digital inputs raise events whenever their level (high/low) changes.

- Digital inputs can also be used as AI simultaneously (even when configured as DI).
- LED 11 ...13 are shown on the device case.

4.4.2. IO<index>.Set=<conf_type> – Defines the output behaviour

Command Syntax	IO<<index>.Set=<config_type>
Example	$PFAL,IO4.Set=high

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command specifies the output behavior of the desired IO. This command works only for digital output (DO) IO's.

Parameter description

<index>

It specifies the index of output ports and LED indicators. The index depends on the outputs and LEDs provided on the Lantronix FOX3, FOX3, FOX3-Lite and FOX3-3G respectively. Table below lists the index of LEDs and pins that can be used as output. For a full list of available **IO's**, please see *Table 4-4*.

Note: *Please note that the indices in grey are not supported.*

<index>	FOX3/-3G/4G Series Function (pin)	BOLERO40 Series Function (pin)	IOBOX-MINI Function (pin)	IOBOX-CAN Function (pin)
0	IO1 (Pin 4)	-	IO1 (Pin 4)	-
1	IO2 (Pin 5)	IO2 (Yellow)	IO2 (Pin 5)	-
2	IO3 (Pin 6)	IO3 (Green)	IO3 (Pin 6)	-
3	-	-	-	-
4	IO1 (Pin 4)	-	IO1 (Pin 4)	-
5	IO2 (Pin 5)	IO2 (Yellow)	IO2 (Pin 5)	-
6	IO3 (Pin 6)	IO3 (Green)	IO3 (Pin 6)	-
7	-	-	-	-
8	IGN (Pin 3)	IGN (Pin 3)	IGN (Pin 3)	-
9	-	-	-	-
11	LED orange	-	LED red	-
12	LED green	LED green	LED green	-
13	LED red	LED red	LED blue	-

<index>	FOX3/-3G/4G Series Function (pin)	BOLERO40 Series Function (pin)	IOBOX-MINI Function (pin)	IOBOX-CAN Function (pin)
14	-	-	Analogue Input (PIN 7) (do not calibrate)	-
15	-	-	-	-
16	-	-	OUT0 (PIN 9)	OUT0 (PIN 5)
17	-	-	OUT1 (PIN 11)	OUT1 (PIN 7)
18	-	-	OUT2 (PIN 15)	OUT2 (PIN 9)
19	-	-	OUT3 (PIN 13)	OUT3 (PIN 11)
20	-	-	DI0 (PIN 8)	AI0/DI0 (PIN 2)
21	-	-	DI1 (PIN 10)	AI1/DI1 (PIN 4)
22	-	-	DI2 (PIN 12)	AI2/DI2 (PIN 6)
23	-	-	DI3 (PIN 14)	AI3/DI3 (PIN 8)
24	-	-	DI4 (PIN 16)	-
25	-	-	DI5 (PIN 2)	-
26	-	-	DI6 (PIN 4)	-
27	-	-	DI7 (PIN 6)	-

<config_type>

It specifies the configuration type of the user-defined output. It can be set to:

Value	Meaning
high	Sets OUT<index> to the High level.
low	Sets OUT<index> to the Low level.
hpulse, <high_time>	Performs a single high pulse for the specified time. The output immediately switches to high level, waits <high_time> ms and then switches back to low level.
lpulse, <high_time>	Performs a single low pulse for the specified time. The output immediately switches to low level, waits <high_time> ms and then switches back to high level.
cyclic <high_time>,<low_time>,<[impulse_cnt]>]	This setting changes periodically the level of the specified output index for the specified times. The output immediately switches to high level, waits <high_time> ms, then switches to low level for <low_time> ms. This procedure is repeated until another functionality is specified. <high_time> Time in ms at which the port is set to high level. Min. time currently 125 ms. smaller values will be accepted but won't show a different behavior. <low_time> Time in ms at which the port is set to low level. Min. time currently 125 ms. smaller values will be accepted but won't show a different behavior. [impulse_cnt] The default value is 0, which results in infinite operation. The value may range between 0 and 255. This is an optional setting for the option "cyclic" setting. It specifies the number of cycles (one cycle is considered as high pulse and following the low pulse) after all pulses have been set, the cyclic functionality stops, and it changes to “low" function.

4.4.3. IO<index>.Get - Returns the current function and level of IO

Command Syntax	IO<index>.Get
Example	$PFAL,IO4.Get

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command reads out the current function and level of the desired **IO**. This command works for all **IO's**, regardless of their current function.

Expected answers (examples):

```
DI<index>= highHigh level on input
DI<index>= lowLow level on input
AI<index>= 3.452Voltage on input (3 digits
accuracy)
DO<index>=highHigh level on output
DO<index>=lowLow level on output
```

Parameter description

Parameter	Value	Meaning
<index>	It specifies the index of digital input. For a full list of available IOs, please see *Table 4-4*.	

4.4.4. IO<index>.GetDI – Returns the level of the digital inputs

Command Syntax	IO<index>.GetDI
Example	$PFAL,IO0.GetDI

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command reads out the level of a desired digital input (DI) IO. This command works for digital input (DI) IO's only and will return an error for any other IOs.

Expected answers (examples):

```
high      high level
low       low level
```

Parameter description

Parameter	Value	Meaning
<index>	It specifies the index of digital input. For a full list of available IOs, please see *Table 4-4*. t	

4.4.5. IO<index>.GetAI – Returns the level of the analog inputs

Command Syntax	IO<index>.GetAI
Example	$PFAL,IO0.GetAI

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command reads out the level of a desired analog input (AI) IO. This command works for analog input (AI) IO's only and will return an error for any other IOs.
Expected answers (examples):

```
<voltage>current voltage on this IO (3 digits
accuracy)
```

Parameter description

Parameter	Value	Meaning
<index>	It specifies the index of digital input. For a full list of available IO's, please see *Table 4-4*.	

4.4.6. IO<index>.GetDO – Returns the level of the digital output

Command Syntax	IO<index>.GetDO
Example	$PFAL,IO0.GetDO

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command reads out the level of a desired analog input (AI) IO. This command works for analog input (AI) IOs only and will return an error for any other IOs.
Expected answers (examples):

```
high      high level
low       low level
```

Parameter description

Parameter	Value	Meaning
<index>	It specifies the index of digital input. For a full list of available IO's, please see *Table 4-4*.	

4.4.7. IO<index>.Config - Configures the functionality and behaviours of IOs

Command Syntax	IO<index>.Config=<function>,[<AI_calibration>],[<DI_calibration>]
Example	$PFAL,IO0.Config=AI $PFAL,IO0.Config=DI,0.5,2.2

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

All settings specified within this command will be stored within non-volatile memory and are loaded during startup of the device. Therefore, it is not necessary to send this command after each start up. This command also affects stored calibration settings within the device configuration – so it may alter, remove or add calibration settings to the device configuration. Some configuration settings may be used only for **IOs** showing a specific functionality, which is shown in the following list. Note that settings within *square brackets []* are optional.

Parameter description

Parameter	Value	Meaning
<index>	It specifies the index of digital input. Table below lists the index of pins that can be used as input. For a full list of available IO's, please see *Table 4-4*. To know the pinout of these devices please refer to the chapter *4.4.* Please note that the indices in grey are not supported.	

	FOX3/-3G/-4G Series	BOLERO40 Series	IOBOX-MINI	IOBOX-CAN
<index>	**Function (pin)**	**Function (pin)**	**Function (pin)**	**Function (pin)**
0	IO1 (Pin 4)	-	IO1 (Pin 4)	-
1	IO2 (Pin 5)	IO2 (Yellow)	IO2 (Pin 5)	-
2	IO3 (Pin 6)	IO3 (Green)	IO3 (Pin 6)	-
3	-	-	-	-
4	IO1 (Pin 4)	-	IO1 (Pin 4)	-
5	IO2 (Pin 5)	IO2 (Yellow)	IO2 (Pin 5)	-
6	IO3 (Pin 6)	IO3 (Green)	IO3 (Pin 6)	-
7	-	-	-	-
8	IGN (Pin 3)	IGN (Pin 3)	IGN (Pin 3)	-
9	-	-	-	-
11	LED orange	LED red	LED red	-
12	LED green	LED green	LED green	-
13	LED red	LED blue	LED blue	-

	FOX3/-3G/-4G Series	BOLERO40 Series	IOBOX-MINI	IOBOX-CAN
<index>	**Function (pin)**	**Function (pin)**	**Function (pin)**	**Function (pin)**
14	-		Analogue Input (PIN 7) (do not calibrate)	-
15	-		-	-
16	-		OUT0 (PIN 9)	OUT0 (PIN 5)
17	-		OUT1 (PIN 11)	OUT1 (PIN 7)
18	-		OUT2 (PIN 15)	OUT2 (PIN 9)
19	-		OUT3 (PIN 13)	OUT3 (PIN 11)
20	-		DI0 (PIN 8)	AI0/DI0 (PIN 2)
21	-		DI1 (PIN 10)	AI1/DI1 (PIN 4)
22	-		DI2 (PIN 12)	AI2/DI2 (PIN 6)
23	-		DI3 (PIN 14)	AI3/DI3 (PIN 8)
24	-		DI4 (PIN 16)	-
25	-		DI5 (PIN 2)	-
26	-		DI6 (PIN 4)	-
27	-		DI7 (PIN 6)	-

<function>

It sets the function of the IO line index. It can be set to:

IO functionality	Used as	Notes, configuration settings
Pure DI (digital input)	DI	No configuration is required for this IO
Pure DO (digital output)	DO	No configuration is required for this IO
IO0..2 DI (digital input) with alternative option AI (analog input)	AI	Optional calibration settings can be specified for each analog IO Syntax: IO<index>.Config=AI[,<AI_calibration>]
	DI	If an analog Port should be used as a digital port, an optional DI setting can be specified to define at which input voltage the digital level is considered as low or high. Additionally, analog calibration settings may be specified to configure the based analog IO pin. *Syntax:* IO<index>.Config=DI[,<DI_calibration>][,<AI_calibration>]

Notes

- Alias names can be used for the index (see chapter above).
- Digital inputs raise events whenever their level (high/low) changes. Analog inputs won't raise events.
- Os can also be used as analog inputs simultaneously (even when configured as digital input).

[<AI_calibration>]

Optional. If it is specified, the syntax of <AI_calibration> is:

[<AI_calibration>]	<min_voltage>,<max_voltage>[,<offset>,<max_value>]

This setting is optional and contains analog calibration settings. Minimal and maximal shown voltages can be specified. These voltages belong to the corresponding calibration settings and can be reconfigured without running a calibration again.

After a firmware update or a factory reset, a configuration command might be required in order to configure the previous settings again (no calibration needs to be performed in this case).

Example:

Let's assume that 3V are applied to IO0. These 3V are the offset of the sensor connected to this IO. Therefore, to calibrate this IO to 3V, the command **$PFAL,IO0.Calibrate,offset=3** should be send to the device. The device will show now 3V when this offset level is applied to IO0.

Let's assume that the application requirement changes and this sensor (e.g. a temperature sensor) measures exactly 5 degree when outputting 3V on IO0.

In order to show degrees instead of voltages, the shown offset voltage can be reconfigured by this configuration command.

Now the device outputs 5 (degrees) instead of 3 (volts).

The calibration itself doesn't have to be performed again because the same offset voltage is used.

Additionally (and optional again), calibration values <offset> and <maxvalue> may be specified.

Please refer to next chapter **"Calibrate"** of how to create and query these settings.

Usually, **<offset>** and **<maxvalue>** settings don't have to be specified. These settings are recommended only if the device must be reconfigured for e.g. another application, which uses different calibration levels. Usually a new calibration must be performed in this situation. If the calibration values are already known (i.e. because the device has been calibrated in the past and these values were read out), they can be specified here and will overwrite existing calibration values. Doing so allows to recalibrate the device over air - without a manual calibration.

It is strongly recommended to use the exact calibration values within the configuration (see command "IO<index>.Calibrate,<type>=<voltage>" for more details).

<min_voltage>	It is a signed decimal value (in volts). Basically any voltage can be specified. Also a fractional value can be specified ranging from **0** to **.999**. It specifies the measured voltage at <offset>.	**Example:** 0 = 0.000 V 1.1 = 1.100 V 1.123 =*1.123*
<max_voltage>	It is a signed decimal value (in volts). Basically, any voltage can be specified. Also, a fractional value can be specified ranging from **0** to **.999**. It specifies the measured voltage at <offset>.	**Example:** 0 = 0.000 V 1.1 = 1.100 V 1.123 =1.123 V

[<offset>]	It is a hexadecimal value (specified without 0x) ranging from 0x00 to 0x7FF. Its default value is 0xDE, this value specifies the internal ADC measurement value for the specified <min_voltage>. It can be read out of the configuration of this IO after a calibration procedure.	
	◆ *Specific behavior if calibration values exist for this IO.* ◆ *In case a calibration has been performed before for a specific IO, these calibrated values will have priority over re-configuration of <offset> and <maxvalue> settings.* ◆ *Reason of this is to refuse a possibility to "corrupt" calibration settings of a device in field with simply using a wrong configuration file (containing possibly wrong configuration values for <offset> and <gain>* ◆ *If no calibration has been performed for a specific IO (which is default status for any new device), configuration parameters can be used* ◆ *If it is desired to use only configuration parameters and NOT parameters from a calibration, these parameters must be deleted (please refer to the command IO<index>.ClearCalibration in chapter 4.4.9.).*	
[<max_value>**]**	It is a hexadecimal value (specified without ox) ranging from 0x00 to 0x7FF. Its default value is 0x6F2, this value specifies the internal ADC measurement value for the specified <max_voltage>. It can be read out of the configuration of this IO after a calibration procedure.	
	◆ *Specific behavior if calibration values exist for this IO.* ◆ *In case a calibration has been performed before for a specific IO, these calibrated values will have priority over re-configuration of <offset> and <maxvalue> settings* ◆ *Reason of this is to refuse a possibility to "corrupt" calibration settings of a device in field with simply using a wrong configuration file (containing possibly wrong configuration values for <offset> and <gain>* ◆ *If no calibration has been performed for a specific IO (which is default status for any new device), configuration parameters can be used.* ◆ *If it is desired to use only configuration parameters and NOT parameters from a calibration, these parameters must be deleted (please refer to the command IO<index>.ClearCalibration in chapter 4.4.9.).*	

[<DI_calibration>]

Optional. If it specified, the syntax of <DI_calibration> entry is:

[DI_calibration]	<max_low_voltage>,<min_high_voltage>

This setting contains digital calibration settings for an analogue IO pin. These settings are used to define at which input voltage the digital level is considered as *LOW* or *HIGH*.

<max_low_voltage>	This setting specifies the maximal voltage to detect a low level (i.e. 0.500 V - voltages between 0 and 0.5 V will be detected as low level always). The default value is 0.3. Note: if the detected input voltage is between <max_low_voltage> and <min_high_voltage>, the detected digital level depends on the previously detected level. This assures, there is no "undefined" voltage range. Please also see the following example:	**Example** <max_low_voltage>=**0.3** <min_high_voltage>=**0.8**
	An input voltage of 2V will be detected as high level. Let's assume this voltage decreases slowly to 0.25V. If this voltage is at 0.7V, the detected level will be still high. As soon as the voltage reaches 0.3V or drops below, low level will be detected. This low level will be reported as long as the voltage doesn't reach 0.8V again. Format: a signed decimal value (in volts). Basically any voltage can be specified. Also a fractional value can be specified ranging from 0 to .999. It specifies the measured voltage at <offset>.	**Example** 0 = 0.000 V 1.1 = 1.100 V 11.123 V
<min_high_voltage>	This setting specifies the minimal voltage to detect a high level (i.e. 7.500 V - voltages greater or equal 7.5 V will be detected as high level always). The default value is 0.8. ◆ *If the detected input voltage is between* ***<max_low_voltage>*** *and* ***<min_high_voltage>,*** *the detected digital level depends on the previously detected level. This assures, there is no "undefined" voltage range. Please, see also the following example.*	**Example** <max_low_voltage>=**0.3** <min_high_voltage>=**0.8**
	An input voltage of 2V will be detected as high level. Let's assume this voltage decreases slowly to 0.25V. If this voltage is at 0.7V, the detected level will be still high. As soon as the voltage reaches **0.3V** or drops below, low level will be detected. This low level will be reported as long as the voltage doesn't reach 0.8V again. Format: a signed decimal value (in volts). Its range is nothing to care about – basically any voltage can be specified (ranges from **– 2 exp 31 -1** to **+2 exp 31 -1**. Also, a fractional value can be specified ranging from 0 to .999 It specifies the measured voltage at <offset>.	**Example** 0 = 0.000 V; 1.1 = 1.100 V; 11.123 V

4.4.8. IO<index>.Calibrate - Calibrates the offset or gain of analog input

Command Syntax	IO<index>.Calibrate,<type>=<voltage>
Example	$PFAL,IO0.Calibrate,offset=0 $PFAL,IO0.Calibrate,gain=15,55

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command works for analog (AI) and digital inputs which have an alternative AI option. Calibration should be done for each device. It increases the accuracy of analog measurements and allows to adapt the device to e.g. sensors or voltages to measure. During the calibration proce-

dure, 2 different voltage levels are adjusted and specified together with their true voltage levels to the device. The specified voltages are stored within device configuration and can be changed anytime without having to run a calibration again. The internal calibration values are stored as parameters within BIOS, which allows to even run a firmware or BIOS update without having to run a calibration again. Only if the application requirements change, a manual calibration needs to be done again. This happens e.g. if sensors with other voltage levels will be used.

This command affects stored calibration settings within the device configuration – so it may alter, remove or add calibration settings to the device configuration.

It is ***strongly recommended*** to calibrate first the ***offset*** and then calibrate the ***gain*** of an IO. Within Notes, a simple step by step description can be found on how to correctly calibrate the analog IOs.

How to correctly calibrate analog IOs (e.g. IO0).

- Connect the corresponding IO hardware pin to the GND pin of the AVL device.
- Send the command **$PFAL,IO0.Calibrate,offset=0**
- Connect the corresponding IO pin to the desired maximal voltage (max. **40V** !!!)
- Send the command **$PFAL,IO0.Calibrate,gain=13.60** (assuming that 13.60**V** are currently applied to that **IO0**)
- Read out the set calibration with **$PFAL,IO0.Info.** The device responses with:

```
<IO0.Info>
$current level=13.600 current function AI
$AI parameter min:0.000, max:13.600, offset:002h, gain:48Ch
$AI internal voltage factors: m:160, d:10
$SUCCESS
```

- To delete the calibration use **$PFAL,IO0.ClearCalibration**
- To set the calibration to all other devices remotely, which have the same specifications regarding the min. and max. voltages use $PFAL,IO0.Config=AI,0.000,13.600,002,48C. It means, you can calibrate a device locally and send the calibration values with the command $PFAL,IO0.Config to all other devices which are already deployed on field without the need to calibrate them locally.

The same procedure can be done for all other analog IOs.

Note: *How to display a percentage instead of the exact voltage:*

*Usually each value read out from an analog IO is measured in volts. If you are not interested in the specific voltage but for example desire the percentage (x% of maximum), you can achieve this by performing the first 3 steps above and for the 4th step, you send the command **PFAL,IO0.Calibrate,gain=100** (for AVL device you must assure that no more than **40V** are applied to this pin!!!).*

Note: *How to connect sensors having a systematic offset and showing negative sensor values:*

*Usually negative values cannot be shown - as the voltage specification for analog input pins is between **0V** and **40 V**.*

*If you have e.g. a temperature sensor which detects temperatures from **- 50.5°C** to **+ 50.5°C** and outputs a voltage between e.g. **5V** and **10 V** (**5V = - 50°C, 10V = 50.5°C**), you can configure the analog input to show the temperature instead of a voltage between 5V and 10 V.*

- Connect 5V to the IO (or your sensor at **- 50.5°C**).
- Send the command **PFAL,IO0.Calibrate,offset=-50.5**
- Connect 10V to the IO (or your sensor at **+50.5°C**).

- Send the command **PFAL,IO0.Calibrate,gain=50.5**

Parameter description

Parameter	Value	Meaning
<index>	Specifies the index of digital output. For a full list of available IO's, please see table *Table 4-4*.	
<type>	Defines the type to be calibrated for an analog IO (*see step by step description*).	
	offset	This is the type which should be calibrated for an analog IO (See step by step description above). Be sure that the lowest voltage to be measured is applied to the corresponding hardware IO pin before sending this command.
	gain	This is the second which should be calibrated for an analog IO (See step by step description above). Be sure that the highest voltage to be measured (**<=40V**) is applied to the corresponding hardware IO pin before sending this command.
<voltage>	It is a signed decimal value *(in volts)*. Basically any voltage can be specified. Also a fractional value can be specified ranging from **0** to **.999**. It specifies the lowest voltage (for offset command) or highest voltage (for gain command) to be measured on this IO. **Example: -5.1 = -5.001 V; 12 = 12.000 V; 1.123 = 1.123 V**	

Notes

Alias names can be used for this index (see chapter above). Please see also the chapter above for IO voltage specification.

4.4.9. IO<index>.ClearCalibration– Erases the stored calibration values for IO

Command Syntax	IO<index>1.ClearCalibration
Example	$PFAL,IO0.ClearCalibration

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command works for analog (AI) and digital inputs which have an alternative AI option. It erases stored calibration values for a specific IO. Usually these values are non-volatile and kept even during factory resets and/or Firmware updates, clearing configuration etc.

Notes

- If a calibration has been performed wrongly (i.e. an IO has invalid calibration values), the device should be recalibrated instead of simply erasing calibration
- If a device is already in field (not reachable locally) or for any other reasons no local IO calibration can be done, this command can be used.
- After this command is sent, optionally configured analogue <offset> and <maxvalue> parameters will be used to setup this IO. In case these optional parameters are not present, default values will be taken.

Parameter description

Parameter	Value	Meaning
<index>	It specifies the index of digital input. For a full list of available IO's, please see *Table 4-4*.	

4.4.10. IO<index>.Info – Returns the current function of specified IO

Command Syntax	IO<index>.Info
Example	$PFAL,IO0.Info

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command works for all IO's, regardless of their current function. Expected information depend on IO type. For any IO, its type is shown plus:

```
DI Logical level for digital input function only.
DO Logical level for output function, high and low times.
AI (used as DI) Logical level, low level voltage, high level
voltage, offset voltage, gain voltage, offset value, gain
value, internal voltage factors (depends on hardware option, is
not relevant for most users because this setting is used
internally only).
AI       (used as AI) Current voltage, offset voltage, gain
voltage, offset value, gain value internal voltage factors
(depends on hardware option, is not relevant for most users
because this setting is used internally only).
```

Parameter description

Parameter	Value	Meaning
<index>	It specifies the index of digital input. For a full list of available IO's, please see *Table 4-4*.	

4.4.11. IO.Counter

The IO Counters can be used to count quick impulses of an impulse width („high level width“) down to 20ms. The voltage of these impulses can be configured very flexibly, so it is no problem counting impulses with offset voltages. Counter values can be transmitted using dynamic variables i.e. **&(PulseCnt0)** usable for impulses of up to 25Hz (50% duty cycle) and approx. 15-20ms of „low level width“ are required to safely detect and count the next impulse.

4.4.11.1. IO<index>.Counter.Info – Returns the current state and the counter value of IO

Command Syntax	IO<index>.Counter.Info
Example	$PFAL,IO0.Counter.Info

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command shows the current state (active/passive) and the counter value of the specified IO.

Parameter description

Parameter	Value	Meaning
<index>	It specifies the index of digital input. For a full list of available IO's, please see *Table 4-4*.	
	0..3	The index of analogue inputs that are currently supported by the software.

4.4.11.2. IO<index>.Counter.Start – Starts the IO specified hardware

Command Syntax	IO<index>.Counter.Start=<v_min>,<v_max>
Example	PFAL,IO0.Counter.Start=0.5,1.5

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command starts the hardware counter for the specified IO.

Parameter description

Parameter	Value	Meaning
<index>	It specifies the index of digital input. For a full list of available IO's, please see *Table 4-4*.	
	0..3	The index of analogue inputs that are currently supported by the software.
<v_min>	It specifies the minimal “low level voltage". Voltages lower than this value are considered as low level. Voltages can also be entered as fractional values by using a point '.' as separator.	
<v_max>	It specifies the maximal “high level voltage". Voltages higher than this value are considered as high level. Voltages can also be entered as fractional values by using a point '.' as separator.	

Notes

- If one or several IO counters are required to work permanently, it is recommended to start them using an alarm (with device start event as a condition).
- Starting an alarm counter does not erase the current counter value (i.e. if it is set before start using the **IO<index>Counter.Set** command).

4.4.11.3. IO<index>.Counter.Set – Sets the value of a specified hardware counter

Command Syntax	IO<index>.Counter.Set=<value>
Example	$PFAL,IO0.Counter.Set=20

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command set the value of a specified hardware counter.

Parameter description

Parameter	Value	Meaning
<index>	It specifies the index of analogue inputs. For a full list of available analogue inputs, please see *Table 4-4*.	
	0..3	The index of analogue inputs that are currently supported by the software
<value>	It specifies integral counter value in the range from **0** to **2 exp 32 -1**.	

Notes

Alias names can be used for the index (see chapter above).

4.5. GPS COMMANDS

4.5.1. GPS.Nav

Commands within this command index can be used to save the last valid positions, retrieve the distances, assign, load or store specific positions.

The distance between different positions and the device can be used inside alarms.

Positions are used for alarm configuration only. Their purpose is to temporarily store GPS positions. Alarm actions may be launched depending on the distance to a defined position.

The distance counter can be used to calculate the number of metres covered by the device. The counter can be retrieved at any time e.g. even during a trip.

Saving a last valid position is used to always assure a valid position. Usually directly after system startup (before GPS gets a valid fix for the first time) there is no available last valid position. If a position has been saved in the past (before device shuts down), this position can be used as **"last valid position"** until GPS gets a new fix.

4.5.1.1. GPS.Nav.Position<buffer_index> – Returns bee-line distance, device to stored location

Command Syntax	GPS.Nav.Position<buffer_index>
Example	$PFAL,GPS.Nav.Position0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command retrieves the number of meters *(bee-line distance)* between the device and the specified position *(which must be assigned first)*.

Parameter Description

Parameter	Value	Meaning
<buffer_index>	It specifies the memory buffer index, in the range from **0** to **4,** to be read. Each memory buffer index is a piece of SRAM memory that is used to temporarily store/ read the data. The content of each buffer index is available as long as the internal software is running. Should the device be reset, switched off, or goes into sleep mode, each index loses its contents forever. Each memory buffer index has a fixed size and each of them can be updated with new data or the available data on them can be erased.	

4.5.1.2. GPS.Nav.Position<buffer_index>=<type> – Saves temporarily or clear device position

Command Syntax	GPS.Nav.Position<buffer_index>=<type>
Example	$PFAL,GPS.Nav.Position1=current $PFAL,GPS.Nav.Position2=none $PFAL,GPS.Nav.Position2=pos50.683317,10.980760,490.0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Current GPS position of the device can be temporarily stored into the specified <buffer_index>. After a position is stored into a <buffer_index>**,** the alarm conditions can be specified to launch certain action based on the distance calculated from the saved position to the current device position.

Parameter Description

Parameter	Value	Meaning
<buffer_index>	It specifies the memory buffer index, in the range from **0** to **4**, to save information for possible further use. Each memory buffer index is a piece of SRAM memory that is used to temporarily store the data. The content of each buffer index is available as long as the internal software is running. Should the device be reset, switched off, or goes into sleep mode, each index loses its contents forever. Each memory buffer index has a fixed size and each of them can be updated with new data or the available data on them can be erased.	
<type>	Determines the type of the data to be saved. Following types can be set.	
	none	Clears the contents of the selected <buffer_index>.
	current	Stores the current GPS position of the device into the selected <buffer_index>

Parameter	Value	Meaning
	pos <lat>,<lon>,<alt>	Stores the latitude, longitude and altitude of a location temporarily. This location enables to execute certain distance-based actions. As reference, see description of the of states "**GPS.Nav.Position.s**. The <lat>,<lon> and <alt> are given in decimal format. The <alt> determines the altitude, in meters, above sea level. If you do not know exactly the altitude of that location then specify a circa value without decimal dot "." (as an integer value).

Notes

- The functionality of this command enables periodically sending of messages that are based on distance covered by the device.
- The current GPS position will be stored into a <buffer_index>, if the <buffer_index> results valid.

4.5.1.3. GPS.Nav.Position<buffer_index>=save<slot_id> – Moves the GPS data from buffer to storage

Command Syntax	GPS.Nav.Position<buffer_index>=save<slot_id>
Example	$PFAL,GPS.Nav.Position3=save0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Saves the position slot to a storage slot. This is used to memorize the position during e.g. a system reset or a shutdown (i.e. IGN shutdown).

Parameter description

Parameter	Value	Meaning
<buffer_index>	It specifies the buffer index, in the range from **0** to **4**, to restore its contents into the FLASH.	
<slot_id>	The ID of the slot which is used to store the state. Only 5 storage slots *(from 0 to 4)* are available in the device for all Timer, Counter, Trigger and GPS.Nav.Position.	

Notes

An Alias name can be defined for each storage index by using ALIAS.STORAGE<storage_index>=<alias_name>.

4.5.1.4. GPS.Nav.Position<buffer_index>=load<slot_id> – Loads the data temporarily from storage to buffer

Command Syntax	GPS.Nav.Position<buffer_index>=load<slot_id>
Example	$PFAL,GPS.Nav.Position3=load0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command loads the contents of the selected <slot_id> into the selected <buffer_index> for temporarily use. The selected buffer index will be automatically refreshed with new data while the data into the <slot_id> remains unchanged. The data selected from the storage index must be validated (means, the <slot_id> must contain only GPS position data and no other data like Timer or Trigger states) before making any attempt to access the new data loaded in the selected <buffer_index>.

Parameter description

Parameter	Value	Meaning
<buffer_index>	It specifies the buffer index, in the range from **0** to **4**, to load it with new data.	
<slot_id>	The ID of the slot which is used to load the state. Only 5 storage slots (from 0 to 4) are available in the device for all Timer, Counter, Trigger and GPS.Nav.Position.	

Notes

An Alias name can be defined for each storage index by using ALIAS.STORAGE<storage_index>=<alias_name>.

4.5.1.5. GPS.Nav.Distance – Reads the distance travelled

Command Syntax	GPS.Nav.Distance
Example	$PFAL,GPS.Nav.Distance

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command retrieves the distance in meters from current distance counter. There is a dynamic variable &(NavDist) that can be used in alarms (AL) to report the travelled distance to your server. Dynamic variables are listed in chapter *7:*

Parameter description

None.

Notes

- The distance covered from the device (since the last device start up) is permanently added to the distance counter if GPS has a valid position and if DOP values are sufficient for Geofence usage (see configuration reference – geofence setting to get details of the defined maximal dop value).
- To retrieve the distance between 2 specific positions the **SetDistance** command must be used to reset the counter on the first position (e.g. when a trip starts). To get the number of

meters covered by the device (e.g. during this trip), simply read out the distance when the end position of this trip is reached.

4.5.1.6. GPS.Nav.Distance=<value> – Sets a user distance

Command Syntax	GPS.Nav.Distance=<value>
Example	$PFAL,GPS.Nav.Distance=0 $PFAL,GPS.Nav.Distance=10

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Sets the distance counter to a fixed value. This command can be used either to reset the distance counter *to 0 (zero)* or to use an offset distance from which the counter starts. To avoid standing problems and GPS jumps the values up to 10 meters are not counted.

Parameter description

Parameter	Value	Meaning
<value>	Specify the number of meters the distance counter starts with. Usually this value is 0 to reset the counter on a specific position.	

Notes

- The distance covered from the device (since startup or last ‚counter reset') is permanently added to the distance counter, if the device has a valid GPS position and if DOP values are sufficient for Geofence usage (see configuration reference – geofence setting to get details of the defined maximal DOP value).
- To retrieve the distance covered by the device after a certain point, simply reset the counter when the device reaches the desired point (e.g. when a trip starts or stops). To get the number of meters covered by the device (e.g. during or after a trip), simply read out the distance counter.

4.5.1.7. GPS.Nav.Distance.Save – Stores the distance

Command Syntax	GPS.Nav.Distance.Save
Example	$PFAL,GPS.Nav.Distance.Save

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command stores the current distance counter to non-volatile memory. The stored value will be automatically loaded during startup, so no further commands are required.

Parameter description

None

4.5.1.8. GPS.Nav.Distance.Load – Loads distance from memory

Command Syntax	GPS.Nav.Distance.Load
Example	$PFAL,GPS.Nav.Distance.Load

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command loads the current distance counter from non-volatile memory. The stored value will be automatically loaded during startup, so no further commands are required at system start.

Parameter description

None

4.5.1.9. GPS.Nav.DeltaDistance – Reads delta distance counter

Command Syntax	GPS.Nav.DeltaDistance
Example	$PFAL,GPS.Nav.DeltaDistance

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Retrieves the distance in meters from current delta distance counter. here is a dynamic variable *&(DeltaNavDist)* that can be used in alarms (AL) to report the delta distance to your server. *Dynamic variables are listed in chapter 7:*

Parameter description

None.

Notes

The distance covered from the device (since startup or last ‚counter reset') is permanently added to the distance counter, if GPS has a valid position and if DOP values are sufficient for Geofence usage (see configuration reference - geofence setting to get details of the defined maximal DOP value).

To retrieve the distance covered by the device after a certain position, simply reset the counter when the device is at the desired position. (e.g. when a trip starts). To get the number of meters covered by the device (e.g. during or after a trip), simply read out the distance.

4.5.1.10. GPS.Nav.DeltaDistance=<value> – Sets a delta distance

Command Syntax	GPS.Nav.DeltaDistance=<StartDistance>
Example	$PFAL,GPS.Nav.DeltaDistance=0 $PFAL,GPS.Nav.DeltaDistance=10

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Sets the distance counter to a fixed value. This function can be used to either reset the distance counter (set it to 0) or to use an offset distance from which the counter starts.

Parameter description

Parameter	Value	Meaning
<StartDistance>	Specify the number of meters the distance counter starts with. Usually this value is 0 to reset the counter on a specific position.	

Notes

The distance covered from the device (since startup or last ‚counter reset') is permanently added to the distance counter, if GPS has a valid position and if DOP values are sufficient for Geofence usage (see configuration reference – geofence setting to get details of the defined maximal DOP value).

To retrieve the distance covered by the device after a certain position, simply reset the counter when the device is at the desired position. (e.g. when a trip starts). To get the number of meters covered by the device (e.g. during or after a trip), simply read out the distance.

4.5.1.11. GPS.Nav.Distance2 – Reads total distance covered since initial startup

Command Syntax	GPS.Nav.Distance2
Example	$PFAL,GPS.Nav.Distance2

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Retrieves the distance in meters from current distance 2 counter. There is a dynamic variable *&(NavDist2)* that can be used in alarms (AL) to report this distance to your server. *Dynamic variables are listed in chapter 7:*

Parameter description

None.

Notes

- Distances are stored non-volatile when device goes to sleep (Sys.Device.Sleep or Sys.Device.Shutdown)
- Stored distances are loaded automatically at system start
- The distance covered from the device is permanently added to Distance2 counter if:
  - A) a distance from>= 10 metre has been covered and the device is moving (GPS speed>1m/s)
  - B) a distance>500m has been covered by the device and the device is not moving (GPS speed <1m/s)

- GPS has a valid position. DOP values are sufficient for Geofence usage (see configuration reference – geofence setting to get details of the defined maximal dop value)

4.5.1.12. GPS.Nav.Distance2=<value> – Sets a fixed distance value

Command Syntax	GPS.Nav.Distance2=<value>
Example	$PFAL,GPS.Nav.Distance2=0 $PFAL,GPS.Nav.Distance2=10

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Sets the distance2 counter to a fixed value. This function can be used to either reset the distance2 counter (set it to 0) or to use an offset distance2 from which the counter starts. This value will automatically increment each driven meter. Even when the device performs a reset this value is available in the FLASH memory. It can be read with the PFAL command $PFAL,GPS.Nav.Distance2. The stored data in the FLASH memory will not be lost during device reset or battery replacement.

Parameter description

Parameter	Value	Meaning
<value>	Specify the number of meters the distance2 counter starts with. Usually this value is 0 to reset the counter on a specific position.	

Notes

- For further details regarding specific operation please refer to “Read Distance2 Counter"
- At next device sleep/shutdown, a newly set value (including added trip distances) will overwrite the internally stored value.

4.5.1.13. GPS.Nav.SetHeadingTolerance=<value> – Defines heading tolerance

Command Syntax	GPS.Nav.SetHeadingTolerance=<value>
Example	$PFAL,GPS.Nav.SetHeadingTolerance=22

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Defines the tolerance for heading feature. The tolerance is stored into the non-volatile memory, and will automatically be used after the system boots up. Each time the specified angle is exceeded, the event ***GPS.Nav.eChangeHeading*** occurs, and the driving direction resets to zero. This state is checked each second and only when the actual vehicle speed exceeds *3.6 km/h*.

Figure below ***(Google Maps™)*** represents graphically the way points in which an event occurs whenever the device deviates the direction for more than 22 degrees.



x - points in which the event "GPS.Nav.eChangeHeading" occurs
In this example, the heading tolerance is set to 22°

Parameter description

Parameter	Value	Meaning
<value>	Defines the angle in degrees to occur the corresponding event. It ranges from **0** ... **359**.	
	0	Default setting. its disables heading and its event.
	1...359	Enables heading and its event (see notes for more details about the suggested value range).

Notes

- It is NOT recommended to use values below 10 and above 320 degrees. The lower the value, the more often the event occurs.
- Values above 180 degrees doesn't make much sense, the suggested value is 45° (default).

4.5.1.14. GPS.Nav.ResetHeading – Resets heading

Command Syntax	GPS.Nav.ResetHeading
Example	$PFAL,GPS.Nav.ResetHeading

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Resets the heading feature to the heading of the currently used GPS position.

Parameter description

None.

Notes

The occurrence of a heading event can be prevented when executing this command periodically.

4.5.1.15. GPS.Nav.SetHeading2Tolerance=<value> – Defines the Heading2 tolerance

Command Syntax	GPS.Nav.SetHeading2Tolerance=<value>
Example	$PFAL,GPS.Nav.SetHeading2Tolerance=22

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Defines the tolerance for heading feature. The tolerance is stored into the non-volatile memory, and will automatically be used after the system boots up. Each time the specified angle is exceeded, the event ***GPS.Nav.eChangeHeading2*** occurs, and the driving direction resets to zero. This state is checked each second and only when the actual vehicle speed exceeds *3.6 km/h*.

Figure below ***(Google Maps™)*** represents graphically the way points in which an event occurs whenever the device deviates the direction for more than 22 degrees.



x - points in which the event "GPS.Nav.eChangeHeading2" occurs
In this example, the heading tolerance is set to 22°

Parameter description

Parameter	Value	Meaning
<value>	Defines the angle in degrees to occur the corresponding event. It ranges from **0** ... **359**.	
	0	Default setting. its disables heading and its event.
	1...359	Enables heading and its event (see notes for more details about the suggested value range).

Notes

It is NOT recommended to use values below 10 and above 320 degrees. The lower the value, the more often the event occurs.

- Values above 180 degrees doesn't make much sense, the suggested value is 45°.
- Default value is 0 (heading2 is disabled)

4.5.1.16. GPS.Nav.ResetHeading2 – Resets Heading2

Command Syntax	GPS.Nav.ResetHeading2
Example	$PFAL,GPS.Nav.ResetHeading2

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Resets the heading feature to the heading of the currently used GPS position.

Parameter description

None.

Notes

The occurrence of a heading event can be prevented when executing this command periodically.

4.5.1.17. GPS.Nav.SaveLastValid – Saves last valid position, if no GPS-fix valid

Command Syntax	GPS.Nav.SaveLastValid[=<time_seconds>,<lat>,<lon>,<lat>,<lon>,<alt>]
Example	$PFAL,GPS.Nav.SaveLastValid=1047388489 //15.03.2013 13:14:49

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

If the system AVL device is currently receiving invalid GPS position data, and this command is executed, then an empty RMC message will be internally stored *(an RMC protocol that contains only zero values)*. To prevent the system AVL device form storing invalid GPS data, use this command in an alarm (*AL2=.eShutdown:GPS.Nav.SaveLastValid*) that saves the last valid GPS position to non-volatile memory before the system initiates a shutdown process. This data, stored before the system performs a shutdown, are needed on the next power up scenario. So, if there is no valid GPS fix during the next system startup, then the last valid position, stored into non-volatile memory, will be automatically attached to ***"MSG.Info.Serverlogin"*** command and sent to the remote server as well as an invalid RMC protocol will be updated by the last valid position. In this way you will always receive/have valid GPS position instead of an invalid RMC protocol containing only zero values.

Additionally, as long as the system has no valid information about the GPS time during the startup, it always uses either the time from the last valid position (the latest known time, if available) or the time starting from ***"06.01.1980 00:00:00"***. The stored time can also be used until the device gets a valid GPS fix again, which allows showing a valid position even if the device has no valid GPS fix after system startup. Once a few GPS satellites are in view the estimated GPS time information can be shown. If this time is older than the last valid position stored in the device, then it will be discarded.

Parameter description

Parameter	Value	Meaning
<time_seconds>	Customized time of last valid position. It specifies the number of seconds after 6.1.1980 (0:0:0) Example: 1047388489 => 15.03.2013 13:14:49	
<lat>,<lon>	Customized coordinates (decimal degrees) of last valid position in fractional notation; max. 7 digits for fractional part. Example: 50.6732382,10.9807934	
<alt>	Customized altitude of last valid position, it specifies the altitude in meters above sea level in fractional notation (2 fractional digits). Example: 429.51	

Notes

Because the non-volatile memory is limited to several 100'000 write operations, it is strongly recommended to prevent periodically saving of the last valid position in a short period of time.

- The saved last valid position will always be displayed in the following device operation such as: during the system startup, when the system exists in an area without GPS coverage and whenever a valid GPS fix returns to invalid.
- This command may need up to 30 seconds to complete its task. To reduce the time to first fix (TTFF) process, the AVL device automatically loads the last stored valid position during the startup.
- This time can be used to write history records and perform data logging. In order to read out the history by time/date span, be sure that the stored time in these records is always up to date. Following the hints to assure correct processing:
  - To assure consistent history times, always store the last valid position before shutting down the device and write history entries after the system startup. Do not write any history entries between saving and device shutdown process.
  - It may never happen that the time of a record to be stored in the history is smaller than the time of the last record already logged in history storage. The time/date of each new record you write must always be greater than the prior stored record.

4.5.1.18. GPS.Nav.HoldPosition – Holds the current GPS position at this location

Command Syntax	GPS.Nav.HoldPosition=<mode>
Example	$PFAL,GPS.Nav.HoldPosition=1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command enables holding of GPS position at specific location to avoid jumps around from your real parking position.

Parameter description

Parameter	Value	Meaning
<mode>	Defines whether to activate holding of the GPS position (e.g. while the car is parked) or not.	
	0	Deactivates holding of GPS position.
	1	Activates holding of GPS position.

4.5.1.19. GPS.NAV.GNSS– Enables/Disables specific GNSS operations

Command Syntax	GPS.NAV.GNSS=<sat_system>
Example	$PFAL,GPS.NAV.GNSS=GLONASS $PFAL,GPS.NAV.GNSS=GPS,GLONASS $PFAL,GPS.NAV.GNSS=GPS,GLONASS,BEIDOU

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command enables to activate the satellite navigation system the device will use for positioning. By default, only tracking of *GPS and GLONASS satellite navigation systems are* enabled. Concurrent reception of up to 3 systems is supported in the firmware 2.16.x and 3.1.x and higher.

Parameter description

Parameter	Value	Meaning
[<sat_system>]	It specifies the satellite navigation system to be activated. Combining of up to 3 different Satellite Navigation system names in one command line is allowed. Names are comma separated.	
	GPS	Activates tracking of GPS satellite navigation system (default) and deactivates tracking of other navigation systems.
	GLONASS	Activates tracking of GLONASS satellite navigation system and deactivates tracking of other navigation systems
	GALILEO	Activates tracking of GALILEO satellite navigation system and deactivates tracking of other navigation systems
	BEIDOU	Activates tracking of BEIDOU satellite navigation system and deactivates tracking of other navigation systems.

4.5.1.20. GPS.Nav.DeadReckoning– Enables/Disables the use of Dead-Reckoning feature

Command Syntax	GPS.Nav.DeadReckoning=<mode>
Example	$PFAL,GPS.Nav.DeadReckoning=enable $PFAL,GPS.Nav.DeadReckoning=disable

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command allows to activate the use of Dead Reckoning function in the FOX3-3G-DR model. Per default configuration, this feature is disabled. Combining multi-GNSS (GPS, GLONASS, BeiDou, Galileo) with the untethered dead-reckoning technology in our FOX3-3G-DR improves position accuracy even where GNSS signals are partial or completely blocked or reflected, such as in urban canyons, tunnels, mines, underpasses, multi-level roads or parking garages. Applications with untethered dead-reckoning include service vehicles from the airports, port facilities, car-sharing, fire departments that require accurate positioning at all times.

Parameter description

Parameter	Value	Meaning
<mode>	It specifies whether or not to activate the DR feature per PFAL command. This setting is stored in the configuration parameter “DEVICE.GPS.CFG” and will be used after new system boot process.	
	enable	Activates the use of Dead Reckoning function in the FOX3-3G-DR model.
	disable	Deactivates the use of Dead Reckoning function in the FOX3-3G-DR model.

4.5.2. GPS.History

History data can be stored at non-volatile memory inside the device. This data can be read out later to retrieve positions, speeds and additional information during the specified timespan.

Special data/event logging possibilities have been added to the basic "position/speed" history. These functionalities allow to write history records even when not having a GPS fix.

As a general remark: writing history records without having a valid GPS fix should be avoided if each history position needs to have a completely reliable time-stamp.

In areas with bad GPS signal strength or for data/event logging purposes, you might have to write history records. Any records you write without having GPS fix are marked inside the history as "no fix". This eases identification of "totally reliable" data and those who aren’t. Please find further notes to this topic at the commands ***History.Read***, ***History.Write***, ***History.SetRead*** and ***GPS.SaveLastValid***.

Additional details to history and especially the binary history format can be found inside ***App Note: Transform History Binary Data in NMEA Format for AVL Devices.*** See *1.3. Related documents*

Note: *History feature isn't available once a remote Update has been started. History will be enabled again, after remote update finished OR a serial update (with option "erase whole flash") has been performed.*

4.5.2.1. GPS.History.Write,<add_prot_to_memory>,<"text"> – Stores GPS position data in history

Command Syntax	GPS.History.Write,<memory>,<"text">
Example	$PFAL,GPS.History.Write,20,"*enter_the_text_to_be stored*" $PFAL,GPS.History.Write,1,"" $PFAL,GPS.History.Write,0,""

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Writes a new history entry including current position, speed, number of satellites and time. Additional information can also be added to this history entry.

Parameter Description

Parameter	Value	Meaning
<memory>	It is a hexadecimal number which specifies the information added to the history entry. Note that, extended information numbers can be added if several information fields are required. *i.e.* to write GSM state and an alarm event to history, the corresponding number would be **0A**. Following are listed for additional information in hexadecimal value:	
	0x0**0**	Writes the current GPS position of the device.
	0x0**1**	Writes the current state of the Input and Output ***(I00..IO3)***.
	0x0**2**	Writes the current state of the GSM (field strength, cell id, area code, of incoming/outgoing SMS etc.)
	0x0**4**	Writes the current system operating mode, GPRS, PPP, TCP and system lifetime.
	0x0**8**	Reserved.
	0x**10**	Analogue values (IO0 ... IO3). Analogue voltage of IO0, IO1, IO3 are stored with 2 digits accuracy. Maximum allowed values: -320 to 320.99 (must be assured via IO calibration). Output (i.e. TXT-format) : 123.45 (fractional part contains of only 2 digits now).
	0x**20**	Writes the specified text from the <"text"> field (up to 99 characters available).
	0x**40**	Writes the current state of the Geofence areas (inside or outside of a marked area).
	0x**80**	Reserved.
<"text">	It is a string, which contains user information. If no user information must be written, this field can be left empty (which results in an empty string ""). Dynamic variables can also be specified inside this text string, see chapter 7:.	

Warning: ***Note that this would extremely increase the size of history entries, which means that the history gets filled much faster and therefore needs to be read out more frequently. If user message exceeds 254 bytes, an error will be reported, and no entry is written. As always no string end/ start sign may be inside the specified string.***

Warning: ***No more than a single history entry should be written each second. Background: Each history entry requires a unique Time-stamp (which has a resolution of ±1 second). If several records are written, the stored timing information is not reliable anymore.***

Notes

- Entries can be written without having a valid GPS fix. In such a case the history entries may store user messages, IN/OUT states or other additional information.
- When a record is stored in the history, it will be "invalid", if there is no GPS-fix currently available. However, each entry can store:

```
The internal last valid position (if available).
Or the last stored valid position (if available).
Or an empty data "ECEF:0,0,0".
```

- When there is no GPS-fix currently available, the most recent available time-stamp is used:

```
From Last valid position if it is the newest time-stamp of the
system.
Or from locally shown RMC if this is the most recent time.
else internal time is used that means the date can start from
06.01.1980.
 -> Only records written with a valid GPS fix show an absolutely
reliable time-stamp when being read out later. Please read Notes of
SetRead and Read for further details.
```

- In order to attach more than one additional information at once, enter the sum of each additional information, for example:

```
The hex value 7 means: IO + GSM + System states will be stored
together with current location of the device at once.
```

4.5.2.2. GPS.History.Clear - Clears the history memory

Command Syntax	GPS.History.Clear
Example	$PFAL,GPS.History.Clear

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Clears the complete history. The command **$PFAL,GPS.History.Write** will not be executed, during the history clear process.

Parameter Description

None.

4.5.2.3. GPS.History.GetStart– Returns the oldest date stored in the history memory

Command Syntax	GPS.History.GetStart
Example	$PFAL,GPS.History.GetStart

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Retrieves the oldest date stored in history.

Parameter Description

None

4.5.2.4. GPS.History.SetRead,all – Selects number of history records to download

Command Syntax	GPS.History.SetRead,all GPS.History.SetRead,<s_date>>,<s_time>-<e_date>,<e_time>
Example	$PFAL,GPS.History.SetRead,all $PFAL,GPS.History.SetRead,14.6.2005,10:5:20-14.6.2005,10:06:16

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Either the complete history is selected for reading or it specifies the start and end date for the next read command. Only entries between the specified timespan are read out. Please read "Notes" for further details.

Parameter Description

Parameter	Format	Meaning/Value
<s_date>	Specifies the start date. Its format is DD.MM.YYYY	
	DD	Represents the day as a number (**1** - **31**)
	MM	Represents the month as a number (**1** - **12**).
	YYYY	Represents the year as a four-digit number (**1900** - **9999**).
<s_time>	Specifies the start time. Its format is HH:MM:SS	
	HH	Represents the hour (**0** - **23**).
	MM	Represents the minute (**0** - **59**).
	SS	Represents the second (**0** - **59**).
<e_date>	Specifies the end date. Its format is DD.MM.YYYY	
	DD	Represents the day as a number (**1** - **31**)
	MM	Represents the month as a number (**1** - **12**).
	YYYY	Represents the year as a four-digit number (**1900** - **9999**).
<e_time>	Specifies the end time. Its format is HH:MM:SS	
	HH	Represents the hour (**0** - **23**).
	MM	Represents the minute (**0** - **59**).
	SS	Represents the second (**0** - **59**).

Notes

This command returns an answer containing the estimated space used by history. This number shouldn't be used to specify or define the number of bytes read out by **$PFAL,GPS.History.Read**. The true number of bytes being read out is returned by **$PFAL,GPS.History.Read** itself.

- In order to read out history by date/time span, be sure that time is always up to date when writing history records, assure the following:
  - You have always stored last valid position right before shutting down the device, and not writing history entries between saving and shutting down to assure consistent history times.
  - Each new record you write has a future time/date compared to the previous written record (i.e. it may never happen that a history record contains a time which is in the past compared to a record written before).
- If one condition of the above is not fulfilled, a **$PFAL,GPS.History.SetRead** by date/time span might not select all data you want to read.
- This happen if you do the following:
  - Store a last valid position and restart the device,
  - Write a record before having a valid GPS fix,
  - Restart the device again (meaning internal time is set to the old "last valid position" time - from the first step), then write a record before having a GPS fix.
- Doing so, 2 history entries are written which have exactly the same time and date.
- You can still read out such history files – but JUST with **$PFAL,GPS.History.SetRead,all** as a date/time span based search might return not all or just a part of the desired records if they exist multiple times inside history.

4.5.2.5. GPS.History.SetRead,<start_index>-<end_index> – Selects number of history indices to download

Command Syntax	GPS.History.SetRead,<start_index>-<end_index>
Example	$PFAL,GPS.History.SetRead,1-100

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command enables reading of specific number of lines in the history.

Parameter Description

Parameter	Value	Meaning
<start_index>	Specifies the start index (line), as a decimal number, for reading the history.	
<end_index>	Specifies the end index (line), as a decimal number, for reading the history	

Notes

This command returns an answer containing the selected lines in the history.

4.5.2.6. GPS.History.Read – Downloads selected history records in parts

Command Syntax	GPS.History.Read,fmt=<format>
Example	$PFAL,GPS.History.Read $PFAL,GPS.History.Read,fmt=**txt**

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command reads out the history. A binary format is returned. Binary data consists of a length indicator showing how many bytes have been read out. Currently there is a maximum of 512 Bytes which can be read out using a single read command.

If a history timespan (or complete history) contains more data, several read commands have to be issued. Each will return 512 byte packet of history data. The last packet will show a "readout completed" inside its answer *(right before "SUCCESS")*. So for larger history readouts, several read commands are required. Please refer ***App Note: Transform History Binary Data in NMEA Format for AVL Devices.*** See *1.3. Related documents*.

Parameter Description

Parameter	Value	Meaning
<format>	Optional. It defines the format of the history (logged) records to be downloaded. It can be defined to:	
	bin	Shows history in binary format (default).
	txt	Shows history entries in a special textual format. A brief description of this format can be found in the next sub-chapter.
	rmc	An RMC protocol is generated for each history entry.
	user	Only user defined texts are shown (specified in history extension 0x20). This allows to display user formatted history entries using Dynamic variables and static user texts.

Notes

The maximum number of bytes that can be downloaded at once is predefined to 512 bytes.

- To download the history records, consider that the start date/time and end date/time are based on the UTC Time, otherwise the stored history records will be downloaded in the wrong time.
- **Advantage**: Usually a complete history readout takes much time. No PFAL commands could be entered/executed within this timespan. Splitting history data to several packets allows the server to execute commands even when a history readout has been started. Reading history packets can be continued whenever desired (until the device is shut down or performed a reset).
- A new submitted "**History.SetRead**" command will reset the current process of reading out a history, so it is recommended to gain a user who reads out history exclusive access to the device.
- Keep in mind, that binary history data first starts with a length info (the first 2 bytes), indicating how many bytes of history data follow. The first 2 bytes are not included in the length info.

4.5.2.6.1. Reading history records in textual format

Each history entry is reported within a single line of the following format.

$<history_entry_standard>><history_entry_extension><CRLF>
$19.05.2008,06:01:55,1,7,50.7295234,13.2345688,571.64,0,1:10010110.10011110

Format		Example	Description
$		$	Start of records
<history_entry_standard>			<date>,<time>,<fix>,<minsats>,<lat>,<lon>,<alt>,<speed>
	dd.mm.yyyy	13.10.2006	Date: dd day, mm month, yyyy year separated by dots.
	hh:mm:ss	13:26.56	Time: hh hours, mm minutes, ss seconds separated by colons.
	x	1	GPS position validity: 0: GPS fix invalid 1: GPS fix valid
	xx	03	Number of satellites in view (0 to 15)
	dd.mmmmmm	50.673325	Latitude, a double value in decimal degrees format.
	dd.mmmmmm	10.980685	Longitude, a double value in decimal degrees format.
	dd.m	600.9	Altitude, an approximate height value (0 … 8000) above sea level in meter format.
	xxx	100	Speed, an integer (0 … 225) representing the speed value over the ground in meter/second format
<history_entry_extension>			<ext1>,<ext2>, ….,<extn>
	<ext_id>:<data>	1:10010110.10011110	It is an optional parameter and it is added only if extensions exist. Extensions are sorted before output, so it is assured that e.g. extension containing IN and OUT state will come first, before all other extensions. A complete set of extensions is shown in the table below (sorted: the upper entry comes first).
<CR><LF>			End of message termination

<ext>		<ext_id>:<data>		
<ext_id>	<data>	**Example**	**Description**	
0	<IN>.<OUT>	1:10010110.10011110	Separated by dots, it shows IN and OUT states. IN e.g. 10010110 (IN7... IN0) OUTe.g. 10011110 (OUT7... OUT0)	
1	<fieldstrength>.<area_code>.<cell_id>.<SMS_in>.<SMS_out>	2:20.5518.4caa.10.9	Separated by dots, it shows both the current GSM state: <fieldstrength> GSM field strength (0 to 31; 99=unknown) <area code> area/country code of GSM operator <cell_id> GSM cell ID <SMS_IN> Incoming SMS number <SMS_out> Outgoing SMS number	

	2	<GPRS>.<PPP>.<TCP>.<Main>.<Lifetime>	4:1.2.3.4.20000	Separated by dots, it shows the current system state: <GPRS> current GPRS state <PPP> current PPP state <TCP> current TCP state <Main> current main state <lifetime> current time since the device started (in milliseconds) States mentioned above are not further documented. They should be used for debugging purposes only (i.e. to report when a system state changes or how the system state was at a specific time). This information might be useful and should be sent within support requests.	
	3			Alarm event (not yet implemented).	
	4	<analog0>;<analog1>	5.24;5.24	the values of analog inputs <analog0> fractional number <analog1> fractional number	
	5	"<user_specified_text>"		Wrapped in quotation marks, it shows the user message. <user specified text>It can be either a simple text or outputs of the dynamic protocols.	
	6	<areas_h>.<areas_l>	40:2000.00FC	Separated by dot, it shows the area states (being inside or outside of an area) <areas_h> hexadecimal value of area 16..31 (area16 is the least significant bit) <areas_l> hexadecimal value of area 0..15 (area0 is the least significant bit) It is almost the same syntax as in the GPAREA protocol.	
	7	-	-	Reserved.	
<CRLF>					

4.5.2.6.2. Further notes for converting history data with special remark to data/event logging features

If a record has *"no GPS fix",* its position should be ignored for any navigation (the position is invalid, and if it is a differential record, its relative position will be 0 for dx, dy, dz). Furthermore, the shown time is no a valid GPS time. As long as the device has a backup battery and / or the external power does not drop, the internal RTC keeps running. All FOX3/-3G/-4G Series has an internal RTC, except the Lite versions.

This time is usually reliable in the following case:

The device has had a GPS fix after startup. This fix got lost due to bad GPS coverage. è The internal time stored inside this record is quite accurate (± a few seconds for a long time span)

This time is not reliable in the following case:

The device has no GPS fix after startup. Only the stored ***LastValid*** position *(and its time)* could be used to initialize the internal clock.

AVL device uses this time (***LastValid***) and increments it as long as no valid GPS time available. However, the internal time can be in the past *(depending on how long the device has been switched off after saving the* ***LastValidPosition*** *for the last time)*.

History records created with ***"last valid"*** times are ordered correctly – so you can assume which record happened before/after another one in the past *(➔ allows event /data logging)*.

Furthermore, time differences between single records are also correctly shown for a session:

In order to distinguish which records belong to a "session" in the past (device started, wrote history records and was sent to sleep later), you can do two things:

- Write a record containing special user data right before saving the last valid position and sending the device to sleep. Whenever, you read out this user data later, you know when the device was sent to sleep (your session ended).
- **As a general hint**: whenever a new "**full record**" is written, the device probably performs a restart.
  - If there is a time gap between the last differential record and the new full record, the device has been sent to sleep (this time gap shows how long the device has been sleeping/shut off)
  - If there is no time gap in between, the device wasn't sent to sleep, which means the time differences between all records of this session are absolutely reliable.

4.5.2.7. GPS.History.Push – Downloads all selected history records at once

Command Syntax	GPS.History.Push,<msg_output>,[<format>]
Example	$PFAL,GPS.History.Push,Serial0 $PFAL,GPS.History.Push,TCP,**fmt=txt**

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	×
2	✓	✓	✓	×
3	✓	✓	✓	✓

Command description

If a range of the history has been selected for readout, this command will automatically read the entire selection. Push creates several packets of data. Only one Push command must be executed in order to read a complete history *(or a part of it)*. A regular answer is created for this Push command. After this, ***History.Read*** will be called periodically until readout is finished.

Parameter Description

Parameter	Value	Meaning
<msg_output>	**Optional.** Defines the channel from which history data will be downloaded. It can be set to:	
	Serial0	Outputs history to serial port 0.
	Serial1[1]	Outputs history to serial port 1.
	USB[2]	Outputs history to USB port.
	CSD[3]	Outputs history via established CSD.
	TCP.Client	Outputs history via TCP (TCP connection must be available).

Parameter	Value	Meaning
[<format>]	**Optional setting** *(if left blank the history data is retrieved in binary format*). It defines the format of the history (logged) data to be downloaded at once via defined channel <*msg_output*>. It can be set to:	
	fmt=bin	Shows history in binary format (default).
	fmt-txt	Shows history entries in a special textual format. A brief description of this format is available in and Table 8: A complete set of extensions.
	fmt=rmc	An RMC protocol is generated for each history entry. NMEA checksum for each RMC protocol can be enabled by enabling the checksums within the parameter.
	fmt=user	Only user defined texts are shown (specified in history extension 0x20). This allows to display user formatted history entries using Dynamic variables and static user texts.

Notes

- Advantage: No multiple Read commands have to be specified.
- Disadvantage: During the history readout process, no commands or low priority alarm actions will be executed. It is also not possible to e.g. accept a voice call or send/receive SMS.
- The answers of this command are 100% compatible to answers generated from **History.Read**. See **History.Read** command notes for more information.

4.5.3. GPS.Geofence

In order to have a basic understanding of conditional logic and geographic coordinates, please refer to chapter *5.17.3.*

Geo-fencing can be used to set up different areas which can itself consist of several single geofences. Whenever the device enters or leaves such areas the corresponding events are generated. Furthermore, geofence states (*being inside an area or geofence – or - being outside*) can be used to set up alarms. Additionally a park position can be specified which might launch certain alarm actions if the device moves outside the defined range *(i.e. thief alarm)*.

Note: *If the park position feature is to be used, **GF0** as well as area0 should not be used otherwise (because GF0 and area0 dedicated for use with park position).*

If ***Park.Set*** / ***Park.Remove*** are not used, **GF0** and **AREA0** can be used as regular geofence/area.

4.5.3.1. GPS.Geofence.Park.Set – Places and activates parking area

Command Syntax	GPS.Geofence.Park.Set
Example	$PFAL,GPS.Geofence.Park.Set

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command activates a parking area (**GF.0**). It places the AVL device (vehicle) into a circular park area, where the current position (including Latitude and Longitude) of the AVL device is the center of circle and the user specified <park_radius> value (see GF.CONFIG parameter) is the radius, in meter, of the circular area. This geofence is automatically attached to **AREA0**.

Parameter Description

None.

Notes

The events **GF.e0=inside** and **AREA.e0=inside** are usually occurred from the **GPS.Geofence.Park.Set**. Both events (**GPS.GF.e0=inside** and **GPS.AREA.e0=inside)** can be used to confirm the proper activation of the park area.

- Usually **Park.Set** will cause the event **eGF.0=inside** and **eAREA.0=inside**. Both events can be used to confirm the proper activation of the park geofence.
- This command works also if the device has no valid position. In this case the last valid position will be taken. Please not that this might lead to an immediate alarm in case the device gets a fix. This happens if the device was moving without a fix and the park position is set. (background: it moved out of the park area defined by last valid position).
- If the AVL device has got a GPS-fix and it is valid, the **GPS.GF.e0=inside** and **GPS.AREA.e0=inside** will occur, which indicates that this park area is properly set up.
- To deactivate the park condition (without occurring the event **GPS.AREA.e0=outside**), use **GPS.Geofence.Park.Remove** command.
- This Geofence setting will be also written in the Flash memory, so even when the device performs a reset, the park zone and area will remain active until manual deactivated. When a system restart occurs, and the park area remains activated, the events **GPS.GF.e0=inside** and **GPS.AREA.e0=inside** are occurred again as soon as the AVL device receives valid GPS position data.

4.5.3.2. GPS.Geofence.Park.Remove– Disables an activated park area

Command Syntax	GPS.Geofence.Park.Remove
Example	$PFAL,GPS.Geofence.Park.Remove

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command deactivates an activated park area (GF.0).

Parameter Description

None.

Notes

The event **GPS.GF.e0=inside** and **GPS.AREA.e0=inside** are no longer available.

4.5.3.3. GPS.Geofence.GeoState,<geo_id>– Returns the state of a Geofence

Command Syntax	GPS.Geofence.GeoState,<geo_ID>
Example	$PFAL,GPS.Geofence.GeoState,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Reads out the state of the specified geofence *(whether the device is inside or outside)*. If configured, the name of the geofence will be also shown. Currently 100 geofences can be defined *(index 0 – 99)*.

Parameter Description

Parameter	Value	Meaning
<geo_ID>	Number from **0** to **99** which specifies the geofence to be read out.	

4.5.3.4. GPS.Geofence.AreaState,<area_id>– Read the state of an area

Command Syntax	GPS.Geofence.AreaState,<area_ID>
Example	$PFAL,GPS.Geofence.AreaState,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Reads out the state of the specified area *(whether the device is inside or outside)*. If configured, the name of the area will be also shown. Up to 32 areas in range *0 – 31* can be defined.

Parameter Description

Parameter	Value	Meaning
<area_ID>	Number from **0** to **31** which specifies the areas to be read out.	

4.5.4. GPS.MultiGeofence

PREMIUM-Feature "EXTENDED-GEOFENCES" should be activated to be able to use these commands.

Multi Geofencing allows to use significantly more circular geofences than regular geofencing can do. In contrast to regular Geofences, Multi Geofences are used by their index only – they do not provide the possibility to define names or combine these geofences into areas.

Currently up to 3000 circular geofences can be defined.

Whenever the device enters or leaves these geofences, corresponding events "inside" or "outside" are generated.

After Start, all geofence states are considered to be “outside" - as soon as GPS fix is valid, states my change according to the position of the device.

Notes

- Using this feature may cause significant impacts to system speed, response time of PFAL commands as well as delay outputs of PFAL protocols.
- It can further cause impacts to the execution of low priority alarms and should therefore be used with caution.
- The caused impact is directly proportional by the number of used multi-geofences.
- The multi-geofences already stored in the device can be deleted only with GPS.MultiGeofence.Clear and NOT with Sys.Device.FactoryReset.

4.5.4.1. GPS.MultiGeofence.Info - Shows the number of geofences being used

Command Syntax	GPS.MultiGeofence.Info
Example	$PFAL,GPS.MultiGeofence.Info $<GPS.MultiGeofence.Info> $MultiGeofence INFO: 4 / 3000 geofences defined $SUCCESS

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command is used to show how many geofences are currently being used. Note that geofence slots have to start from index zero. There may be no “empty" slots in between – otherwise only geofences before this empty slot will be used (and shown as used).

Parameter Description

None

4.5.4.2. GPS.MultiGeofence.Clear - Clears the list of geofences

Command Syntax	GPS.MultiGeofence.Clear
Example	$PFAL,GPS.MultiGeofence.Clear

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command is used to empty the slot of the geofences list. This command is required for all “overwrite" operations – before writing new data on a slot, all existing slots have to be erased with this command.

Parameter Description

None.

4.5.4.3. GPS.MultiGeofence.GetWP – Gets the position and radius of specific multi-geofence

Command Syntax	GPS.MultiGeofence.GetWP,<id>
Example	$PFAL,GPS.MultiGeofence.GetWP,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command reads out information about the specified waypoint ID. If it is configured, the position and radius of the waypoint ID will be shown.

Parameter Description

Parameter	Value	Meaning
<id>	Defines the identifier of the waypoint. It is a number ranging from **0** to **2999** which specifies which waypoint should be read out.	

4.5.4.4. GPS.MultiGeofence.SetWP – Sets the position and radius of specific multi-geofence

Command Syntax	GPS.MultiGeofence.SetWP,<id>>,<lat>,<lon>,<alt>,<radius>
Example	$PFAL,GPS.MultiGeofence.SetWP,0,50.673447,10.980627,470,2000

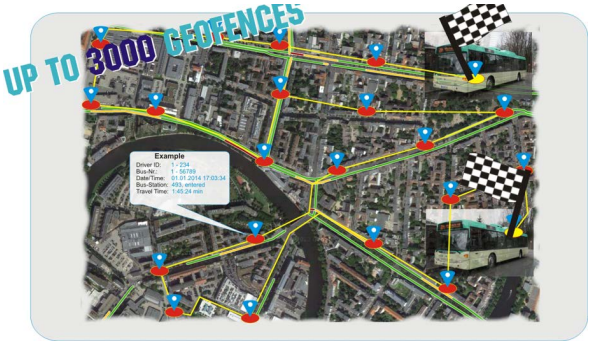
DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command adds (configures and activates) a new waypoint into the waypoint list to create corridor boundaries. For further information about waypoint functionality, please see example of minimal waypoint list below the important notes.

Note:

- *Note that geofence slots have to start from index zero. There may be no “empty" slots in between – otherwise only geofences before this empty slot will be used.*
- *Geofences do not need to be defined in ascending order. However geofences will be used only if there is no empty slot before.*
- *A “non-empty" (used) geofence slot may not be overwritten with other data - use a Clear command before.*



Parameter Description

Parameter	Value	Meaning
<id>	Specifies the waypoint ID in the range from 0 to 2999. Currently up to 3000 waypoints can be defined.	
<lat>	Specifies the latitude, in decimal degrees (e.g.: 50.673447)	

Parameter	Value	Meaning
<lon>	Specifies the longitude, in decimal degrees (e.g.: 10.980627)	
<alt>	This setting is not required and should be defined as 0. The value 0 is required to allow compatibility to tools used for creating waypoint Geofencing lists.	
<radius>	Specifies the radius of the circle, in meter. It also defines width of corridor to the previous waypoint. Only integral values are allowed (e.g.: 500). the covered area is considered as “inside" the geofence - whenever the device has sufficient GPS coverage, events will be created when entering or leaving the geofence. (See alarm configuration for more details).	

4.5.4.5. GPS.MultiGeofence.Test,<lat>,<lon> – Tests if GPS position is inside corridor boundaries

Command Syntax	GPS.MultiGeofence.Test,<lat>>,<lon>
Example	$PFAL,GPS.MultiGeofence.Test,50.673447,10.980627

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command tests all geofences if they are inside or outside the specified GPS position. Events will be created if a geofence state changes (i.e. an “inside" event would occur if the device was outside of this geofence right before a test command is entered specifying a position within the geofence). Right after the test command is finished, all geofence states restore to the previous state. (This might result in further events). ***Example:***

```
One configured geofence (Geofence 0) - device is outside
this geofence
 Test command is entered with a position within geofence
0 => Event GPS.MultiGeofence.e0=inside
Right after the test command state is reset => Event
GPS.MultiGeofence.e0=outside
```

Parameter Description

Parameter	Value	Meaning
<lat>	Latitude of a position to be tested (in decimal degrees - a fractional part is separated by dot i.e.: 50.673447).	
<lon>	Longitude of a position to be tested (in decimal degrees - a fractional part is separated by dot i.e.: 10.980627).	

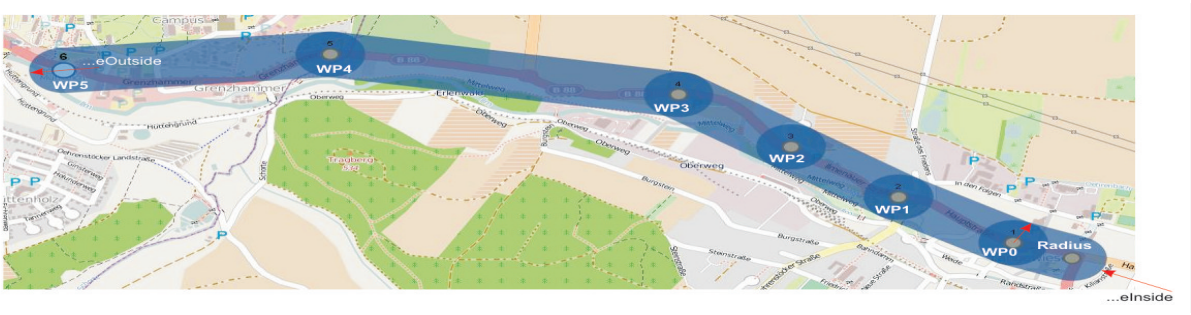
4.5.5. GPS.WPGeofence

In this section you will find a short description about the functionality of waypoints used in AVL devices.

What is a waypoint and what is it used for?

Waypoints are reference points or set of coordinates that identify GPS locations and helps you cover a route. A waypoint includes an identifier, latitude, longitude data and radius. An AVL device allows to store up to 2000 waypoints that are used to manage specific and very complex routes. The identifier of waypoints is a number that ranges from 0 to 1999. Specific routes may be established by using multiple waypoints, so you drive within a corridor from one waypoint to the next. Each time this corridor built of waypoints is entered or left does not meter from which side by the device the corresponding event GPS.WPGF.eInside or GPS.WPGF.eOutside is generated and the status of that corridor changes from GPS.WPGF.sOutside (during the device is being out of this corridor) to GPS.WPGF.sInside (during the device is being within this corridor) and vice-versa. The GPS fix and accuracy are very important factors. If the satellite geometry is poor the waypoint solution will be inaccurate.

How to store a waypoint into the list of way points, refer to the PFAL-Command GPS.WPGeofence.SetWP.



4.5.5.1. GPS.WPGeofence.Info – Shows information about waypoints

Command Syntax	GPS.WPGeofence.Info
Example	$PFAL,GPS.WPGeofence.Info

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

Shows information about the currently used set of waypoints and their configuration.

Parameter Description

None.

4.5.5.2. GPS.WPGeofence.Clear – Erases the entire waypoint list

Command Syntax	GPS.WPGeofence.Clear
Example	$PFAL,GPS.WPGeofence.Clear

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

Erases the entire waypoint list. This command is executed automatically when changing waypoints from 2D into 3D mode and vice versa.

Parameter Description

None.

4.5.5.3. GPS.WPGeofence.GetWP,<id> – Gets the position and radius of the waypoint

Command Syntax	GPS.WPGeofence.GetWP,<id>
Example	$PFAL,GPS.WPGeofence.GetWP,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command reads out information about the specified waypoint ID. If it is configured, the position and radius of the waypoint ID will be shown.

Parameter Description

Parameter	Value	Meaning
<id>	Defines the identifier of the waypoint. It is a number ranging from **0** to **1999** which specifies which waypoint should be read out.	

4.5.5.4. GPS.WPGeofence.SetMode2D – Change mode of waypoint to 2D (two-dimensional)

Command Syntax	GPS.WPGeofence.SetMode2D
Example	$PFAL,GPS.WPGeofence.SetMode2D

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command changes the setup mode from 3D to 2D and clears the waypoint list, if the default mode of the waypoints is 3D. For more details of its functionality, please refer to the PFAL-Command **GPS.WPGeofence.SetWP**.

Parameter Description

None.

4.5.5.5. GPS.WPGeofence.SetMode3D – Change mode of waypoint to 3D (three-dimensional)

Command Syntax	GPS.WPGeofence.SetMode3D
Example	$PFAL,GPS.WPGeofence.SetMode3D

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command changes the setup mode from 2D to DD and clears the waypoint list, if the default mode of the waypoints is 2D. For more details of its functionality, please refer to the PFAL-Command **GPS.WPGeofence.SetWP**.

Parameter Description

None.

4.5.5.6. GPS.WPGeofence.SetWP,<id>,<lat>,<lon>,<alt>,<radius> – Adds an entry to waypoint list

Command Syntax	GPS.WPGeofence.SetWP,<id>,<lat>,<lon>,<alt>,<radius>
Example	$PFAL,GPS.WPGeofence.SetWP,0,50.673447,10.980627,470,2000

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command adds (configures and activates) a new waypoint into the waypoint list. For further information about waypoint functionality, please see example of minimal waypoint list below the important notes.

Note:

- *All waypoints have to be defined in row (ascending order). I.e. waypoint 0 must be configured first, then waypoint 1 etc.*
- ***2D Mode:** When using 2D Mode, altitude must be still specified, but can be set to 0 or any other value. Remark: Altitude of waypoints will be ignored in 2D mode.*
- ***3D Mode:** When using 3D Mode, altitude plays an important influence on the behaviour of waypoints. It should be entered as accurate as possible (at least it should be roughly estimated).*

Example of minimal waypoint list:

A minimal waypoint list consists of 2 waypoints (which define a single corridor between waypoint 0 and 1.

The width of this corridor is specified by radius of waypoint 1.

A device is considered as being inside this corridor if it is:

- Inside the radius of waypoint 0
- Inside the radius of waypoint 1
- *Inside the corridor between waypoint 0 and 1, which means the device is near the direct connection between waypoint 0 and 1 (width of corridor 0 is defined by radius of waypoi*nt 1, with of corridor 1 by waypoint 2 and so on)

Parameter Description

Parameter	Value	Meaning
<*id*>	Specifies the waypoint ID in the range from 0 to 2999. Currently up to 3000 waypoints can be defined.	
<lat>	Specifies the latitude, in decimal degrees (e.g.: 50.673447)	
<lon>	Specifies the longitude, in decimal degrees (e.g.: 10.980627)	
<alt>	Specifies the altitude above sea level, in meter of the specified waypoint . It corresponds to the center of the circle (e.g.: 480.64). This parameter is required and very important for 3D mode. For 2D mode any value can be entered. The maximum accuracy is centimeters, so a fractional part separated by dot may be specified.	
<radius>	Specifies the radius of the circle, in meter. It also defines width of corridor to the previous waypoint. Only integral values are allowed (e.g.: 500).	

4.5.5.7. GPS.WPGeofence.Test,<id>,<lat>,<lon>,<alt>,<radius> – Tests if GPS position is inside corridor boundaries

Command Syntax	GPS.WPGeofence.Test,<id>,<lat>,<lon>,<alt>,<radius>
Example	$PFAL,GPS.WPGeofence.Test,0,50.673447,10.980627,470,10

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command tests if a specified position are within the corridors of the currently specified waypoint list. Note that this command is used for local test only and it can cause the creation of WPGF events.

Warning: ***2D Mode: When using 2D Mode, altitude must be still specified, but can be set to 0 or any other value. Remark: Altitude of waypoints will be ignored in 2D mode.***

Warning: ***3D Mode: When using 3D Mode, altitude plays an important influence on the behaviour of waypoints. It should be entered as accurate as possible (at least it should be roughly estimated).***

Parameter Description

Parameter	Value	Meaning
<id>	Specifies the waypoint ID in the range from 0 to 1999 that is already specified in the waypoint list.	
<lat>	Latitude of a position to be tested (in decimal degrees - a fractional part is separated by dot i.e.: 50.673447)	
<lon>	Longitude of a position to be tested (in decimal degrees - a fractional part is separated by dot i.e.: 10.980627)	

Parameter	Value	Meaning
<alt>	Specifies the altitude above sea level, in meter of the specified waypoint . It corresponds to the center of the circle (e.g.: 480.64). This parameter is required and very important for 3D mode. For 2D mode any value can be entered. The maximum accuracy is centimeters, so a fractional part separated by dot may be specified.	
<radius>	Specifies the radius of the circle, in meter. It also defines width of corridor to the previous waypoint. Only integral values are allowed (e.g.: 500).	

4.6. EcoDrive

The Eco-Drive commands are used to evaluate the cost-effectiveness and for estimating the fuel consumption of a vehicle. Using these commands, feet managers are able to create a path or route based statistics to the cost-effectiveness of the vehicle fleet. For each trip there are available measurement data such as length, time, speed, fuel consumption, speed, etc. To use the Eco-Drive commands a valid configuration must be stored on the AVL unit. *$PFAL,EcoDrive.TripStart* and *$PFAL,EcoDrive.TripStop* commands can be executed manually or as an Action in an alarm line if the *ECODRIVE.AUTOSTART=<Type>,<Speed>,<Timeout>* parameter is not set. In addition to the commands described here, there are also some events and dynamic variables available for evaluating a trip (see chapter 7:). The following commands are available for using EcoDrive features.

4.6.1. EcoDrive.TripStart - Starts a new EcoDrive trip

Command Syntax	EcoDrive.TripStart
Example	$PFAL,EcoDrive.TripStart

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command starts a new EcoDrive trip with new measured data. The current measurement data of the trip is available in the dynamic variable &(EcoTripCurData), which can be reported to a TCP server using an alarm. If you are going to use this command, do not specify any setting in the ECODRIVE.AUTOSTART parameter.

Parameter Description

None.

Notes

If the ECODRIVE.AUTOSTARTparameter for the EcoDrive autostart is already configured when you execute the $PFAL,EcoDrive.TripStart command, then the $PFAL,EcoDrive.TripStart command responds an ERROR.

4.6.2. EcoDrive.TripStop - Ends a started EcoDrive trip

Command Syntax	EcoDrive.TripStop
Example	$PFAL,EcoDrive.TripStop

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command ends the current trip. The current measurement data of the ended trip is available in the dynamic variable & (EcoTripResult), which can be reported to a TCP server using an alarm. If you are going to use this command, do not specify any setting in the parameter ECODRIVE.AUTOSTART.

Parameter Description

None.

Notes

If the ECODRIVE.AUTOSTART parameter for the EcoDrive autostart is already configured when you execute the $PFAL,EcoDrive.TripStart command, then the $PFAL,EcoDrive.TripStart command responds an ERROR.

4.6.3. EcoDrive.CurrentTrip - Reports current trip data

Command Syntax	EcoDrive.CurrentTrip
Example	$PFAL,EcoDrive.CurrentTrip

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command reports the data of the current trip even if it is still running.

Example: $PFAL,Ecodrive.CurrentTrip

$<EcoDrive.CurrentTrip>

$current trip 1 - *ID of the trip (automatically incremented).*

$current car "IK-AF260" - *Car name.*

$total time 2220 - *Total time of the trip (s).*

$total distance 45000 - *Total distance of the trip (m).*

$current speed 83.0 - *Current speed (km/h).*

$average speed 73.0 - *Average speed (km/h).*

$topologie country - *Current topology.*

$total fuel 7.500 - *Total fuel consumption (l).*

$current fuel 5.5 - *Current fuel consumption (l/100km).*

$idle time 0 - *Idle time (s).*

$cruise control time 0 - *Cruise control time (s).*

$data city 900,15000,1,12,7.900 - *Time,Distance,Overspeed,Counter,Fuel consumption (l)*

$data country 1200,27000,0,6.700 - Time,Distance,Overspeed,Counter,Fuel consumption (l)

$data highway 210,3000,0,5.300 - Time,Distance,Overspeed,Counter,Fuel consumption (l)

$invalid positions 0 - Counter of invalid GPS Info.

$SUCCESS

4.6.4. EcoDrive.LastTrip - Reports last trip data

Command Syntax	EcoDrive.LastTrip
Example	$PFAL,EcoDrive.LastTrip

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Command description

This command reports the data of the last trip.

Example: $<EcoDrive.LastTrip>

$current trip 97 - ID of the trip (automatically incremented).

$current car "IK-AH-74" - Car name.

$total time 3246 - Total time of the trip (s).

$total distance 80753 - Total distance of the trip (m).

$current speed 12.1 - Current speed (km/h).

$average speed 89.6 - Average speed (km/h).

$topologie None - Topology used in this trip.

$total fuel 6.624 - Total fuel consumption (l).

$current fuel 0.0 - Current fuel consumption (l/100km).

$idle time 0 - Idle time (s).

$cruise control time 2055 - Cruise control time (s).

$data city 819,9638,96,0.788 - Time,Distance,Overspeed,Counter,Fuel consumption (l)

$data country 2312,67036,114,5.473 - Time,Distance,Overspeed,Counter,Fuel consumption (l)

$data highway 15,4079,0,0.363 - Time,Distance,Overspeed,Counter,Fuel consumption (l)

$invalid positions 0 - Counter of invalid GPS Info.

$SUCCESS

Parameter Description

None.

4.7. GSM

4.7.1. GSM General

4.7.1.1. GSM.PIN=<"pin"> - Enters the PIN number of the used SIM card

Command Syntax	GSM.PIN=<"pin">
Example	$PFAL,GSM.PIN="1111"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to enter the PIN code of the used SIM card. If SIM PIN has already been entered and the target device is already registered into the GSM network, no further entry needed (the device returns error). See also the description in chapter *5.9.1.* more details.

Parameter Description

Parameter	Value	Meaning
<"pin">	It specifies the PIN number of the used SIM card, wrapped in quotation marks. This may be for example the SIM PIN to register onto the GSM network, or the SIM PIN to replace the current PIN number with a new one. **4** to **8** digits are available.	

4.7.1.2. GSM.PUK=<"puk">,<"pin"> - Enters the PUK and PIN numbers

Command Syntax	GSM.PUK=<"puk">,<"pin">
Example	$PFAL,GSM.PUK,"22222222","1111"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to unblock the SIM card by entering the associated PUK code.

Parameter Description

Parameter	Value
<"puk">	Entering incorrect PIN three times, the SIM card will be blocked. To unblock it, you have to enter the PUK code of the used SIM card, wrapped in quotation marks (" "). After ten failed attempts to enter the PUK, the SIM card will be invalidated and no longer operable. In such a case, the card needs to be replaced. PIN consists of 4 to 8 digits; PUK is an 8-digit code only.
<"pin">	It specifies the PIN number of the used SIM card, wrapped in quotation marks (" ").

4.7.1.3. GSM.IMEI – Returns the international mobile equipment identity

Command Syntax	GSM.IMEI
Example	$PFAL,GSM.IMEI

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command is intended to request the International Mobile Station Equipment Identity (IMEI) of the GSM modem that looks more like a serial number which distinctively identifies a mobile station internationally.

Parameter Description

None.

4.7.1.4. GSM.IMSI – Returns the International Mobile Subscriber Identity

Command Syntax	GSM.IMSI
Example	$PFAL,GSM.IMSI

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command is intended to request the International Mobile Subscriber Identity (IMSI) .

Parameter Description

None.

4.7.1.5. GSM.ICCID – Returns the Integrated Circuit Card Identifier of the SIM card

Command Syntax	GSM.ICCID
Example	$PFAL,GSM.ICCID

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command is intended to request the Integrated Circuit Card Identifier (ICCID) of the inserted SIM card. It is a unique 19-digit number printed on the SIM card.

Parameter Description

None.

4.7.1.6. GSM.OwnNumber– Returns the phone number of the SIM card

Command Syntax	GSM.OwnNumber
Example	$PFAL,GSM.OwnNumber

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command is intended to request the phone number of the used SIM card.

Parameter Description

None.

Notes

The phone number of the used SIM card must already be stored into the SIM card, before sending this command to the device, otherwise the device will report error.

4.7.1.7. GSM.Balance– Returns account balance of an used prepaid SIM card

Command Syntax	GSM.Balance
Example	$PFAL,GSM.Balance
Example	$<GSM.USSD> $error in USSD command $ERROR $<end>

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command allows you to obtain the information about the current account balance in pre-paid GSM services. It requests the amount of money and the validity period of your account balance depends on the specific services the operator is offering.

Parameter Description

None.

Notes

- Once the user sends this command to the AVL device, it will automatically dial "ATD*100#" access number which is available only for the E-plus German network operator. Other countries may have other dial numbers for checking the account balance.
- If the command response with an error or garbage, that means that the SIM card does not offer this option.

4.7.1.8. GSM.USSD – Performs an USSD call and return its answer

Command Syntax	GSM.USSD,"<ussd_cmd>",<timeout>
Example	$PFAL,GSM.USSD,"*100#",10

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command allows you to perform an USSD call and return its answer. USSD is a standard for transmitting information over GSM signaling channels. It is mostly used as a method to query the available balance and other similar information in pre-paid GSM services. USSD is network-dependent and depends on the specific services the operator is offering.

Parameter Description

Parameter	Value	Meaning
<ussd_cmd>	Specifies the USSD call command (i.e. *100# to query the SIM account balance).	
<timeout>	Specifies the time in seconds to wait for the USSD answer.	

Notes

USSD command responses will be outputted ONLY in text mode, PDU decoding is not supported.

4.7.1.9. GSM.MCC – Gets the current mobile country code

Command Syntax	GSM.MCC
Example	$PFAL,GSM.MCC

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command allows you to read out the current mobile country code information of the operator where the device is registered to.

Parameter Description

None.

Notes

A valid operator is required to read out the MMC.

4.7.1.10. GSM.Band – Specifies the GSM band used by the device

Command Syntax	GSM.Band=<band>
Example	$PFAL,GSM.Band=Eur

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

GSM operates on multiple frequency bands around the world - 900 MHz and 1800Mhz, used in Europe, Africa, Asia, Australia and South America; 850 MHz and 1900 MHz, used in the North and South America. Different service providers operate on different frequency within the country. For example, T-Mobile and Vodafone operate on GSM 900 MHz in the Germany, while E-Plus and O2 operate on GSM 1800 MHz T-Mobile operates on 1900 MHz in the USA, while Cingular operates on 1900 MHz or 850 MHz.

This command allows you to specify the GSM band used by the device. The main purpose of this command is to speed up GSM registration if the correct GSM band is selected. The specified Band will be stored into non-volatile memory and used whenever the device starts, so this command must be entered just once. However, the device will change its bands automatically, if it cannot find an operator in the currently selected band.

Parameter Description

Parameter	Value	Meaning
<band>	It can be set to a value as follow:	
	Eur	900Mhz+1800Mhz (European GSM band)
	Misc1	900Mhz+1900Mhz (some smaller countries)
	Misc2	850Mhz+1800Mhz (some smaller countries)
	USA	850Mhz+1900Mhz (USA - GSM band)
	AUTO	Selects the GSM band currently available.

Notes

The GSM engine will be restarted when sending this command.

4.7.2. GSM.CMB

GSM Cell Broadcast Message commands and functionality can be found within this chapter.

4.7.2.1. GSM.CBM.Add,<message_slot>,<cbm_id> - Adds a CBM message to the message slot

Command Syntax	GSM.CBM.Add,<message_slot>,<cbm_id>
Example	$PFAL,GSM.CBM.Add,0,50

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command adds a new CBM message to an empty message slot. Whenever a CBM message with the specified ID is received, its data will be stored at this message slot. Dynamic variables *&(CBM0) .. &(CBM4)* can be used to display the last received message contents. Added messages will be stored within device configuration (non-volatile memory) and will be automatically activated whenever the system starts.

Parameter Description

Parameter	Value	Meaning
<message_slot>	Specifies the message slot in which the received message data will be stored. It is a decimal number ranging from **0** to **4**.	
<cbm_id>	Specifies the broadcast message ID. Only data of the specified message ID will be stored within the message slot.	

4.7.2.2. GSM.CBM.Remove,<message_slot> - Removes data from message slot

Command Syntax	GSM.CBM.Remove,<message_slot>
Example	$PFAL,GSM.CBM.Remove,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command removes a previously added message from its message slot. The slot itself can be configured with another message then. This command also erases the corresponding device configuration entry (non-volatile memory), so a previously removed message will not be activated again at next system start or unless an Add command is used.

Parameter Description

Parameter	Value	Meaning
<message_slot>	Specifies the message slot in which the received message data will be stored. It is a decimal number ranging from **0** to **4**.	

4.7.2.3. GSM.CBM.Info - Queries CBM information

Command Syntax	GSM.CBM.Info
Example	$PFAL,GSM.CBM.Info

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Command description

This commands queries information about the used message slots and their current message.

Parameter Description

None.

4.7.3. GSM.VoiceCall

4.7.3.1. GSM.VoiceCall.Dial,<"p_number"> - Makes a GSM voice call

Command Syntax	GSM.VoiceCall.Dial,<"p_number">
Example	$PFAL,GSM.VoiceCall.Dial,"+4912345678"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command is intended to make an outgoing voice call.

Parameter Description

Parameter	Value	Meaning
<"p_number">	Specifies the phone number to be dialed, wrapped in quotation marks (" "). It originates an outgoing voice call to the specified target phone number <*"p_number"*>. It includes the area code, country code and phone number.	

4.7.3.2. GSM.VoiceCall.Accept - Accepts an incoming voice call

Command Syntax	GSM.VoiceCall.Accept
Example	$PFAL,GSM.VoiceCall.Accept

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command is intended to accept an incoming voice call, which is usually indicated by an incoming ring alarm event.

Parameter Description

None.

Notes

No GSM voice calls are accepted while the AVL device is trying to establish a GPRS connection. During this time which takes approx. 10-20 seconds, the device is unreachable (this is GSM related). Therefore it is wise not to perform a GPRS attach/detach in such short periods.

4.7.3.3. GSM.VoiceCall.Hangup – Hangs up an active voice call

Command Syntax	GSM.VoiceCall.Hangup
Example	$PFAL,GSM.VoiceCall.Hangup

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command is intended to cancel an established call.

Parameter Description

None.

4.7.3.4. GSM.VoiceCall.SendDTMF,<duration>,<"dtmf_tones"> - Send DTMF tones while inside a call

Command Syntax	GSM.VoiceCall.SendDTMF,<duration>,<"dtmf_tones">
Example	$PFAL,GSM.Voicecall.SendDTMF,2,"123&(fix)"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✗	✗	✗

Command description

This command sends DTMF tones while the device is inside a voice call.

Parameter Description

Parameter	Value	Meaning
<"p_number">	Specifies the phone number to be dialed, wrapped in quotation marks (" "). It originates an outgoing voice call to the specified target phone number <*"p_number"*>. It includes the area code, country code and phone number.	
<duration>	Specifies the duration of each single DTMF tone in 100 milliseconds.	
<"dtmf_tones">	Defines the user defined text, up to 100 chars, to be sent to. This text may also contain Dynamic variables, which are described in chapter *7:*	

4.7.4. GSM.Audio

This chapter contains all audio related GSM settings. Up to 5 audio profiles can be independently configured, selected and stored to non-volatile memory.

All commands are optional - if no audio profile have been saved/specified, a default audio profile will be loaded as *Profile0* and will be automatically used.

This profile can be viewed using the command ***PFAL,GSM.Audio.ShowProfile=0***

4.7.4.1. GSM.Audio.ActiveProfile - Selects and activates an audio profile

Command Syntax	GSM.Audio.ActiveProfile=<profile> GSM.Audio.ActiveProfile,<profile>
Example	$PFAL,GSM.Audio.ActiveProfile=0 $PFAL,GSM.Audio.ActiveProfile,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✗	✗	✗

Exception: The GSM.Audio feature is suported only by FOX3-3G AU variant.

Command description

This command selects and activates an audio profile.

Parameter Description

Parameter	Value	Meaning
<profile>	It can be set to a value from 0..4.	

Notes

If no audio profiles have been saved (default factory setting), default audio settings will be used after system start

It is impossible to select a profile which has not been saved before (selecting empty profiles is forbidden).

4.7.4.2. GSM.Audio.ShowProfile - Shows details of audio profile

Command Syntax	GSM.Audio.ShowProfile=<profile> GSM.Audio.ShowProfile,<profile>
Example	$PFAL,GSM.Audio.ShowProfile=0 $PFAL,GSM.Audio.ShowProfile,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command shows all details of the specified audio profile. It also shows whether this profile is currently active (used) or inactive.

Parameter Description

Parameter	Value	Meaning
<profile>	It can be set to a value from 0..4.	

Notes

Default audio settings are shown after first startup, if no audio settings have been modified.

4.7.4.3. GSM.Audio.SaveProfileAs - Stores audio settings to a profile

Command Syntax	GSM.Audio.SaveProfileAs=<profile> GSM.Audio.SaveProfileAs,<profile>
Example	$PFAL,GSM.Audio.SaveProfileAs=0 $PFAL,GSM.Audio.SaveProfileAs,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command stores the currently used audio settings to a profile.

Parameter Description

Parameter	Value	Meaning
<profile>	It can be set to a value from **0**..**4**.	

Notes

It is possible to store default audio settings to a profile too (this might be helpful if several profiles have to be used, because it is not possible to switch to an "empty" profile)

This command also changes the currently active profile setting – so if a profile is stored as profile 3, the current active profile will be 3.

4.7.4.4. GSM.Audio.DeleteProfile - Erases a stored profile

Command Syntax	GSM.Audio.DeleteProfile=<profile> GSM.Audio.DeleteProfile,<profile>
Example	$PFAL,GSM.Audio.DeleteProfile=0 $PFAL,GSM.Audio.DeleteProfile,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command erases a stored profile.

Parameter Description

Parameter	Value	Meaning
<profile>	It can be set to a value from **0**..**4**.	

Notes

This command does NOT affect currently active audio settings. It also doesn't change the currently used profile I. It can erase the currently active profile

```
An empty profile is selected in this case. After next system start,
default audio settings would be used if no other valid profile has
been activated.
```

4.7.4.5. GSM.Audio.EchoCancel - Activates or deactivates echo cancellation

Command Syntax	GSM.Audio.EchoCancel=<value>
Example	$PFAL,GSM.Audio.EchoCancel=0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command activates or deactivates echo cancellation for a handsfree speaker/microphone.

Parameter Description

Parameter	Value	Meaning
<value>	It can be set to a value as follow:	
	0	Disables echo canceling.
	1	Enables echo canceling (default)

Notes

- This command does NOT affect the settings of the stored audio profile.
- In order to keep permanently stored the settings of the selected audio, you must store them to an audio profile - else they will be discarded after next system start

4.7.4.6. GSM.Audio.SideTone - Activates audible feedback to speaker

Command Syntax	GSM.Audio.SideTone=<value>
Example	$PFAL,GSM.Audio.SideTone=0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command routes some of the microphone input directly to the speaker output so the user can hear some of his own voice in the speaker.

Parameter Description

Parameter	Value	Meaning
<value>	It can be set to a value as follow:	
	0	Disables side tones (default)
	1	Enables side tones

Notes

- This command does NOT affect stored audio profile settings.
- In order to permanently keep selected audio settings, the settings must be stored to an audio profile – else they are discarded after next system start.

4.7.4.7. GSM.Audio.SpeakerMute - Activates or deactivates speaker output

Command Syntax	GSM.Audio.SpeakerMute=<value>
Example	$PFAL,GSM.Audio.SpeakerMute=0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command activates or deactivates speaker output.

Parameter Description

Parameter	Value	Meaning
<value>	It can be set to a value as follow:	
	0	Unmutes (activates) the speaker (default)
	1	Mutes (deactivates) the speaker

Notes

- This command does NOT affect stored audio profile settings.
- In order to permanently keep selected audio settings, the settings must be stored to an audio profile – else they are erased after system restarts.

4.7.4.8. GSM.Audio.SpeakerGain - Sets speaker gain

Command Syntax	GSM.Audio.SpeakerGain=<value>
Example	$PFAL,GSM.Audio.SpeakerGain=0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command sets speaker gain (loudness).

Parameter Description

Parameter	Value	Meaning
<value>	It can be set to a value as follow:	
	0...14	Loudness of the speaker (maximum might depend on GSM version). Default is 4.

Notes

- This command does NOT affect stored audio profile settings.
- In order to permanently keep selected audio settings, the settings must be stored to an audio profile - else they are discarded after next system start.

4.7.4.9. GSM.Audio.MicrophoneMute - Activates or deactivates microphone

Command Syntax	GSM.Audio.MicrophoneMute=<value>
Example	$PFAL,GSM.Audio.MicrophoneMute=0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command activates or deactivates microphone.

Parameter Description

Parameter	Value	Meaning
<value>	It can be set to a value as follow:	
	0	Unmutes (activates) the microphone (default)
	1	Mutes (deactivates) the microphone.

Notes

- This command does NOT affect stored audio profile settings.
- In order to permanently keep selected audio settings, the settings must be stored to an audio profile – else they are discarded after next system start.

4.7.4.10. GSM.Audio.HandsfreeMicroGain - Sets microphone gain

Command Syntax	GSM.Audio.HandsfreeMicroGain=<value>
Example	$PFAL,GSM.Audio.HandsfreeMicroGain=0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command sets microphone gain (loudness) for handsfree microphone (keep in mind that this microphone line might not be available for older stepp3 devices (hardware revision 2d) . In this case, Handset microphone setting should be used.

Parameter Description

Parameter	Value	Meaning
<value>	It can be set to a value as follow:	
	0...7	loudness of the microphone (maximum might depend on GSM version). Default is 3.

Notes

- This command does NOT affect stored audio profile settings.
- In order to permanently keep selected audio settings, the settings must be stored to an audio profile – else they are discarded after next system start

4.7.4.11. GSM.Audio.HandsetMicroGain - Sets microphone gain

Command Syntax	GSM.Audio.HandsetMicroGain=<value>
Example	$PFAL,GSM.Audio.HandsetMicroGain=0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command sets microphone gain (*loudness*) for handset microphone (keep in mind that this microphone line might not be available for older AVL devices (*hardware revision 2d*). In this case, Handset microphone setting should be used.

Parameter Description

Parameter	Value	Meaning
<value>	It can be set to a value as follow:	
	0...7	loudness of the microphone (maximum might depend on GSM version). Default is **3**.

Notes

- This command does NOT affect stored audio profile settings.
- In order to permanently keep selected audio settings, the settings must be stored to an audio profile – else they are discarded after next system start.

4.7.4.12. GSM.Audio.AudioRingPath - Selects path of ring signals

Command Syntax	GSM.Audio.AudioRingPath=<value>
Example	$PFAL,GSM.Audio.AudioRingPath=0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command selects the path to which ring signals (*i.e. voice input/output*) are directed.

Parameter Description

Parameter	Value	Meaning
<value>	It can be set to a value as follow:	
	0	Automatic path selection (depending on used path)
	1	Handsfree audio path
	2	Handset audio path
	3	Internal (not used) audio path

Notes

- This command does NOT affect stored audio profile settings.
- In order to permanently keep selected audio settings, the settings must be stored to an audio profile – else they are discarded after next system start.

4.7.4.13. GSM.Audio.RingTone - Selects ring tone for incoming calls

Command Syntax	GSM.Audio.RingTone=<value>
Example	$PFAL,GSM.Audio.RingTone=0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command selects the used ring tone for incoming calls.

Parameter Description

Parameter	Value	Meaning
<value>	It can be set to a value as follow:	
	0...18	Sequence 0... 18

Notes

- This command does NOT affect stored audio profile settings.
- In order to permanently keep selected audio settings, the settings must be stored to an audio profile - else they are discarded after next system start.

4.7.4.14. GSM.Audio.RingGain - Sets gain of ring tones

Command Syntax	GSM.Audio.RingGain=<value>
Example	$PFAL,GSM.Audio.RingGain=0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command sets the gain (*loudness*) for ring tones. Note that the gain will be adjusted at the next incoming call. It doesn't affect a current call.

Parameter Description

Parameter	Value	Meaning
<value>	It can be set to a value as follow:	
	0...4	Ringer gain (loudness). Default is **3**.

Notes

- This command does NOT affect stored audio profile settings.
- In order to permanently keep selected audio settings, the settings must be stored to an audio profile – else they are discarded after next system start.

4.7.4.15. GSM.Audio.AudioPath - Selects path for regular audio signals

Command Syntax	GSM.Audio.AudioPath=<value>
Example	$PFAL,GSM.Audio.AudioPath=0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command selects the path for regular audio signals (i.e. Voice).

Parameter Description

Parameter	Value	Meaning
<value>	It can be set to a value as follow:	
	0	Automatic path selection (depending on used path)
	1	Handsfree audio path
	2	Handset audio path
	3	Internal (not used) audio path

Notes

- This command does NOT affect stored audio profile settings.
- In order to permanently keep selected audio settings, the settings must be stored to an audio profile – else they are discarded after next system start.

4.7.4.16. GSM.Audio.SoundMode - Selects global sound mode

Command Syntax	GSM.Audio.SoundMode=<value>
Example	$PFAL,GSM.Audio.SoundMode=0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	×	×	×

Command description

This command selects a global sound mode for the device.

Parameter Description

Parameter	Value	Meaning
<value>	It can be set to a value as follow:	
	0	normal - signals and voice is enabled
	1	silent – just alarm sounds are generated
	2	stealth – no sound is generated (default)

Notes

- This command does NOT affect stored audio profile settings.
- In order to permanently keep the selected audio settings, such settings must be stored to an audio profile – else they are discarded after next restart.

4.7.4.17. GSM.Audio.SMSSignalToneMode =<mode>- Sets the SMS Signal Tone Mode

Command Syntax	GSM.Audio.SMSSignalToneMode=<mode>
Example	$PFAL,GSM.Audio.SMSSignalToneMode=1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✗	✗	✗

Command description

This command is used to enable or disable the signal tones when receiving SMS messages.

Parameter description

Parameter	Value	Meaning
<mode>	Sets the signal tone mode when receiving SMS messages.	
	0	SMS signal tones are disabled.
	1	SMS signal tones are enabled.

Notes

- This command does NOT affect stored audio profile settings.
- In order to permanently keep selected audio settings, the audio settings must be stored to an audio profile – else they are discarded after restarting the device.

4.7.5. GSM.SMS

4.7.5.1. GSM.SMS.Send,<"p_number">,<protocols>,<"text"> - Sends out standard SMS

Command Syntax	GSM.SMS.Send,"<"p_number">",<protocols>,"<"text">" GSM.SMS.Send,"<"p_number">",<protocols>,"<"text">**&(**variable**)**"
Example	$PFAL,GSM.SMS.Send,"+491111111",8,"FOX3" $PFAL,GSM.SMS.Send,"+491111111",0,"on &(Date) at &(Time) it is moving at &(Speed) m/s" $PFAL,GSM.SMS.Send,"&(SMSNumber)",0,"Auto SMS Reply&(SMSNumber)"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to send a SMS message to the specified target phone number <"p_number"> including identification text and defined protocols. The format the device uses to send out the protocols and entered text is configuration-dependent.

Parameter Description

Parameter	Value	Meaning
<"p_number">	It specifies the phone number, max 30 digits, where the alarm (including the specified protocols and identification text) must be sent. The phone number includes the area, country codes and the phone number. It must be wrapped in quotation marks (" "). It can be a short number too. It is also possible to enter a dynamic protocol, for example **&(SMSNumber)** which can be used to setup alarms which reply incoming SMS containing user defined text. The last example in table above, replies an SMS back to the sender of the last received SMS.	
<protocols>	It defines the output NMEA messages, which will be sent to the specified target phone number. It must be specified in the hex format without leading the "0x. Supported protocols are listed in chapter *11.2*.	
<"text">	It specifies the text message, up to 160 characters, which will be sent to the specified target phone number via GSM. The text message may include the user specified text and/or system information. It must be wrapped in quotation marks (" "). If it is required to attach also system information (variable) at certain times the following syntax of the <<"text">is also possible: "text&(<entry$_1$>)text&(<entry$_2$>)text…&(<entry$_n$>)" Each dynamic variable is separated by ampersand "&" without spaces and enclosed in parentheses "( )". For example: $PFAL,GSM.SMS.Send,"+491111111",0,"on &(Date) at &(Time) it is moving at &(Speed) m/s" Dynamic variables are listed in chapter 7.	

Notes

- Longer the <"text"> results less protocols can be attached to the SMS message. Only protocols, which fit completely in are attached to the SMS.
- For longer text use the command **GSM.SMS.SendMulti**

4.7.5.2. GSM.SMS.SendRaw,<*"p_number"*>,<*protocols*>,<*"text"*> - Sends out SMS in raw data

Command Syntax	GSM.SMS.SendRaw,"<"p_number"> ",<protocols>,<"text"> GSM.SMS.SendRaw,<"p_number">,<protocol>,<"text&(entry)">
Example	$PFAL,GSM.SMS.SendRaw,"+491111111",8,"FOX3" $PFAL,GSM.SMS.SendRaw,"+491111111",0,"on &(Date) at &(Time) it is moving at &(Speed) m/s" $PFAL,GSM.SMS.SendRaw,"&(SMSNumber)",0,"Auto SMS Reply&(SMSNumber)"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Sends the specified protocols and/or user text via SMS to the target phone number in raw format. No format characters are used to display the message text (i.e. the text appears exactly as it is – no $ in front or CRLF at the end will be sent).

Parameter Description

Parameter	Value	Meaning
<"p_number">	It specifies the phone number, max 30 digits, where the alarm (including the specified protocols and identification text) must be sent. The phone number includes the area, country codes and the phone number. It must be wrapped in quotation marks (" "). It can be a short number too. It is also possible to enter a dynamic protocol, for example **&(SMSNumber)** which can be used to setup alarms which reply incoming SMS containing user defined text. The last example in table above, replies an SMS back to the sender of the last received SMS.	
<protocols>	It defines the output NMEA messages, which will be sent to the specified target phone number. It must be specified in the hex format without leading the "0x. Supported protocols are listed in chapter *11.2.*	
<"text">	It specifies the text message, up to 160 characters, which will be sent to the specified target phone number via GSM. The text message may include the user specified text and/or system information. It must be wrapped in quotation marks (" "). If it is required to attach also system information (variable) at certain times the following syntax of the <"text">is also possible: "text&(<entry$_1$>)text&(<entry$_2$>)text…&(<entry$_n$>)" Each dynamic variable is separated by ampersand "&" without spaces and enclosed in parentheses "( )". For example: $PFAL,GSM.SMS.Send,"+491111111",0,"on &(Date) at &(Time) it is moving at &(Speed) m/s" Dynamic variables are listed in chapter 7.	

4.7.5.3. GSM.SMS.SendMulti – Send out SMS with more than 160 chars

Command Syntax	GSM.SMS.SendMulti,”<"p_number">”,<protocols>,”<"text">
Example	$PFAL,GSM.SMS.SendMulti,"+491111111",8,"on &(Date) at &(Time) it is moving at &(Speed) m/s"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to send multi SMS messages (more than 160 chars to the specified target phone number <"p_number"> including identification text and defined protocols. The SMS message will then be split in two or more messages on the receiving modem. The format the device uses to send out the protocols and entered text is configuration-dependent.

Parameter Description

Parameter	Value	Meaning
<"p_number">	It specifies the phone number, max 30 digits, where the alarm (including the specified protocols and identification text) must be sent. The phone number includes the area, country codes and the phone number. It must be wrapped in quotation marks (" "). It can be a short number too. It is also possible to enter a dynamic protocol, for example **&(SMSNumber)** which can be used to setup alarms which reply incoming SMS containing user defined text. The last example in table above, replies an SMS back to the sender of the last received SMS.	

Parameter	Value	Meaning
<protocols>	It defines the output NMEA messages, which will be sent to the specified target phone number. It must be specified in the hex format without leading the "0x. Supported protocols are listed in chapter *11.2.*	
<"text">	It specifies the text message, up to 160 characters, which will be sent to the specified target phone number via GSM. The text message may include the user specified text and/or system information. It must be wrapped in quotation marks (" "). If it is required to attach also system information (variable) at certain times the following syntax of the <"text">is also possible: "text&(<entry$_1$>)text&(<entry$_2$>)text…&(<entry$_n$>)" Each dynamic variable is separated by ampersand "&" without spaces and enclosed in parentheses "( )". For example: $PFAL,GSM.SMS.Send,"+491111111",0,"on &(Date) at &(Time) it is moving at &(Speed) m/s" Dynamic variables are listed in chapter *7:*	

4.7.5.4. GSM.SMS.Inbox.Clear – Clears all stored SMS Messages from the SMS memory

Command Syntax	GSM.SMS.Clear
Example	$PFAL,GSM.SMS.Clear

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command clears all incoming SMS messages stored in the SMS memory.

Parameter Description

None.

4.7.5.5. GSM.SMS.Inbox.State – Returns all inbox SMS Messages from the SMS memory

Command Syntax	GSM.SMS.Inbox.State
Example	$PFAL,GSM.SMS.Inbox.State

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command indicates the total number of incoming SMS messages associated with the SMS storage.

Parameter Description

None.

4.7.5.6. GSM.SMS.ClearSIM – Delets all SMS Messages from the SIM memory

Command Syntax	GSM.SMS.ClearSIM
Example	$PFAL,GSM.SMS.ClearSIM

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command deletes all SMS messages from the SIM memory.

Parameter Description

None.

4.7.5.7. GSM.SMS.Outbox.Clear – Clears outgoing SMS from the SMS memory

Command Syntax	GSM.SMS.Outbox.Clear
Example	$PFAL,GSM.SMS.Outbox.Clear

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command deletes all unsent SMS messages from the SMS memory.

Parameter Description

None.

4.7.5.8. GSM.SMS.Outbox.State – Returns all outbox SMS Messages from the SMS memory

Command Syntax	GSM.SMS.Outbox.State
Example	$PFAL,GSM.SMS.Outbox.State

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command indicates the total number of unsent SMS messages associated with the SMS storage.

Parameter Description

None.

4.7.6. GSM.DataCall

4.7.6.1. GSM.DataCall.Send,<protocols>,<"text"> - Sends data via GSM data call

Command Syntax	GSM.DataCall.Send,<protocols>,"<"text">" GSM.DataCall.Send,<protocols>,"<"text">&(entry)">
Example	$PFAL,GSM.DataCall.Send,08,"*text to be received*" $PFAL,GSM.DataCall.Send,0,"on &(Date) at &(Time) it is moving at &(Speed) m/s"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to send messages and/or GPS protocols to the remote modem/phone via an established/initiated data call. The format the device uses to send out the protocols and entered text is configuration-dependent.

Parameter Description

Parameter	Value	Meaning
<protocols>	It defines the output NMEA messages, which will be sent to the specified target phone number. It must be specified in the hex format without leading the "0x. Supported protocols are listed in chapter *11.2.*	
<"text">	It specifies the text message, up to 160 characters, which will be sent to the specified target phone number via GSM. The text message may include the user specified text and/or system information. It must be wrapped in quotation marks (" "). If it is required to attach also system information (variable) at certain times the following syntax of the <"text">is also possible: "text&(<entry$_1$>)text&(<entry$_2$>)text…&(<entry$_n$>)" Each dynamic variable is separated by ampersand "&" without spaces and enclosed in parentheses "( )". For example: $PFAL,GSM.SMS.Send,"+491111111",0,"on &(Date) at &(Time) it is moving at &(Speed) m/s" Dynamic variables are listed in chapter *7:*.	

4.7.6.2. GSM.DataCall.Accept - Accepts incoming GSM data call

Command Syntax	GSM.DataCall.Accept
Example	$PFAL,GSM.DataCall.Accept

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to accept an incoming Data call, which is usually indicated by an incoming ring alarm event.

Parameter Description

None.

Notes

No GSM data calls are accepted while the AVL device is trying to establish a GPRS connection. During this time which takes approx. 10-20 seconds, the device is unreachable (this is GSM related). Therefore, it is wise not to perform a GPRS attach/detach in such short periods.

4.7.6.3. GSM.DataCall.Hangup – Hangs up active data call

Command Syntax	GSM.DataCall.Hangup
Example	$PFAL,GSM.DataCall.Hangup

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is intended to reject an incoming GSM data call, or it finishes an established GSM data call.

Parameter Description

None.

Notes

This feature is currently under development and won't be fully functional for the first release versions.

4.7.7. GSM.GPRS

4.7.7.1. GSM.GPRS.Connect – Performs a GPRS attach

Command Syntax	GSM.GPRS.Connect
Example	$PFAL,GSM.GPRS.Connect

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is used to attach the AVL device to the GPRS service. After the message has been completed and the AVL device is already in the requested state, the message is ignored. GPRS starts only when sufficient signal strength *(8-94)* is found. If the AVL device is not able to attach for more than 5 minutes, no message is returned, but AVL device is still trying to attach.

Parameter description

None.

Notes

If a GPRS attach will be initiated by this message and the AVL device is not able to attach, the

AVL device is started with the default value, due to missing values of the GPRS.APN, GPRS.QOS,GPRS.QOSMIN,PPP.USERNAME and PPP.PASSWORD parameters according to the used SIM card.

- When the AVL device is GPRS attached and a PLMN reselection occurs to a non-GPRS capable network or to a network where the SIM is not subscribed for using GPRS services, the resulting GSM (GPRS mobility management) state according to GSM 24.008 is REGISTERED/NO CELL.
- No GSM calls are accepted while the AVL device is trying to establish a GPRS connection. During this time which takes approx. 10-20 seconds, the device is unreachable (this is GSM related). Therefore it is wise not to perform a GPRS attach/detach in such short periods.

4.7.7.2. GSM.GPRS.Disconnect – Performs a GPRS detach

Command Syntax	GSM.GPRS.Disconnect
Example	$PFAL,GSM.GPRS.Disconnect

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is used to detach the AVL device from the GPRS service. Any active PDP contexts will be automatically deactivated when the attachment state changes to detached. If the AVL device is not able to detach for more than 1 minute, no message is returned, but AVL device is still trying to detach. If an attachment is issued during a running detach this message is ignored.

Parameter description

None.

Notes

- If a GPRS connection is already available and the "Disconnect" command is executed, ensure that the value <value> of the GPRS.AUTOSTART parameter is set to 0 (zero) instead of 1 (one), otherwise the device will try to re-establish automatically that connection.
- The GSM engine will be restarted, if the GPRS connection could not be closed after multiple attempts.

4.7.7.3. GSM.GPRS.State - Returns GPRS state

Command Syntax	GSM.GPRS.State
Example	$PFAL,GSM.GPRS.State

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command is used to request the GPRS and PPP state. A TCP/IP connection can be established only when both states result connected.

Parameter description

None.

4.7.7.4. GSM.GPRS.Traffic[=<complete>,<incoming>,<outgoing>] – Sets/returns GPRS traffic counter

Command Syntax	GSM.GPRS.Traffic GSM.GPRS.Traffic=<complete>,<incoming>,<outgoing>
Example	$PFAL,GSM.GPRS.Traffic $PFAL,GSM.GPRS.Traffic=0,0,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command reads or sets the GPRS traffic counter, the volume, in byte, of GPRS data stream. It reads the GPRS traffic, if the values are omitted.

Parameter description

Parameter	Value	Meaning
<complete>	Number of bytes in range of **0** to **2147483647** for completeness.	
<incoming>	Number of bytes in range of **0** to **2147483647** for incoming data stream.	
<outgoing>	Number of bytes in range of **0** to **2147483647** for outgoing data stream	

Notes

To delete the GPRS traffic counter set all values to “0” zero.

4.7.8. GSM.SetInternal/ExternalAntenna

Both commands below are used to switch manually the device to operate with external or internal antennas.

4.7.8.1. GSM.SetExternalAntenna - Switches to the external antenna

Command Syntax	GSM.SetExternalAntenna
Example	$PFAL,GSM.SetExternalAntenna

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✗

Command description

This command is used to switch manually or automatically with an alarm (AL) the device to the external connected antennas. When the connection of GPS external antenna is detected by the device, the corresponding event occurs "GPS.eExtAntPlugged". With this event you can configure an alarm e.g. (*$PFAL,Cnf.Set,AL98=GPS.eExtAntPlugged:GSM.SetExternalAntenna)* that switch the device automatically to external GSM antenna when the GPS antenna is plugged).

Parameter description

None.

4.7.8.2. GSM.SetInternalAntenna - Switches to the internal antenna

Command Syntax	GSM.SetInternalAntenna
Example	$PFAL,GSM.SetInternalAntenna

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✗

Command description

This command is used to switch manually or automatically with an alarm (AL) the device to the internal connected antennas. When the GPS external antenna is unplugged from device, the corresponding event occurs "GPS.eExtAntUnplugged". With this event you can configure an alarm e.g. (*$PFAL,Cnf.Set,AL98=GPS.eExtAntUnplugged:GSM.SetInternalAntenna)* that switch the device automatically to internal GSM antenna when the GPS antenna is unplugged).

Parameter description

None.

4.8. TCP COMMANDS

In order to have a basic understanding for communication between the AVL device and the remote server, please refer to chapter *2.2.*

4.8.1. TCP. Client

4.8.1.1. TCP.Client.Connect - Performs a TCP connection to the used server

Command Syntax	TCP.Client.Connect
Example	$PFAL,TCP.Client.Connect

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Using this command the AVL device initiates a TCP connection to a remote server. Before the AVL device can send packets using TCP protocol, it needs to know the remote server address and port number it will be sending data to. To assign the address, and the port use the TCP.CLIENT.CONNECT=<s_enable>,<ip_address>,<tcp_port> parameter. The server can decide whether or not to accept the connection. After the TCP connection has been established successfully, use the **TCP.Client.Send,<protocols>,<"text">** command to send the data.

Parameter description

None.

Notes

- If a TCP connection will be initiated by this message and the AVL device is not able to establish, the AVL device is started with the default value, due to missing values of the TCP.CLIENT.CONNECT, parameters according to the used remote server.
- Before using this message, make sure that the AVL device is already GPRS attached, otherwise the AVL device is not able to initiate a TCP connection even if the TCP settings are correctly specified.

4.8.1.2. TCP.Client.Disconnect - Disconnects from the used server

Command Syntax	TCP.Client.Disconnect
Example	$PFAL,TCP.Client.Disconnect

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Using this command the AVL device is able to terminate an existing TCP connection with a remote server. No more data will be sent from the AVL device by the TCP until a new connection to the remote server is initiated and accepted.

Parameter description

None.

Notes

If a TCP connection to the remote server is already available and the Disconnect message is called, ensure that the value <s_enable> from the TCP.CLIENT.CONNECT parameter is set to 0 (zero) instead of 1 (one), otherwise the AVL device will try to reconnect automatically, to the remote server, after each TCP connection failure.

4.8.1.3. TCP.Client.State – Returns TCP connection state

Command Syntax	TCP.Client.State
Example	$PFAL,TCP.Client.State

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command is used to request the state information of the TCP connection. The read command returns the current state of the TCP connection.

Parameter description

None.

4.8.1.4. TCP.Client.Send,<protocols>,<"text"> - Sends a TCP packet to the connected server

Command Syntax	TCP.Client.Send,<protocols>,<"text"> TCP.Client.Send,<protocols>,<"text">&(entry)">
Example	$PFAL,TCP.Client.Send,8,"AVL device sends its GPS positions"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

After a connection has been established, use this command to stream data (GPS positions, user text etc.) to a remote server. Each time when a packet of data has been sent successfully to the remote server, the **TCP.Client.ePacketSent** event occurs. The remote server may evaluate the entered text and use it for further application. The format the device uses to send out the protocols and entered text is configuration-dependent.

Parameter Description

Parameter	Value	Meaning
<protocols>	It defines the output NMEA messages, which will be sent to the specified target phone number. It must be specified in the hex format without leading the "0x. Supported protocols are listed in chapter *11.2.*	
<"text">	Specifies the text message, up to 200 characters, to be sent to the connected remote server. It must be wrapped in quotation marks (" "). If it is required to attach also system information (**entry**) at certain times the following syntax of the <"text"> is also possible. *Dynamic variables are listed in chapter 7:* "`text&(<entry1>)text&(<entry2>)text…&(<entryn>)`" Each dynamic variable is separated by ampersand "&" without spaces and enclosed in parentheses. `For example: $PFAL,TCP.Client.Send,8,"on &(Date) at &(Time) it is moving at &(Speed) m/s"`	

Notes

- If the TCP connection is currently not established, this data will be written into the TCP buffer and it will be sent as soon as the TCP connection is available.
- If the command fails to execute due to the used up buffer, this data will be lost.

4.8.1.5. TCP.Client.ClearSendBuffer - Clears the outgoing TCP buffer

Command Syntax	TCP.Client.ClearSendBuffer
Example	PFAL,TCP.Client.ClearSendBuffer

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Clears all data from the outgoing TCP buffer.

Parameter Description

None.

4.8.1.6. TCP.Client.RxKey= <"key"> - AES KEY incoming packets

Command Syntax	TCP.Client.RxKey=<" Key">
Example	$PFAL,TCP.Client.RxKey="00000000000000000000000000000000"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command setup AES128 key for decrypting incoming packets of a secured TCP transmission. The PREMIUM-FEATURE "AES-TCP" must be enabled to support this function. It is recommended to reset firmware after setting up the key to safely restart TCP connection with the new key.

Parameter Description

Parameter	Value	Meaning
<"Key">	Separated by commas or another "non-hexadecimal" sign, it defines a 16 hexadecimal key for decrypting incoming packets between the device and destination TCP server.	

4.8.1.7. TCP.Client.TxKey= <"key"> - AES KEY outgoing packets

Command Syntax	TCP.Client.TxKey=<"Key">
Example	$PFAL,TCP.Client.TxKey="00000000000000000000000000000000"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command setup AES128 key for encrypting outgoing packets of a secured TCP transmission. The PREMIUM-FEATURE AES-TCP must be enabled to support this function. It is recommended to reset firmware after setting up the key to safely restart TCP connection with the new key.

Parameter Description

Parameter	Value	Meaning
<"Key">	Separated by commas or another "non-hexadecimal" sign, it defines a 16 hexadecimal key for decrypting incoming packets between the device and destination TCP server.	

4.8.1.8. TCP.Client.FlushSendBuffer - Transfers buffered data immediately

Command Syntax	TCP.Client.FlushSendBuffer
Example	$PFAL,TCP.Client.FlushSendBuffer

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command can be used to transfer the data available in the TCP.CLIENT.SEND-MODE=2[,<buffer_level>] before the buffer is filled up.

Parameter Description

None.

4.8.1.9. TCP.Client.SetCertificate – Set certificate used by TLS library

Command Syntax	TCP.Client.SetCertificate
Example	$PFAL,TCP.Client.SetCertificate

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Set certificate used by the TLS library.
The certificate must be sent after the command and the transmission is finished by "<CR><LF>".

Parameter description

The used certificate chain for the TCP channel if using a TLS connection.

4.8.1.10. TCP.Client.ShowCertificate – Shows the used TLS certificate on the main TCP connection

Command Syntax	TCP.Client.ShowCertificate
Example	$PFAL,TCP.Client.ShowCertificate

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Shows the certificate used by the TLS library on the main TCP connection.

Parameter description

None.

4.8.1.11. TCP.Client.ClearCertificate – Clear certificate used by TLS library

Command Syntax	TCP.Client.ClearCertificate
Example	$PFAL,TCP.Client.ClearCertificate

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

No certificate is used by the TLS library. The device will not check the identity of the used server.

Parameter description

None.

4.8.2. TCP.Storage

The TCP Storage is an additional TCP buffer, which enables to collect information before it is sent. This reduces transmission cost and enables to send quickly various information to the connected TCP server.

Currently the TCP storage supports two operation modes:

Manual mode (default)

If this mode is selected, the TCP storage must be dispatched manually (i.e. you can specify when to send stored information via TCP)

Automatic dispatch

This mode allows the system to dispatch data inside storage automatically whenever it is used up.

The operation modes as well as the size of the TCP storage can be accessed via parameter configuration. Please, refer to configuration reference for more details, see chapter *5.12.11.*

Note: *All information you are going to transmit via TCP storage is transmitted exactly in the way you specified it. In order to ease the server based readout process of this data, it is recommended to add additional "identification" characters. Please see further nodes inside the command* ***"AddProtocol"*** *below. Using of such identification characters allows you to distinguish easily between textual* ***"AddProtocol"*** *and binary data (generated by* ***"AddRecord"****).*

4.8.2.1. TCP.Storage.Dispatch - Sends a TCP packet to the connected server

Command Syntax	TCP.Storage.Dispatch
Example	$PFAL,TCP.Storage.Dispatch

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command enqueues currently stored information inside the TCP storage to the outgoing TCP

buffer. This assures a consistent data transfer via TCP. After storage data has been appended to this outgoing buffer, the storage will be cleaned. New data can then be appended.

Parameter description

None.

Notes

The format of the created TCP buffers has been extended. Now it is possible to detect binary storage contents within each message. The format is similar as a **GPS.History.Read** message:

```
Start header $<TCP.Storage.Data><CRLF>
Length info (2 bytes binary information) - value 0x00 – 0xFFFF
This length specifies the amount of storage data bytes contained in this
packet.
Data content (the number of bytes specified with length info)
End header $<end><CRLF>
```

4.8.2.2. TCP.Storage.Clear - Clears TCP storage

Command Syntax	TCP.Storage.Clear
Example	$PFAL,TCP.Storage.Clear

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command clears the contents of the created TCP storage. This command can be used to discard unwanted information and empty the TCP storage completely *(without sending its data away)*.

Parameter description

None.

4.8.2.3. TCP.Storage.AddProtocol,<protocol>,<"text"> - Adds data to the TCP storage

Command Syntax	TCP.Storage.AddProtocol,<protocols>,<"text">
Example	$PFAL,TCP.Storage.AddProtocol,0,"your text"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command writes the specified protocols and/or user text to TCP storage.

Parameter description

Parameter	Value	Meaning
<protocols>	It allows you to request the current protocol data of the specified protocol(s). Supported protocols are listed in chapter *11.2.*	

Parameter	Value	Meaning
<"text">	It specifies the text message, up to 200 characters, to be transmitted to the connected remote server via an available TCP connection. This text may also contain dynamic variables listed in chapter 7:. `The hex value 27 added on the message "$PFAL,TCP.Storage.AddProtocol,27,"test"" means the GGA+GSA+GSV+VTG protocols will be received at once,` `The hex value 4F added on the message "$PFAL, TCP.Storage.AddProtocol,4F,"test" " means the GGA+GSA+GSV+RMC+IOP protocols will be received at once.`	

Notes

In order to receive more than one protocol at once, you have to specify the <protocols> in the hexadecimal value and add the corresponding hex value of required protocols, for example:

You can use dynamic variables within “user text”. Specified protocols are formatted using PFAL send format. This may ease an automatic readout of data via TCP server.

4.8.2.4. TCP.Storage.AddRecord,<protocol>,<"text"> - Appends data to TCP storage

Command Syntax	TCP.Storage.AddRecord,<add_protocols>,<"text">
Example	$PFAL,TCP.Storage.AddRecord,0,"" $PFAL,TCP.Storage.AddRecord,20,"user note: freeway reached "

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command appends a binary data frame to TCP storage. This data frame has the same format as a full record history entry *(including possible extensions)*. The contents to be written are limited by available TCP storage. If there is not enough memory available to satisfy a TCP storage requirement, AVL device will report an error upon attempting to start this process.

Parameter description

Parameter	Value	Meaning
<add_protocols>	It determines whether or not additional information must be recorded in the TCP storage. It specifies the value in hexadecimal format (without "0x"). Following are listed additional information that can be defined in hexadecimal value:	
	0x01	Writes the current state of the Input and Output.
	0x02	Writes the current state of the GSM (field strength, cell id, area code, of incoming/outgoing SMS etc.)
	0x04	Writes the current system operating mode, GPRS, PPP, TCP and system lifetime.
	0x08 and 0x10	Reserved.
	0x20	Writes the specified text from the <"text"> field (up to 99 characters available).
	0x40	Writes the current state of the Geofence areas (inside or outside of a marked area).
	0x80	Reserved.
<"text">	It defines a string value that contains user information. The specified text is limited to 200 characters and it must be wrapped in quotation marks (" "). If no user text must be written, this field can be left empty, except quotation marks (""). The user information will be written if the corresponding hex value (20) has been set in the <add_protocols> field	

Notes

In order to attach more than one additional information at once, specify the sum determined by adding the corresponding hex value of each additional information, for example:

```
The hex value 7 means: IN/OUT +GSM+ system states will be stored together
with current location of the device at once.
```

Writing of this data frame into the TCP storage does NOT require a valid GPS fix. Also invalid data can be added to TCP storage, which stands in contrast to the GPS.History.Write command.

4.8.3. TCP.SMTP

4.8.3.1. TCP.SMTP.Send,<"email_address">,<protocols>,<"text"> - Sends emails

Command Syntax	TCP.SMTP.Send,<"email_address">,<protocols>,<"usertext"> TCP.SMTP.Send, <"email_address">,<protocols>,<"usertext&(entry)"
Example	$PFAL,TCP.SMTP.Send,"test@mailserver.com",8,"FOX3 GPS positions"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command enables sending of E-Mails, including GPS position data and user text, via an Internet mail server. Exactly, it prepares an Email for sending. The user text can be evaluated and used for further application processes. The remote mail server to connect is defined in the settings <mail_server_address> and <mail_server_port> of the ***TCP.SMTP.CONNECT*** parameter. When

an E-Mail message is successfully delivered, the ***TCP.SMTP.eSent*** event occurs. If there is an error during sending of E-Mail, the ***TCP.SMTP.eFailed*** event raises and this email message gets deleted. The format the device uses to send out the text is configuration-dependent.

The subject, the AVL device uses for outgoing E-Mail is:

Subject syntax	Message from:	<hardware name>	rev:<string>	(Name="<name>",IMEI=<GSM.IMEI>)
Subject example	Message from: FOX3 rev:2D-N (Name="my FOX3 device", IMEI=123456789012345)			

Parameter Description

Parameter	Value	Meaning
<"email_address">	Specifies the e-mail address of the message recipient. Only one E-Mail address is permitted per message. The E-Mail address must be wrapped in quotation marks (" ").	
<protocols>	Defines the NMEA messages to be sent to the recipient of the E-Mail message. It must be specified in the hex format without leading the "0x". Supported protocols are listed in chapter *11.2.*	
<"usertext">	Specifies the text message, up to 200 characters, to be sent to the recipient of the E-Mail message. It must be wrapped in quotation marks (" "). If it is required to attach also system information (**entry**) at certain times the following syntax of the <"text"> is also possible. *Dynamic variables are listed in chapter 7:.* "text&(<entry$_1$>)text&(<entry$_2$>)text…&(<entry$_n$>)" Each dynamic variable is separated by ampersand "&" without spaces and enclosed in parentheses "( )". `Example:` `$PFAL,TCP.SMTP.Send,"test@mailserver.com",8,"on &(Date) at &(Time) Car is moving at &(Speed) m/s"` `To view the location of the device in the Google Earth, just enter the following web link in the usertext:` `$PFAL,TCP.SMTP.Send,"test@mailserver.com",8,"http://maps.google.com/maps?f=q&hl=en&q=&(lat)+&(lon)"` `When you receive this e-mail, copy and paste it in the address field of your browser.`	

Notes

- If there is no GSM operator and no active GPRS/TCP connection currently available, the device stores the email data into the outbox and when the GPRS connection is available again it sends this data.
- If there is an error during sending an email message, the data of this email message is lost.
- Using only SMTP service, the configuration of **"GPRS.AUTOSTART=1"** is not required. It can be set to **"GPRS.AUTOSTART=0".**
- Do not try to send another email without getting response from the prior one. After an email has been delivered successfully, a new mail can be sent with this command.

4.8.3.2. TCP.SMTP.SendRaw,<"email_address">,<protocols>,<"text"> - Sends emails in raw format

Command Syntax	TCP.SMTP.Send,<"email_address">,<protocols>,<"usertext"> TCP.SMTP.Send, <"email_address">,<protocols>,<"usertext&(entry)">
Example	$PFAL,TCP.SMTP.Send,"test@mailserver.com",8,"FOX3 sends its GPS positions"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command enables sending of E-Mails, including GPS position data and user text, via an Internet mail server. Exactly, it prepares an Email for sending. The user text can be evaluated and used for further application processes. The remote mail server to connect is defined in the settings <mail_server_address> and <mail_server_port> of the ***TCP.SMTP.CONNECT*** parameter. When an E-Mail message is successfully delivered, the ***TCP.SMTP.eSent*** event occurs. If there is an error during sending of E-Mail, the ***TCP.SMTP.eFailed*** event raises and this email message gets deleted. The format the device uses to send out the text is configuration-dependent.

The subject the AVL device uses for outgoing E-Mail is:

Subject syntax	Message from:	<hardware name>	rev:<string>	(Name="<name>",IMEI=<GSM.IMEI>)
Subject example	Message from: AVL device rev:2D-N (Name="my FOX3device", IMEI=123456789012345)			

Parameter Description

Parameter	Value	Meaning
<"email_address">	Specifies the e-mail address of the message recipient. Only one E-Mail address is permitted per message. The E-Mail address must be wrapped in quotation marks (" ").	
<protocols>	Defines the NMEA messages to be sent to the recipient of the E-Mail message. It must be specified in the hex format without leading the "0x". Supported protocols are listed in chapter *11.2.*	

Parameter	Value	Meaning
<"usertext">	Specifies the text message, up to 200 characters, to be sent to the recipient of the E-Mail message. It must be wrapped in quotation marks (" "). If it is required to attach also system information (**entry**) at certain times the following syntax of the <"text"> is also possible. *Dynamic variables are listed in chapter 7:.* "text&(<entry$_1$>)text&(<entry$_2$>)text…&(<entry$_n$>)" Each dynamic variable is separated by ampersand "&" without spaces and enclosed in parentheses "( )". Example: $PFAL,TCP.SMTP.Send,"test@mailserver.com",8, "on &(Date) at &(Time) Car is moving at &(Speed) m/s" To view the location of the device in the Google Earth, just enter the following web link in the usertext: $PFAL,TCP.SMTP.Send,"test@mailserver.com",8, "http://maps.google.com/ maps?f=q&hl=en&q=&(lat)+&(lon)" When you receive this e-mail, copy and paste it in the address field of your browser.	

Notes

- If there is no GSM operator and no active GPRS/TCP connection currently available, the device stores the email data into the outbox and when the GPRS connection is available again it sends this data.
- If there is an error during sending an E-Mail message, the data of this email message is lost.
- Using only SMTP service, the configuration of **"GPRS.AUTOSTART=1"** is not required. It can be set to **"GPRS.AUTOSTART=0".**
- Do not try to send another email without getting response from the prior one. After an email has been delivered successfully, a new mail can be sent with this command.

4.8.4. TCP.Client2

Second TCP channel with the same performance as the primary TCP channel.

4.8.4.1. TCP.Client2.SetCertificate – Set certificate on the second TCP connection

Command Syntax	TCP.Client2.SetCertificate
Example	$PFAL,TCP.Client2.SetCertificate

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Set certificate used by the TLS library on the second TCP connection.
The certificate must be sent after the command and the transmission is finished by “<CR><LF>”.

Parameter description

The used certificate chain for the TCP channel if using a TLS connection.

4.8.4.2. TCP.Client2.ShowCertificate - Shows the used TCP certificate on the second TCP connection

Command Syntax	TCP.Client2.ShowCertificate
Example	$PFAL,TCP.Client2.ShowCertificate

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Shows the certificate used by the TLS library on the main TCP connection.

Parameter description

None.

4.8.4.3. TCP.Client2.ClearCertificate - Clear certificate used by TLS library

Command Syntax	TCP.Client2.ClearCertificate
Example	$PFAL,TCP.Client2.ClearCertificate

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

No certificate is used by the TLS library. The device will not check the identity of the used server.

Parameter description

None.

4.8.4.4. TCP.Client2.Connect – Opens a second TCP connection

Command Syntax	TCP.Client2.Connect
Example	$PFAL,TCP.Client2.Connect

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Using this command the AVL device initiates a TCP connection to a remote server. Before the AVL device can send packets using TCP protocol, it needs to know the remote server address and port number it will be sending data to. To assign the address, and the port use the TCP.CLIENT2.CONNECT=<>,<>,<> parameter. The server can decide whether or not to accept the connection. After the TCP connection has been established successfully, use the **TCP.Client2.Send,<protocols>,<"text">** command to send the data.

Parameter description

None.

Notes

- If a TCP connection will be initiated by this message and the AVL device is not able to establish, the AVL device is started with the default value, due to missing values of the TCP.CLIENT2.CONNECT, parameters according to the used remote server.
- Before using this message, make sure that the AVL device is already GPRS attached, otherwise the AVL device is not able to initiate a TCP connection even if the TCP settings are correctly specified.

4.8.4.5. TCP.Client2.Disconnect – Closes the second TCP connection

Command Syntax	TCP.Client2.Disconnect
Example	$PFAL,TCP.Client2.Disconnect

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Using this command the AVL device is able to terminate an existing TCP connection with a remote server. No more data will be sent from the AVL device by the TCP until a new connection to the remote server is initiated and accepted.

Parameter description

None.

Notes

If a TCP connection to the remote server is already available and the Disconnect message is called, ensure that the value <s_enable> from the TCP.CLIENT2.CONNECT parameter is set to 0 (zero) instead of 1 (one), otherwise the AVL device will try to reconnect automatically, to the remote server, after each TCP connection failure.

4.8.4.6. TCP.Client2.State – Shows the state of the second TCP connection

Command Syntax	TCP.Client2.State
Example	$PFAL,TCP.Client2.State

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command is used to request the state information of the second TCP connection. The read command returns the current state of the second TCP connection.

Parameter description

None.

4.8.4.7. TCP.Client2.Send,<protocols>,<"text"> - Sends a TCP packet from the second TCP connection

Command Syntax	TCP.Client2.Send,<protocols>,<"text"> TCP.Client2.Send,<protocols>,<"text&(entry)"
Example	$PFAL,TCP.Client2.Send,8,"AVL device sends its GPS positions"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

After a connection has been established, use this command to stream data (GPS positions, user text etc.) to a remote server. Each time when a packet of data has been sent successfully to the remote server, the **TCP.Client.ePacketSent** event occurs. The remote server may evaluate the entered text and use it for further application. The format the device uses to send out the protocols and entered text is configuration-dependent.

Parameter Description

Parameter	Value	Meaning
<protocols>	Defines the NMEA messages to be sent to the recipient of the E-Mail message. It must be specified in the hex format without leading the "0x". Supported protocols are listed in chapter *11.2.*	
<"usertext">	Specifies the text message, up to 200 characters, to be sent to the recipient of the E-Mail message. It must be wrapped in quotation marks (" "). If it is required to attach also system information (**entry**) at certain times the following syntax of the <"text"> is also possible. *Dynamic variables are listed in chapter 7:.* `"text&(<entry1>)text&(<entry2>)text…&(<entry n>)"` Each dynamic variable is separated by ampersand "&" without spaces and enclosed in parentheses "( )". `Example:` `$PFAL,TCP.SMTP.Send,"test@mailserver.com",8,` `"on &(Date) at &(Time) Car is moving at` `&(Speed) m/s"`	

Notes

- If the TCP connection is currently not established, this data will be written into the TCP buffer and it will be sent as soon as the TCP connection is available.
- If the command fails to execute due to the used up buffer, this data will be lost.

4.8.4.8. TCP.Client2.ClearSendBuffer – Clears the outgoing TCP buffer from the second TCP connection

Command Syntax	TCP.Client2.ClearSendBuffer
Example	$PFAL,TCP.Client2.ClearSendBuffer

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Clears all data from the outgoing TCP buffer.

Parameter Description

None.

4.8.4.9. TCP.Client2.FlushSendBuffer – Flushes the outgoing TCP buffer for the second TCP connection

Command Syntax	TCP.Client2.FlushSendBuffer
Example	$PFAL,TCP.Client2.FlushSendBuffer

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command can be used to transfer the data available in the TCP.CLIENT.SEND-MODE=2[,<buffer_level>] before the buffer is filled up.

Parameter Description

None.

4.8.4.10. TCP.Client2.TxKey=<"key"> - AES encrypts the outgoing TCP packet on the second TCP connection

Command Syntax	TCP.Client2.TxKey=<"Key">
Example	$PFAL,TCP.Client2.TxKey="00000000000000000000000000000000"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command sets up AES128 key for encrypting outgoing packets of a secured TCP transmission. The PREMIUM-FEATURE AES-TCP must be enabled to support this function. It is recommended to reset firmware after setting up the key to safely restart TCP connection with the new key.

Parameter Description

Parameter	Value	Meaning
<"Key">	Separated by commas or another "non-hexadecimal" sign, it defines a 16 hexadecimal key for encrypting outgoing packets between the device and destination TCP server.	

4.8.4.11. TCP.Client2.RxKey=<"key"> - AES decrypts the incoming TCP packet on the second TCP connection

Command Syntax	TCP.Client2.RxKey=<"Key">
Example	$PFAL,TCP.Client2.RxKey="00000000000000000000000000000000"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command setup AES128 key for decrypting incoming packets of a secured TCP transmission. The PREMIUM-FEATURE "AES-TCP" must be enabled to support this function. It is recommended to reset firmware after setting up the key to safely restart TCP connection with the new key.

Parameter Description

Parameter	Value	Meaning
<"Key">	Separated by commas or another "non-hexadecimal" sign, it defines a 16 hexadecimal key for encrypting outgoing packets between the device and destination TCP server.	

4.8.5. TCP.MQTT

4.8.5.1. TCP.MQTT.Connect - Performs a TCP connection to a MQTT server / broker

Command Syntax	TCP.MQTT.Connect
Example	$PFAL,TCP.MQTT.Connect

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Using this command the AVL device initiates a TCP connection to a MQTT server / broker.

Before the AVL device can send messages, it needs to know the MQTT server / broker address and port number it will be sending data to. To assign the address, and the port.

Use the MQTT.CLIENT.CONNECT=<s_enable>,<ip_address>,<tcp_port> parameter. In addition, the certificate used and the client's private key must be installed on the device.

Based on the certificates and the key, the server can decide whether the connection should be accepted or not. After the MQTT connection has been established successfully, use the TCP.MQTT.Send,"<topic>@<message>" command to send messages to the broker.

Parameter Description

None.

Notes

Before using this command, make sure that the AVL device is already GPRS attached, otherwise the AVL device is not able to initiate a MQTT connection even if the TCP settings are correctly specified.

4.8.5.2. TCP.MQTT.Disconnect - Disconnects from the used MQTT server / broker

Command Syntax	TCP.MQTT.Disconnect
Example	$PFAL,TCP.MQTT.Disconnect

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Using this command the AVL device is able to terminate an existing MQTT connection with a server / broker. No more data will be sent from the AVL device to the connection until a new connection to the server / broker is initiated and accepted.

Parameter Description

None.

Notes

If a TCP connection to the remote server is already available and the disconnect message is called, ensure that the value <s_enable> from the MQTT.CLIENT.CONNECT parameter is set to 0 (zero) instead of 1 (one). Otherwise the AVL device will try to reconnect automatically, to the server / broker, after each MQTT connection failure or manually disconnection.

4.8.5.3. TCP.MQTT.State - Returns MQTT connection state

Command Syntax	TCP.MQTT.State
Example	$PFAL,TCP.MQTT.State

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command is used to request the state information of the MQTT connection. The read command returns the current state of the MQTT connection.

Parameter Description

None.

4.8.5.4. TCP.MQTT.Send,"<topic>@<message>" - Sends a message to the connected MQTT server / broker

Command Syntax	TCP.MQTT.Send,"<topic>@<message>"
Example	$PFAL,TCP.MQTT.Send,"$aws/things/&(ThingID)/shadow/update@{"state":{"reported":{"temperature":"\&(counter0*0.1)"}}}"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

After a connection has been established, use this command to send messages to a MQTT server/broker. Each time when a packet of data has been sent successfully to the server/broker, the TCP.MQTT.ePacketSent event occurs. The server/broker may evaluate the entered text and use it to distribute to all connected clients. The format the device uses to send out the message text is freely configurable and application dependent. You can find out more about technologies for the AWS IOT cloud, for example at https://docs.aws.amazon.com/iot/latest/developerguide/iot-device-shadows.html .

Parameter Description

Parameter	Value	Meaning
<"topic">	Describes the topic to be updated. The &(ThingID) is available as a special variable. This allows the ThingID of the device to be accessed.	
<"message">	A JSON formatted string with the values to be updated.	

Notes

If the MQTT connection is currently not established, this data will be written into the outgoing MQTT buffer and it will be sent as soon as the MQTT connection is available.

4.8.5.5. TCP.MQTT.Clear - Clears the outgoing MQTT buffer

Command Syntax	TCP.MQTT.Clear
Example	$PFAL,TCP.MQTT.Clear

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Clears all data from the outgoing MQTT buffer.

Parameter Description

None.

4.8.5.6. TCP.MQTT.SetRootCA - Set the server certificate for the MQTT connection

Command Syntax	TCP.MQTT.SetRootCA
Example	$PFAL,TCP.MQTT.SetRootCA

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Set the used server certificate used by the TLS library.
The certificate must be sent after the command and the transmission is finished by ".<CR><LF>”.

Parameter Description

The used certificate for the MQTT connection.

4.8.5.7. TCP.MQTT.GetRootCA - Shows the server certificate for the MQTT connection

Command Syntax	TCP.MQTT.GetRootCA
Example	$PFAL,TCP.MQTT.GetRootCA

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Shows the used server certificate for the MQTT connection.

Parameter Description

None.

4.8.5.8. TCP.MQTT.SetCertificate - Set the client certificate for the MQTT connection

Command Syntax	TCP.MQTT.SetCertificate
Example	$PFAL,TCP.MQTT.SetCertificate

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Set the used client certificate for the MQTT connection.
The client certificate must be sent after the command and the transmission is finished by ".<CR><LF>"

Parameter Description

The used client certificate for the MQTT connection.

4.8.5.9. TCP.MQTT.GetCertificate - Shows the client certificate for the MQTT connection

Command Syntax	TCP.MQTT.GetCertificate
Example	$PFAL,TCP.MQTT.GetCertificate

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Set the used client certificate for the MQTT connection.

The client certificate must be sent after the command and the transmission is finished by ".<CR><LF>"

Parameter Description

None.

4.8.5.10. TCP.MQTT.SetPrivateKey - Set the client private key for the MQTT connection

Command Syntax	TCP.MQTT.SetPrivateKey
Example	$PFAL,TCP.MQTT.SetPrivateKey

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Set the used client private key for the MQTT connection.

The client private key must be sent after the command and the transmission is finished by ".<CR><LF>".

Parameter Description

The used client private key for the MQTT connection.

4.8.5.11. TCP.MQTT.GetPrivateKey - Shows the client private key for the MQTT connection

Command Syntax	TCP.MQTT.GetPrivateKey
Example	$PFAL,TCP.MQTT.GetPrivateKey

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Shows the used client private key for the MQTT connection.

Parameter Description

None.

4.8.5.12. TCP.MQTT.ClearCertificate - Clears MQTT certificates

Command Syntax	TCP.MQTT.ClearCertificate
Example	$PFAL,TCP.MQTT.ClearCertificate

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command Description

Clear all the certificates used by the TLS library. The device no longer has any certificates and keys installed for the communication with the MQTT server.

Parameter Description

None.

4.9. WLAN

Please refer to the chapter *4.2.17.1.* for more details about the WLAN.

4.9.1. WLAN.Scan - Scan for new WLAN networks

Command Syntax	WLAN.Scan
Example	$PFAL,WLAN.Scan

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command is used to start scanning for new WLAN networks in your near environment. After the scan is finished, the WLAN module will be configured for the first network that matches a network in the "known networks list". After successful configuration for a network, the module will try to connect automatically. With dhcp, no static.

Parameter description

None.

Notes

- If the connection cannot be established (authentication failed or network is lost while connecting), the device will enter the unconnected state and a new scan can be started.
- This command returns an error if WLAN is disabled, already connected or the module is not ready.

4.9.2. WLAN.Connect – Connect to a WLAN network profile

Command Syntax	WLAN.Connect,<index>
Example	$PFAL,WLAN.Connect,1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command is used to connect to the wireless access point even when it is not broadcasting its SSID.

Parameter description

Parameter	Value	Meaning
<index>	Specify an already pre-configured WLAN profile. The <index> ranges from 0 to 4	

Notes

- If the connection cannot be established (authentication failed or network is lost while connecting) or if this access point is not in the range, the device will continuously try to connect.
- This command returns an error if WLAN is disabled, already connected or the module is not ready.
- Do not use the command WLAN.Scan to scan for new WLAN networks when using WLAN.Connect,<index>.

4.9.3. WLAN.Send,<protocol>,<"text"> – Sends protocols & user text via WLAN to server

Command Syntax	WLAN.Send,<protocols>>,<"text">
Example	$PFAL,WLAN.Send,8,"GPS" $PFAL,WLAN.Send,0,"DT=&(Date)&(Time); SP=&(Speed) m/s"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command sends the defined protocols and/or user text to the connected server via WLAN. The device should already be connected to the WLAN network, otherwise this command returns an error.

Parameter Description

Parameter	Value	Meaning
<protocols>	See protocol definition in chapter *11.2.*	
<"text">	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (**entry**), which are described in chapter *7:*.	

Notes

If WLAN is disabled or not connected (WLAN.TCP.sConnected=false) this command returns an error.

4.9.4. WLAN.Client.Disconnect – Disconnects from a TCP server

Command Syntax	WLAN.Client.Disconnect
Example	$PFAL,WLAN.Client.Disconnect

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command allows the IOBOX-WLAN to disconnect from the server. An auto reconnect is done by IOBOX-WLAN as soon as possible.

Parameter Description

None.

4.9.5. WLAN.Disconnect – Disconnects from WLAN network

Command Syntax	WLAN.Disconnect
Example	$PFAL,WLAN.Disconnect

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command allows the IOBOX-WLAN to disconnect from a connected WLAN network. It will automatically try to reconnect after 60 seconds.

Parameter Description

None.

4.9.6. WLAN.MAC – Shows the MAC Address of the WIFI module

Command Syntax	WLAN.MAC
Example	$PFAL,WLAN.MAC

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

Command description

This command is used to request the MAC address of the WIFI module. The read command returns the MAC address of the WIFI module.

Parameter Description

None.

4.10. MSG

This chapter contains commands which request or send information as well as commands to change current message processing.

4.10.1. MSG.Send

The commands in this group are used to send dynamic variables* (plus additional protocols if desired) to the specified message output (serial, CSD or TCP). CSD and TCP send commands can be found inside the corresponding command types too but are additionally grouped inside "**MSG**".

*** Dynamic variables**

This feature allows to create user defined messages, which may contain textual information plus system information. For example, a message containing the current date looks like this: “This message was generated at 20.12.2004".

In order to place system information inside user text, special entries can be used which work as system variables.

For a complete set of available dynamic variables, please refer to chapter 7:.

Additionally, it is possible to append predefined protocols to user messages.

The following list consists of all available protocols, which can be attached to messages <protocols> is a hexadecimal value (without leading 0x).

Notes

- Protocol numbers can be added if several have to be sent via a single message. i.e. to send GPIOP and GPGSM, the corresponding number would be C0.
- All sent PFAL-Commands used as alarm action will be executed until they succeed. (i.e. an alarm containing a **CSD.Send** command will attempt to send its information until a CSD connection is established and it can be successfully sent). So special care must be taken to assure that a connection is established before executing the specific send command. Please refer to the alarm examples documentation chapter "things to consider".

4.10.1.1. MSG.Send.Serial<port>,<protocols>,<"text"> - Sends protocols & user text to the serial port

Command Syntax	MSG.Send.Serial<port>,<protocols>,<"text"> MSG.Send.Serial<port>,<protocols>,<"text">&(entry)">
Example	$PFAL,MSG.Send.Serial0,8,"GPS positions" $PFAL,MSG.Send.Serial1,0,"&(Speed)"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✗

Command description

Sends the specified protocols and/or user text to serial output.

Parameter Description

Parameter	Value	Meaning
<port>	It specifies the serial port number to be used for transmitting data from the device. Each of them can be used to transmit the data given in <protocols> and <"text"> entry. FOX3/-3G/-4G series supports 2 serial ports, while Lite models and BOLERO40 series support just one serial port and this is the first serial port 0. This parameter can be set to:	
	0	Serial port 0
	1[1]	Serial port 1
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (entry), which are described in chapter *7:*.	

4.10.1.2. MSG.Send.HexSerial<port>,<protocols>,<"text"> - Outputs data in hex to serial port

Command Syntax	MSG.Send.HexSerial<port>,<protocols>,<"text"> MSG.Send.HexSerial<port>,<protocols>,<"text**&**(entry)">
Example	$PFAL,MSG.Send.HexSerial0,8,"FF,AB,CD,3,44" $PFAL,MSG.Send.HexSerial1,0,"&(Speed)"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✗

Command description

Sends back the specified protocols and/or text in hex format to serial port.

Parameter Description

Parameter	Value	Meaning
<port>	It specifies the serial port number to be used for transmitting data from the device. Each of them can be used to transmit the data given in <protocols> and <"text"> entry. FOX3/-3G/-4G series supports 2 serial ports, while Lite models and BOLERO40 series support just one serial port and this is the first serial port 0. This parameter can be set to:	
	0	Serial port 0
	1[1]	Serial port 1
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (entry), which are described in chapter *7:*.	

4.10.1.3. MSG.Send.RawSerial<port>,<protocols>,<"text"> - Outputs raw data to serial port

Command Syntax	MSG.Send.RawSerial<port>,<protocols>,<"text"> MSG.Send.RawSerial<port>,<protocols>,<"text&(entry)">
Example	$PFAL,MSG.Send.RawSerial0,8,"GPS positions" $PFAL,MSG.Send.RawSerial1,0,"&(Speed)" // Reports the current speed in m/s $PFAL,MSG.Send.RawSerial0,0,"Usertext&(bin=0x0D)" // Adds a Carriage Return at the end of the usertext $PFAL,MSG.Send.RawSerial0,0,"Position of device\x0D\x0A" // Adds a Carriage Return and Line Feed at the end of the usertext $PFAL,MSG.Send.RawSerial0,0,"User&(bin=0x0A)text" //Adds a Line Feed between user and text $PFAL,MSG.Send.RawSerial0,0,"&(bin=65,0x42,0x43,0x44,69,70)" //Reports ABCDEF

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✗

Command description

Sends the specified protocols and/or user text to serial output. No format characters are used to display the message text *(i.e. the text appears exactly as it is – no $ in front or CRLF at the end will be sent)*.

Parameter Description

Parameter	Value	Meaning
<port>	It specifies the serial port number to be used for transmitting data from the device. Each of them can be used to transmit the data given in <protocols> and <"text"> entry. FOX3/-3G/-4G series supports 2 serial ports, while Lite models and BOLERO40 series support just one serial port and this is the first serial port 0. This parameter can be set to:	
	0	Serial port 0
	1[1]	Serial port 1
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (entry), which are described in chapter 7:. It is possible to add Carriage Return and Line Feed inside or at the end of the text using \x0D\x0A (see example above).	

4.10.1.4. MSG.Send.USB,<protocols>,<"text"> - Sends data to USB port

Command Syntax	MSG.Send.USB,<protocols>,<"text"> MSG.Send.USB,<protocols>,<"text&(entry)">
Example	$PFAL,MSG.Send.USB,8,"GPS positions" $PFAL,MSG.Send.USB,0,"&(Speed)"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✗

Command description

This command sends the specified protocols and/or user text to the USB port.

Parameter Description

Parameter	Value	Meaning
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (entry), which are described in chapter *7:*.	

4.10.1.5. MSG.Send.RawUSB,<protocols>,<"text"> - Sends raw data to USB port

Command Syntax	MSG.Send.RawUSB,<protocols>,<"text"> MSG.Send.RawUSB,<protocols>,<"text&(entry)">
Example	$PFAL,MSG.Send.RawUSB,8,"GPS positions" $PFAL,MSG.Send.RawUSB,0,"&(Speed)"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✗

Command description

This command sends the specified protocols and/or user text to the USB port. No format characters are used to display the message text *(i.e. the text appears exactly as it is – no $ in front or CRLF at the end will be sent)*.

Parameter Description

Parameter	Value	Meaning
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (entry), which are described in chapter *7:*.	

4.10.1.6. MSG.Send.FlashBuffer,<protocols>,<"text"> - Stores data to TcpFlashBuffer

Command Syntax	MSG.Send.FlashBuffer,<protocols>,<"text"> MSG.Send.FlashBuffer,<protocols>,<"text&(entry)">
Example	$PFAL,MSG.Send.FlashBuffer,8,"GPS positions" $PFAL,MSG.Send.FlashBuffer,0,"&(Speed)"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✗

Command description

This command stores the specified protocols and/or user text into the nonvolatile TcpFlashBuffer of TCP client.

Parameter Description

Parameter	Value	Meaning
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (entry), which are described in chapter *7:*.	

Notes

Data is stored inside non-volatile storage regardless of the used TCP sendmode. However the use of the Sendmode 2 is strongly recommended, as this mode only allows automatic sendout of the flash buffer contents.

4.10.1.7. MSG.Send.RawFlashBuffer,<protocols>,<"text"> - Stores raw data to TcpFlashBuffer

Command Syntax	MSG.Send.RawFlashBuffer,<protocols>,<"text"> MSG.Send.RawFlashBuffer,<protocols>,<"text&(entry)">
Example	$PFAL,MSG.Send.RawFlashBuffer,8,"GPS positions" $PFAL,MSG.Send.RawFlashBuffer,0,"&(Speed)"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command sends the specified protocols and/or user text into the nonvolatile TcpFlashBuffer of TCP client. No format characters are used to store the message text *(i.e. the text is stored exactly as it is – no $ in front or CRLF at the end will be added)*.

Parameter Description

Parameter	Value	Meaning
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (**entry**), which are described in chapter *7:*	

Notes

Data is stored inside non-volatile storage regardless of the used TCP sendmode. However the use of the Sendmode 2 is strongly recommended, as this mode only allows automatic sendout of the flash buffer contents.

4.10.1.8. MSG.Send.CSD,<protocols>,<"text"> - Sends data via data call

Command Syntax	MSG.Send.CSD,<protocols>,<"text"> MSG.Send.CSD,<protocols>,<"text&(entry)">
Example	$PFAL,MSG.Send.CSD,8,"AVL device outputs its GPS positions" $PFAL,MSG.Send.CSD,0,"on &(Date) at &(Time) it is moving at &(Speed) m/s"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Sends the specified protocols and/or user text to the remote modem during an established data call connection.

Parameter Description

Parameter	Value	Meaning
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (**entry**), which are described in chapter *7:*	

4.10.1.9. MSG.Send.TCP,<protocols>,<"text"> - Sends data via TCP to 1st server

Command Syntax	MSG.Send.TCP,<protocols>,<"text"> MSG.Send.TCP,<protocols>,<"text&(entry)">
Example	$PFAL,MSG.Send.TCP,8,"AVL device outputs its GPS positions" $PFAL,MSG.Send.TCP,0,"on &(Date) at &(Time) it is moving at &(Speed) m/s"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Sends the specified protocols and/or user text to the configured server during an established TCP connection.

Parameter Description

Parameter	Value	Meaning
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (**entry**), which are described in chapter *7:*	

Notes

If the TCP connection is currently not established, this data will be written into the TCP history buf-

fer and it will be sent as soon as the TCP connection will be re-established.

4.10.1.10. MSG.Send.RawTCP,<protocols>,<"text"> - Sends raw data via TCP to 1st server

Command Syntax	MSG.Send.RawTCP,<protocols>,<"text"> MSG.Send.RawTCP,<protocols>,<"text&(entry)">
Example	$PFAL,MSG.Send.RawTCP,8,"AVL device outputs its GPS positions" $PFAL,MSG.Send.RawTCP,0,"on &(Date) at &(Time) it is moving at &(Speed) m/s"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Sends the specified protocols and/or user text in raw format to the configured server during an established TCP connection.

Parameter Description

Parameter	Value	Meaning
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (**entry**), which are described in chapter *7:*	

Notes

If the TCP connection is currently not established, this raw data will be written into the TCP history buffer and it will be sent as soon as the TCP connection will be re-established.

4.10.1.11. MSG.Send.Service,<protocols>,<"text"> - Sends data via TCP to second (service) server

Command Syntax	MSG.Send.Service,<protocols>,<"text"> MSG.Send.Service,<protocols>,<"text&(entry)">
Example	$PFAL,MSG.Send.Service,8,"AVL device outputs its GPS positions" $PFAL,MSG.Send.Service,0,"on &(Date) at &(Time) it is moving at &(Speed) m/s"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Sends the specified protocols and/or user text to the configured second (service) server during an established TCP connection.

Parameter Description

Parameter	Value	Meaning
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (**entry**), which are described in chapter *7:*.	

Notes

If the TCP connection is currently not established, this data will be written into the TCP history buffer and it will be sent as soon as the TCP connection will be re-established.

4.10.1.12. MSG.Send.TCPBuffer,<protocols>,<"text"> - Sends data immediately to the TCP server

Command Syntax	MSG.Send.TCPBuffer,<protocols>,<"text"> MSG.Send.TCPBuffer,<protocols>,<"text&(entry)">
Example	$PFAL,MSG.Send.TCPBuffer,8,"AVL device outputs its GPS positions" $PFAL,MSG.Send.TCPBuffer,0,"on &(Date) at &(Time) it is moving at &(Speed) m/s"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command sends data immediately to the TCP server even if the device is currently sending the (buffered) data to the server. This command is used to send actual data to the server, if device is currently sending big data archived from the FLASH.

Parameter description

Parameter	Value	Meaning
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (**entry**), which are described in chapter *7:*	

4.10.1.13. MSG.Send.RawTCPBuffer,<protocols>,<"text"> - Sends raw data immediately to the TCP server

Command Syntax	MSG.Send.RawTCPBuffer,<protocols>,"<"text"> MSG.Send.RawTCPBuffer,<protocols>,"<"text&(entry)">
Example	$PFAL,MSG.Send.RawTCPBuffer,8,"AVL device outputs its GPS positions" $PFAL,MSG.Send.RawTCPBuffer,0,"on &(Date) at &(Time) it is moving at &(Speed) m/s"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command sends raw data immediately to the TCP server even if the device is currently sending the (buffered) data to the server. This command is used to send actual raw data to the server, if device is currently sending big data archived from the FLASH.

Parameter description

Parameter	Value	Meaning
<protocols>	See the protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (**entry**), which are described in chapter *7:*	

4.10.1.14. MSG.Send.UDP,<protocols>,<"text"> - Sends data via UDP

Command Syntax	MSG.Send.UDP,<protocols>,<"text"> MSG.Send.UDP,<protocols>,<"text&(entry)">
Example	$PFAL,MSG.Send.UDP,8,"AVL device outputs its GPS positions" $PFAL,MSG.Send.UDP,0,"on &(Date) at &(Time) it is moving at &(Speed) m/s"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Sends the specified protocols and/or user text to the configured server during an established UDP connection.

Parameter description

Parameter	Value	Meaning
<protocols>	Defines the protocols to be sent to a remote server via UDP. See protocol definition in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (**entry**), which are described in chapter *7:*	

Notes

- If the UDP connection is still not established, this data will be written into the UDP history buffer and sent as soon as the UDP connection will be established.
- If the UDP connection is already established, all data will be sent; no acknowledges available, so the data might get lost during the transmission.

4.10.1.15. MSG.Send.SMTP,<email_address>,<protocols>,<"text"> - Sends data via Email

Command Syntax	MSG.Send.SMTP,<"email_address">,<protocols>,<"usertext"> MSG.Send.SMTP,<"email_address">,<protocols>,<"usertext&(entry)">
Example	$PFAL,MSG.Send.SMTP,"test@mail.com",8,"AVL device outputs its GPS positions" $PFAL,MSG.Send.SMTP,"test@mailserver.com",0,"on &(Date) at &(Time) it is moving at &(Speed) m/s"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Please refer to *4.8.3.1.* **TCP.SMTP.Send** for further details to parameters and general use.

Parameter description

Parameter	Value	Meaning
<"email_address">	It specifies the recipient of the E-Mail message to be sent. Only one E-Mail address is permitted per message. The E-Mail address must be wrapped in quotation marks (" ").	
<protocols>	It specifies the kind of protocols to be transmitted to an email address. Supported protocols are listed in chapter *11.2.*	
<“text”>	Up to 1450 chars of user defined text can be specified here. This text may also contain dynamic variables (**entry**), which are described in chapter *7:*	

4.10.2. Msg.ClearBuffer - Clears all information in a buffer

Command Syntax	Msg.ClearBuffer,<interface>
Example	$PFAL,Msg.ClearBuffer,TCP

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✗

Command description

This command is used to delete all buffer information in the specified interface

Parameter description

Parameter	Value	Meaning
[<interface>]	Optional. specifies the interface on which the event for the user defined command must be generated. If the interface parameter is omitted, then the device will answer/respond on the channel receiving the command/text.	
	Serial0	Serial port 0
	Serial1[1]	Serial port 1
	USB[2]	USB port
	TCP	TCP interface

4.10.3. MSG.Mode

Commands in this group allow to change the mode of all available message in/outputs (i.e. SERIAL, CSD, TCP input/output). Usually all message inputs are configured for command mode, which allows PFAL commands being executed there. Several **<output_messages>** can be disabled/enabled for each message output. Once the mode of one message input has been changed to data mode, this input will direct all its data to the configured **<msg_outputs>.**

Warning: ***You can change a message input to data mode by sending a command to it, but you cannot exit data mode this way. You would have to use another message input in order to change the mode. If all message inputs are configured for data mode, commands can be only entered via SMS. You can change data modes always via SMS.***

4.10.3.1. MSG.Mode.<interface>=<output_mode>,<input_mode> - Transparent mode between interfaces/channels

Command Syntax	MSG.Mode.<interface> *//read command* MSG.Mode.<interface>=<output_mode>,<input_mode> *//set command*
Example	$PFAL,MSG.Mode.Serial0 $PFAL,MSG.Mode.CSD=6F,C $PFAL,MSG.Mode.TCP=60,D=4

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	×
2	✓	✓	✓	×
3	✓	✓	✓	✓

Command description

Allows to change the mode of all available message in/outputs (i.e. SERIAL, CSD, TCP input/output). To read settings of one interface, just specify the interface as a string without adding any other settings (***e.g.*** $PFAL,MSG.Mode.Serial0). Reading data at a time on the serial ports (serial data size) is limited to 1024 bytes.

Parameter description

Parameter	Value	Meaning
<interface>	Defines the interface to be configured.	
	Serial0	Serial port 0.
	Serial1[1]	Serial port 1
	USB[2]	USB port (mode settings not available, default: 7F,C)
	CSD[3]	CSD interface
	TCP	TCP interface
<output_mode>	It is a hexadecimal number which specifies the information sent out via the corresponding interface output.	
	1	Transmits GPSTATE.
	2	Transmits GPERROR
	4	Transmits GPACTION
	8	Transmits GPEVENT
	10	Protocols (GGA, RMC.., configurable with protocol settings (see configuration reference).
	20	Data Transfer (interface shows data being transferred from other interfaces in data transfer mode).
	40	Data Output (interface shows data being created by commands like ***MSG.Send.Serial, GPS.Histoy.Push*** etc.).
	80	Reserved.
<input_mode>		
	C	Command mode (default).
	Note: *Non command text ended with <LF> is considered as event (like in event text mode). If just events are expected, event text mode should be chosen instead of command mode (as the event text could interfere with command text and might be misinterpreted then).*	
	T	Textual command mode. New "experimental" textual PFAL command mode which: - can execute classic textual PFAL commands like classic PFAL command mode "C" - can execute AVL Logger compatible binary debug commands. - currently does not create text events for texts ended with CRLF. Use Text mode for this feature - it is recommended to switch back to classic pfal command mode before starting classic remote update on port serial0.
	D= <msg_output_channel>	**Data transfer mode** (incoming data is directly transferred to **<msg_output_channel>,** no commands will be executed). <msg_output_channel> It is a hex value specifying where the serial input data will be forwarded to. **Value** **Meaning** **1** Serial 0 output. **2** [1] Serial 1 output. **4** [2] USB port **10** TCP output (is buffered if connection isn't established).

Parameter	Value	Meaning
	E	Event text mode (required for serial/CSD interface only). All incoming textual data is considered as text (even when it contains a PFAL Command). Whenever a <LF> characters is received, an event will be generated (containing the previously received text). This text can be reported using the serial data dynamic variable, so it is possible to e.g. launch an alarm action to send out this text (e.g. bar-code scanner application, RFID application).
	B	Binary event mode. Incoming data is considered as an event when the configured binary event settings match (see configuration reference binary event mode for more details). Whenever a configurable stop byte is received, an event will be generated (containing the previously received event data). This text can be reported using the serial data dynamic variable, so it is possible e.g. to launch an alarm action to send out this text (e.g. customized barcode scanner application, RFID application).

Notes

- Output message numbers can be added if several information fields are required. (i.e. to enable only GPEVENT, GPSTATE and GPERROR messages (prevent GPACTION), the corresponding number would be 0B).
- <msg_output_channel> numbers can be added, if data must be sent to several outputs (i.e. to send it to Serial0 and TCP, the corresponding number would be 5).

4.10.4. MSG.Event

4.10.4.1. MSG.Event – Creates event with text on different interfaces

Command Syntax	MSG.Event[,<interface>],<"text">
Example	$PFAL,MSG.MSG.Event,Serial0,"get.data"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	×
2	✓	✓	✓	×

Command description

This command generates events on the specified interface. If the <interface> is omitted, the event is generated on that interface the command was received.

Parameter description

Parameter	Value	Meaning
[<interface>]	Defines the interface to generate the event based on the specified text.	
	Serial0	Serial port 0.
	Serial1[1]	Serial port 1
	USB[2]	USB port
	User	User Interface

Parameter	Value	Meaning
	TCP	TCP interface
<“text”>	Defines the event text.	

4.10.5. MSG.Version

4.10.5.1. MSG.Version.Complete - Retrieves device firmware and hardware versions

Command Syntax	MSG.Version.Complete
Example	$PFAL,MSG.Version.Complete

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command retrieves the complete version information of current device.

Parameter description

None.

4.10.5.2. MSG.Version.Modules - Retrieves modules versions

Command Syntax	MSG.Version.Modules
Example	$PFAL,MSG.Version.Modules

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command retrieves version information from modules of current device

Parameter description

None.

4.10.5.3. MSG.Version.BIOS - Retrieves BIOS version

Command Syntax	MSG.Version.BIOS
Example	$PFAL,MSG.Version.BIOS

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command retrieves the software version of the currently used firmware.

Parameter description

None.

4.10.5.4. MSG.Version.HardwareRev - Retrieves device hardware revision

Command Syntax	MSG.Version.HardwareRev
Example	$PFAL,MSG.Version.HardwareRev
Responses	<MSG.Version.HardwareRev> $11-NUCH $SUCCESS

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command retrieves the hardware revision number of current device

Parameter description

None.

4.10.5.5. MSG.Version.Hardware - Retrieves device hardware name

Command Syntax	MSG.Version.Hardware
Example	$PFAL,MSG.Version.Hardware

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command retrieves the hardware name of the device.

Parameter description

None.

4.10.5.6. MSG.Version.Software - Returns the software version of the target device

Command Syntax	MSG.Version.Software
Example	$PFAL,MSG.Version.Software

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This read command returns the version of the operating software of AVL device.

Parameter description

None.

4.10.5.7. MSG.Version.AddTag – Adds a configuration profile identifier in device name

Command Syntax	MSG.Version.AddTag,<”tag”>
Example	$PFAL,MSG.Version.AddTag,"EcoDrive"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command adds an identifier to easily identify the profiles of your device configuration. It is appended to the device name (DEVICE.NAME), closed in square brackets. To read your added identifiers, just call “$PFAL,Cnf.Get, DEVICE.NAME” from your server. The device name will be shown e.g. FOX3-3G-V123 [EcoDrive] [1-Wire] [Basic]

Parameter description

Parameter	Value	Meaning
<tag>	Specifies a string (e.g. “EcoDrive” or “1-Wire” etc.)	

4.10.5.8. MSG.Version.DelTag – Removes a configuration functionality identifier from device name

Command Syntax	MSG.Version.DelTag,”<tag>”
Example	$PFAL,MSG.Version.DelTag,"EcoDrive"

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command removes an already specified configuration profile identifier from the device name.

Parameter description

Parameter	Value	Meaning
<tag>	Specifies a string (e.g. “EcoDrive” or “1-Wire” etc.)	

4.10.6. MSG.Info

This chapter contains commands which can be used to retrieve various information form the ALV device. The desired information will be returned within the PFAL answer.

4.10.6.1. MSG.Info.ServerLogin – Shows the login data to be reported to a server

Command Syntax	MSG.Info.ServerLogin
Example	$PFAL,MSG.Info.ServerLogin
Responses	-----*TCP.CLIENT.LOGIN=2,[1,2]* ---------- $<MSG.Info.ServerLogin> $DeviceName=Unnamed AVL $Software=avl_3.0.0_rc14 $Hardware=FOX3 rev:03-NUCH $IMEI=353816054739497 $PhoneNumber=+491773456789 $Position=$GPRMC,143545.000,A,5040.4086,N,01058.8543,E, 0.00,0.00,040315,, $SUCCESS $<end> ------*with TCP.CLIENT.LOGIN=1,[1,2]* ---------- $<MSG.Info.ServerLogin> $DeviceName=FOX3 $Software=avl_3.0.0_rc14 $Hardware=FOX3 rev:03-NUCH $LastValidPosition=$GPRMC,092151.000,A,5040.4078,N,01058 .8530,E,0.05,0.00,110309,, $IMEI=357023001066142 $LocalIP=191.142.223.24 $CmdVersion=2 $SUCCESS $<end>

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Reports login information data from the device which is needed to identify it on the Lantronix server. This information consists of device name, hardware, software version, IMEI and Local IP as well as the last valid position.

Feature: “AES security" :

- Entry “Security=<security_mode>" new added.
- If the feature “AES security" is enabled and AES security mode configured (see TCP.CLIENT.LOGIN for more information), this feature will be activated directly after an initial MSG.Info.ServerLogin message has been sent to the server.
- If MSG.Info.ServerLogin is inactive, security is set directly after the TCP client eConnected event.

Note: *When querying MSG.Info.ServerLogin from serial interface before a TCP connection is established, “Security=0" will be displayed.*

Parameter description

None.

4.10.6.2. MSG.Info.Protocol,<protocols>,<"text"> - Reports the protocol information data

Command Syntax	MSG.Info.Protocol,<protocols>,<"text">
Example	$PFAL,MSG.Info.Protocol,8,""

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Reports protocol information plus user text from the device. The specified user text is displayed first.

Parameter description

Parameter	Value	Meaning
<protocols>	See protocol definition in chapter *11.2.*	
<"text">	Up to 200 chars of user defined text can be specified here. This field can be also left empty like this **<protocols>,""**.	

4.10.6.3. MSG.Info.Time – Reports the time, date, week number and week day of the device

Command Syntax	MSG.Info.Time
Example	$PFAL,MSG.Info.Time

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

Retrieves current time, date, week number and week day of the device. A valid GPS position is required for successful execution of this command*.*

Parameter description

None.

4.10.6.4. MSG.Info.Alarm,<alarm_index> - Shows all conditions of specific alarm index

Command Syntax	MSG.Info.Alarm,<alarm_index>
Example	$PFAL,MSG.Info.Alarm,0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	🗝	🗝	🗝	🗝

Command description

Displays all conditions of the selected alarm and shows their current state *(whether they are true or false)*.

Parameter description

Parameter	Value	Meaning
<alarm_index>	Specifies a number between **0** and **249** of the available alarm indices to be checked.	
	0...99	Indices of the standard alarms
	100...249[1]	Indices of the extended alarms

Notes

- This command can be used to check even most complex alarm configurations step by step to validate the desired behavior.
- Events are always shown as "true" (even if there are several events inside an alarm - in reality such an alarm could never be executed).

4.10.7. MSG.Feature

This group allows control the device PREMIUM-FEATURE management. Currently set options can be viewed – or new feature codes installed in the device. Feature codes allow enabling specific device features which otherwise are not accessible.

4.10.7.1. MSG.Feature – Reports device PREMIUM-FEATURE and installation state.

Command Syntax	MSG.Feature
Example	$PFAL,MSG.Feature
Responses	If features already installed on the device: $<MSG.Feature> $IndexedHistory: inactive $AES_TCP: active (never expires) $EcoDrive: active (never expires) $Can: active (never expires) $RS485: inactive $eCall: inactive $ExtGeofence: active (never expires) $ExtAlarms: inactive $SUCCESS If no features installed on the device: $<MSG.Feature> $IndexedHistory: inactive $AES_TCP: inactive $EcoDrive: inactive $Can: inactive $RS485: inactive $eCall: inactive $ExtGeofence: inactive $ExtAlarms: inactive $SUCCESS

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command gives an overview about available PREMIUM-FEATURES and installation state within the device. Do not use this command within alarms.

Parameter description

None.

4.10.7.2. MSG.Feature=<"featurecode"> – Installs specific feature code on the device.

Command Syntax	MSG.Feature=<"feature_code">
Example	-

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Command description

This command installs a feature code on the device. A Feature code can enable/disable one or several features. For a list of available features which can be installed on your device, please contact your local sales office. Do not use this command within alarms.

Parameter description

Parameter	Value	Meaning
<"feature_code">	Enters the code for the activation and use of the Premium-features. Please contact your local sales office for further details on Premium-features.	

5: Configuration Settings

This chapter gives a full overview of all available basic configuration settings which are stored in the non-volatile FLASH memory inside the device.

All parameter settings that are supported by the firmware version 2.5.0 and later can only be sent to the target device by using the "$PFAL,Cnf.Set,<parameter>" command, where <parameter> (with extended syntax: <parameter>=<value>) can be one of the parameters given within this chapter.

The AVL device provides the following configuration setting types.

User Setting	It is only for special configuration settings, which are set/defined by the user. This configuration can be deleted or overwritten by the user.
Factory Setting	It consists of configuration settings preloaded at the factory in the AVL device. This configuration is dependent on the device variants and usually is factory installed as User Settings. The user has the possibility of personalizing this configuration and with the help of PFAL, Sys.Device.FactoryReset command it is possible to overwrite the User Settings with the Factory Settings.
Default Setting	It is about basis configurations that the AVL device needs during each system startup. This configuration is required to enable the AVL device to run stably. Default settings are always called up, if there are no settings available in the User Settings. If, for whatever reason this parameter settings are deleted, then the system calls up the Default Setting for that parameter. Remark: The parameters configuration settings that are available in the Default Setting could not be deleted they can only be overwritten.

Each parameter name is listed in its own subsection. The parameter syntax table indicates the format <parameter>=<value> (which could not be sent to the device in that form).

Parameter syntax	DEVICE.NAME=<name>

The example table shows how the parameter can be configured and sent to the AVL device. The entered values demonstrate the use of these fields.

Set configuration	$PFAL,Cnf.Set,DEVICE.NAME=alfa_car
Get configuration	$PFAL,Cnf.Get,DEVICE.NAME

Each parameter name described in the sections below should be sent via the PFAL message ($PFAL), which is transmitted in the form of "sentences". The sentence begins with "$" character (not by SMS), next come the four letters "PFAL" separated by comma ‘,’. The command "Cnf.Set", followed by the parameter separated by commas, and terminated by a calculated checksum (if used), and a carriage return/line feed. The checksum in the examples below is omitted.

Note: *Configuration parameter names which contain upper and lower case letters are stored but won't be used by the system. They are considered as special user settings, not as device settings).*

Table 5-1 List of Configuration Parameters

PARAMETER NAMES	BRIEF DESCRIPTION	SECTION
DEVICE SETTINGS		
DEVICE.NAME	Defines or changes the device name.	5.1.1.
DEVICE.SERIAL<port>.BAUDRATE	Modifies/changes the baud rate of the serial communication line.	5.1.2.

PARAMETER NAMES	BRIEF DESCRIPTION	SECTION
DEVICE.SERIAL<port>.FORCEON=<on_off>	Sets the main port on the AVL device permanently active or passive	5.1.3.
DEVICE.SERIAL<port>.HANDSHAKE	Enables or disables the software handshake for the specified serial port	5.1.4.
DEVICE.CMD.PFAL.EN	Enables/disables the supported communication services to the device.	5.1.5.
DEVICE.COMM.<interface>	Determines the communication mode on the allowed interfaces. This parameter doesn't have to be configured with **CNF.SET**. The PFAL command **"Msg.Mode"** can be used instead.	5.1.6.
DEVICE.SERIAL1.MODE485=<on_off>	EOL (end of life feature). Enables or disables the communication with other RS-458 devices.	5.1.7.
DEVICE.COMM.BINEVENT<port>	Allows the device to control the start byte(s) and stop byte of the data received on the serial port for occurring the serial data event	5.1.8.
DEVICE.BAT.MODE	Specifies the current battery mode	5.1.9.
DEVICE.BAT.CHARGEMODE	Specifies the operating mode of battery charger	5.1.10.
DEVICE.IGNTIMEOUT	Determines the amount of timeout the system will wait until it shuts down.	5.1.11.
DEVICE.GSM.STARTUP	Enables/disables the GSM engine on the startup.	5.1.12.
DEVICE.GPS.AUTOCORRECT	Implements an internal GPS position check routine, to filter out the incorrect or improbable GPS positions. Its main purpose is to filter out the "GPS Jumps" in areas with very poor GPS signal quality.	5.1.13.
DEVICE.GPS.CFG	Specifies the number of satellites to consider a GPS fix as valid.	5.1.14.
DEVICE.GPS.TIMEOUT	Allows to specify a timeout after which the device performs a GPS reset if it has no valid fix.	5.1.15.
DEVICE.GPS.NAVDISTMUL	Used as a multiplicator for NavDis feature to adapt it to the ODOmeter of the vehicle.	5.1.16.
DEVICE.PFAL.SEND.FORMAT	Generates a specific format for all **MSG.Send** commands	5.1.17.
DEVICE.CAN.OBD.STARTUP	Saves and defines the frame format to be enabled for reading of OBDII messages on the specified CAN interface.	5.1.18.
DEVICE.CAN.FMS.STARTUP	Saves and enables or disables reading of FMS messages on the specified CAN interface.	5.1.19.
DEVICE.CAN.STARTUP	Saves and enables or disables reading of CAN messages on the specified CAN interface using the user specified baudrate.	5.1.20.
DEVICE.CAN.DTCOFMS.STARTUP	Saves and enables or disables reading of tachograph data via FMS interface on the specified CAN interface.	5.1.21.
DEVICE.CAN.ERR.EVENTS	Saves and enables or disables error CAN messages on the both CAN interface.	5.1.22.
DEVICE.CAN.EVENT_<slot>	Generates events out of changed CAN variables without checking them periodically for changes.	5.1.23.
DEVICE.DTCO.D8	Enables or disables the D8 input (IN4, PIN 8) of the IOBOX-CAN accessory device	5.1.24.
DEVICE.LOWPOWER	Sets the device into a low power mode	5.1.25.
DEVICE.GPS.JAMMINGLEVEL=<min>,<max>	Detects abnormalities during operation e.g. the possible presence of GPS jammers, interference, noise, jamming, after re-acquisition.	5.1.26.

PARAMETER NAMES	BRIEF DESCRIPTION	SECTION
DEVICE.WLAN.STARTUP	Sets up the general configuration settings for using the IOBOX-WLAN	5.1.27.
DEVICE.NFC	Enables or disables connection to a Lantronix NFC reader on a serial port.	5.1.28.
DEVICE.VIN	Enables or disables connection to a Lantronix NFC reader on a serial port.	5.1.29.
DEVICE.SERIAL1.MODE485	Enables or disables the RS485 mode of the Serial1 interface.	5.1.30.
MACRO SETTINGS		
MACRO<index>	Designed for executing a large number of commands within a single line.	5.20.12.
REPLACE SETTINGS		
REPLACE<index>	Allows you to replace a specified number of strings (such as tel, numbers, user text etc.) with a specified replacement string. It helps you change the text just one time instead each Alarm text (AL) separately.	5.2.1.
USER SETTINGS		
USER<index>	Allows to define user text for sending to the TCP server.	5.3.1.
MOTION SETTINGS		
MOTION.FILTER	Sets the motion filter.	5.4.1.
MOTION.BEARING	Defines the configuration settings to filter bearings of device motions	5.4.2.
MOTION.FORCE	Used to raise the Force event.	5.4.3.
MOTION.3DFORCE	Raises Force events whenever one of the specified axis directions of the attitude sensor exceeds the specified acceleration.	5.4.4.
ALIAS SETTINGS		
ALIAS.<type>	Defines alias names for each command type.	5.5.1.
DEBUBG SETTINGS		
DBG.EN	Enables or disables debug information to be output on the serial interface.	5.6.1.
PROTOCOL SETTINGS		
PROT.<message_id>	Not only allows certain messages to be enabled or disabled but also specifies the rate at which they are sent to the serial interface.	5.7.1.
PROT.START.BIN	Creates your own BIN protocol. Consists of 18 Bytes and used as a device identifier for example.	5.7.2.
PROT.ERR	Defines a value as a string when a read command returns error	5.7.3.
PROT.NA	Defines a value as a string when the source of the value is not available	5.7.4.
PROT.EMPTY	Defines a value as a string when the read variable is empty yet.	5.7.5.
ECODRIVE SETTINGS		
ECODRIVE.CAR	Used to define the car parameters when controlling fuel consumption	5.8.1.
ECODRIVE.LIMITS	Used to define the braking deceleration and acceleration limits for city and country road topologies.	5.8.2.
ECODRIVE.TOPOLOGY	Used to define the limits for the detection of the different road types and times for switching between the different topologies.	5.8.3.

PARAMETER NAMES	BRIEF DESCRIPTION	SECTION
ECODRIVE.AUTOSTART	Used to define the trigger conditions and limits for an automatic start and stop of a Eco-Drive trip.	5.8.4.
GSM SETTINGS		
GSM.PIN	Sets the PIN of used SIM card (only locally possible).	5.9.1.
GSM.BALANCE.DIAL	This setting might have to be modified only for some operators, which do not rely on this standard dial number.	5.9.2.
GSM.CALLID.EN	Enables/disables the caller identification.	5.9.3.
GSM.OPERATOR.BLACKLIST	Creates a customized blacklist consisting of GSM operator names.	5.9.4.
GSM.OPERATOR.SELECTION	Configures the GSM operator selection.	5.9.5.
GSM.EXT.WHITELIST	Specifies which operator IDs should additionally be used to the list of operator IDs added in the parameter GSM.OPERATOR.SELECTION.	5.9.6.
GSM.OPERATOR.GPRSCHECK	Enables or disables the check for GSM base cells whether they provide GPRS service or not.	5.9.7.
GSM.OPLOST.RESTART	Determines whether the system should reinitialize the GSM engine when the GSM operator becomes lost.	5.9.8.
GSM.SMS.RESPONSE	Enables/disables responses from the device when SMS communication is applied.	5.9.9.
GSM.MODEPREF	Selects radio access modes and preferences for the FOX3 (2G)/3G devices	5.9.10.
GSM.SIMSLOT	Selects the used SIM card slot in the Lantronix device	5.9.11.
GPRS/PPP SETTINGS		
GPRS.APN	Specifies the APN (Access Point Name) context for GPRS connection.	5.10.1.
GPRS.AUTOSTART	Enables/disables the GPRS auto start. It performs automatically a GPRS attachment even if it fails unexpectedly.	5.10.4.
GPRS.DIAL	Specifies the dial text used for GPRS connection.	5.10.5.
GPRS.TIMEOUT	Specifies the time out in which the GPRS connection will be automatically shut down if there is no TCP communication currently available.	5.10.6.
GPRS.QOSMIN	Specifies the minimum acceptable profile for identified context.	5.10.7.
GPRS.QOS	Specifies the acceptable profile for identified context.	5.10.8.
PPP.USERNAME	Specifies the user name to connect to the GPRS network.	5.11.1.
PPP.PASSWORD	Specifies the password to connect to the GPRS network operator.	5.11.3.
PPP.AUTOPING	Enables/disables the maximal idle time until a ping is sent to the GPRS network to keep the GPRS connection alive.	5.11.5.
PPP.AUTH	Defines the authentication method to be used over PPP.	5.11.6.
TCP SETTINGS		
TCP.CLIENT.CONNECT	Specifies the IP address and port number of the remote server.	5.12.1.
TCP.CLIENT.ALTERNATIVE	Defines an alternative server if the primary server fails.	5.12.2.
TCP.CLIENT.PING	Enables/disables the sending of pings to the remote server to keep that connection alive.	5.12.3.

PARAMETER NAMES	BRIEF DESCRIPTION	SECTION
TCP.CLIENT.TIMEOUT	Specifies the length of time the device waits between attempts to establish a TCP connection.	5.12.4.
TCP.CLIENT.DNS.TIMEOUT	Specifies the DNS cache timeout.	5.12.5.
TCP.CLIENT.LOGIN	Specifies the login data to log the device into the used remote server.	5.12.6.
TCP.CLIENT.LOGIN.EXT	Specifies additional information that will be added to the regular login data and sent to the remote server.	5.12.7.
TCP.CLIENT.SENDMODE	Selects between fast and safe TCP transmissions	5.12.8.
TCP.SERVICE.CONNECT	Configures a TCP service connection, which allows to run services like remote control, configuration, remote updates, AGPS etc.	5.12.9.
TCP.SERVICE.TIMEOUT	Defines the period of time the device will wait for a response and between two connection attempts when the TCP connection fails.	5.12.10.
TCP.STORAGE	Specifies the size of TCP storage and the operation mode.	5.12.11.
TCP.SMTP.CONNECT	Configures the SMTP email settings	5.12.12.
TCP.SMTP.SUBJECT	Defines the authentication for the SMTP session	5.12.13.
TCP.SMTP.LOGIN	Defines the authentication for the SMTP session	5.12.14.
TCP.SMTP.FROM	Configures the email address of the message sender	5.12.15.
TCP.SLA	Activates or deactivates the SLA (Service License Agreements) connections to the Lantronix SLA-Server	5.12.16.
TCP.CLIENT2.CONNECT	Specifies the IP address and port number of the second TCP connection	5.12.17.
TCP.CLIENT2.ALTERNATIVE	Defines an alternative server for the second TCP connection in case the primary server fails.	5.12.18.
TCP.CLIENT2.SENDMODE	Selects between fast and safe TCP transmissions for the second TCP connection.	5.12.19.
TCP.CLIENT2.PING	Enables/disables the sending of pings to the remote server for the second TCP connection to keep that connection alive	5.12.20.
TCP.CLIENT2.TIMEOUT	Specifies the length of time the device waits between attempts to establish a TCP connection for the second TCP connection.	5.12.21.
TCP.CLIENT2.LOGIN.EXT	Specifies additional information that will be added to the regular login data and sent to the remote server for the second TCP connection.	5.12.22.
TCP.CLIENT2.LOGIN	Specifies the login data to log the device into the used remote server for the second TCP connection.	5.12.23.
TCP.CLIENT2.DNS.TIMEOUT	Specifies the DNS cache timeout for the second TCP connection.	5.12.24.
TCP.CLIENT2.CERT	Sets the use of certificates for TLS for the second TCP connection.	
SMTP SETTINGS		
TCP.SMTP.CONNECT	Configures the sending of E-Mail via an Internet mail server.	5.12.12.
TCP.SMTP.SUBJECT	Specifies the subject when sending emails	5.12.13.
TCP.SMTP.LOGIN	Identities the authentication for the SMTP session when sending E-Mail to an Internet mail server.	5.12.14.
TCP.SMTP.FROM	Configure the E-Mail address of the sender of the message.	5.12.15.
WLAN SETTINGS		

PARAMETER NAMES	BRIEF DESCRIPTION	SECTION
WLAN.CLIENT.LOGIN	activates or deactivates the SLA (Service License Agreements) connections to the Lantronix SLA-Server	5.13.2.
WLAN.NET<id>.SSID	Sets the SSID of the access points to which the IOBOX-WLAN could be connected	5.13.3.
WLAN.NET<id>.PSK	set the Hotspot password to connect	5.13.5.
WLAN.NET<id>.SECURITY	Sets the encryption method of data transmission between IOBOX-WLAN and Hotspot.	5.13.6.
WLAN.NET<id>.IP	Sets the IP address of the remote server to connect to after connecting to the WLAN network.	5.13.7.
WLAN.NET<id>.PORT	Sets the TCP port of the remote server to connect to after connecting to the WLAN network.	5.13.8.
WLAN.RSSIMIN	Scans for networks with RSSI levels greater than the user defined level	5.13.9.
BLE SETTINGS		
BLE.ADVNAME	Sets up the name of the FOX3-3G-BLE devices for using BLE features	5.14.1.
BLE.WHITELIST	Creates a list with BLE sensors and specific attribute/property	5.14.2.
UDP SETTINGS		
UDP.CLIENT.CONNECT	Specifies the type, IP-address and port that your application will use to connect to the remote server via UDP protocol.	5.16.1.
UDP.CLIENT.TIMEOUT	Specifies the allowed timeout between reconnections to the remote server, when the TCP connection fails.	5.16.2.
GEOFENCE SETTINGS		
GF.CONFIG	Sets the global configuration of the Geofence functionalities.	5.17.1.
GF.AREA	Allows setting up 32 areas in a range of 0 to 31.	5.17.2.
GF<id>	Sets up the shape of the Geofence zone.	5.17.3.
AL SETTINGS		
AL<index>	Allows to specify a powerful multi-configuration and reporting messages as well as to create a maximum monitoring of system events and states which can be occurred during the operation of the device.	5.18.1.
CONSOLEFLOW SETTINGS		
CF.CLIENT.CONNECT	Enables, disables ConsoleFLow client and configures server URL.	5.19.1.
CF.CLIENT.DEVICE_NAME	Configures device name.	5.19.2.
CF.CLIENT.DEVICE_DESCRIPTION	Configures device description.	5.19.3.
CF.CLIENT.STATUS_UPDATE_INTERVAL	Configures interval between status updates sent by device to ConsoleFlow server.	5.19.4.
CF.CLIENT.CONTENT_CHECK_INTERVAL	Configures interval between checks made by device for available content at the server.	5.19.5.
CF.CLIENT.GROUPS_CAP_EXCHANGE_DATA	Informs device about custom telematic groups to publish to the ConsoleFlow sever.	5.19.6.
CF.CLIENT.GROUPS_CAP_SELECTION_DATA	Configures metadata of the custom telematic groups published.	5.19.7.

PARAMETER NAMES	BRIEF DESCRIPTION	SECTION
CF.CLIENT.GROUPS_TELEMETRY_DATA	Configures telematic data of each parameter with its value.	5.19.8.

5.1. DEVICE

5.1.1. DEVICE.NAME

This parameter allows you to set or change the device name.

Parameter syntax	DEVICE.NAME=<name>

<name>

It identifies the name of AVL device. When the device sends an SMS message to the message sender or target phone number, it identifies itself using this identifier.

How the configuration could be set/requested:

Set configuration	$PFAL,Cnf.Set,DEVICE.NAME=alfa_car
Get configuration	$PFAL,Cnf.Get,DEVICE.NAME

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

If the device name has already been set, each other <name> message sent to the device overwrites the existing entry.

5.1.2. DEVICE.SERIAL<port>.BAUDRATE

This parameter allows you to modify the baud rate of specified serial port.

Parameter syntax	DEVICE.SERIAL<port>.BAUDRATE=<baudrate>[,<data><parity><stop>]

Parameter	Value	Meaning
<port>	AVL device supports 2 serial ports. Each of them can be used to transmit data. This entry specifies the serial port number to change its baudrate. It can be set to:	
	0	Serial port 0
	1[1]	Serial port 1
<baudrate>	Specifies the baud rate of the serial communication line. The default setting for the firmware version 2.10.0 and later is 115200 bps. The default setting for the firmware version 2.9.0 and lower is 57600 bps. Following baud rates can be used: 4800, 9600, 19200, 38400, 57600 and 115200 bps.	
[<data>]	This setting is optional and may contain additional data bits. The data bit can set to 7 or 8.	

Parameter	Value	Meaning
[<parity>]	This setting is optional and may contain additional parity check bits. The parity bit can be set to:	
	N	No parity bit (default if <parity> is not specified)
	E	Even parity
	O	Odd parity
[<stop>]	This setting is optional and may contain additional stop bits. The stop bit can be set to 1 or 2.	

Hardware support available is limited, to implement this. Only the modes 7E1, 7O1, 8N1, 8E1 and 8O1 with 1 or 2 stop bits, are supported by the AVL firmware.

How the configuration could be set/requested:

Set configuration	$PFAL,Cnf.Set,DEVICE.SERIAL0.BAUDRATE=115200 $PFAL,Cnf.Set,DEVICE.SERIAL0.BAUDRATE=115200,8N1 $PFAL,Cnf.Set,DEVICE.SERIAL0.BAUDRATE=115200,7E1 $PFAL,Cnf.Set,DEVICE.SERIAL0.BAUDRATE=115200,7O1 $PFAL,Cnf.Set,DEVICE.SERIAL0.BAUDRATE=115200,8E1 $PFAL,Cnf.Set,DEVICE.SERIAL0.BAUDRATE=115200,8O1 $PFAL,Cnf.Set,DEVICE.SERIAL0.BAUDRATE=115200,8O2
Get configuration	$PFAL,Cnf.Get,DEVICE.SERIAL0.BAUDRATE

If no mode is specified the default mode used is 8N1. For example,
$PFAL,Cnf.Set,DEVICE.SERIAL0.BAUDRATE=115200 is same as
$PFAL,Cnf.Set,DEVICE.SERIAL0.BAUDRATE=115200,8N1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	×

Notes

If the baud rate already has been set, each other value overwrites the existing entry. Changing this value becomes active within 10 seconds (GPSTATE will display the countdown). Also, the response of PFAL command will be transmitted using the new baud rate.

5.1.3. DEVICE.SERIAL<port>.FORCEON=<on_off>

Parameter syntax	DEVICE.SERIAL<port>.FORCEON=<on_off>

This parameter sets the main port on the AVL device permanently active or passive.

Parameter	Value	Meaning
<port>	It defines the serial port number on the AVL device to be configured.	
	0	Serial port 0

Parameter	Value	Meaning
<on_off>	It defines the mode you want to set on the defined serial port number. Only the 1st serial port on the main port (8pin connector) can be configured. The 2nd serial port on the accessory port (6pin) does not support this setting.	
	on	Permanently active. In this mode, the 1st serial port will stay all the time active.
	off	Permanently passive (default).

How the configuration could be set/requested:

Set configuration	$PFAL,Cnf.Set,DEVICE.SERIAL0.FORCEON=on
Get configuration	$PFAL,Cnf.Get,DEVICE.SERIAL0.FORCEON

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.1.4. DEVICE.SERIAL<port>.HANDSHAKE

Parameter syntax	DEVICE.SERIAL<port>.HANDSHAKE=<on_off>

This parameter enables or disables the software handshake for the specified serial port of the AVL device. Handshaking is used to control data flow on connected serial ports.

Parameter	Value	Meaning
<port>	Specifies the number of the serial port of the AVL device.	
	0	Serial port 0
	1[1]	Serial port 1
<on_off>	This entry specifies the serial port number you want to set to either permanently active or passive.	
	off	Disables the handshake
	soft[<set>]	Enables the software handshake using the following optional settings: [<set>]=,<map>,<xon>,<xoff>,<esc>,<xor> <map>: async map - a bit field as hex number, which defines the control characters to be sent to with the help of an escape sequence. Bit 0 means the character code 0x0, bit 31 the character code 0x1f. The DEL-code (0xff) is define by bit 32. <xon>: The hex code of the XON-character (default code is: 0x11) <xoff>: The hex code of the XOFF-character (default code is: 0x13) <esc>: The hex code of the ESC-character (default code is: 0x1B) <xor>: The hex code of the value, that will be XORed to the escaped character (default code is: 0x20)

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.SERIAL1.HANDSHAKE=soft $PFAL,CNF.Set,DEVICE.SERIAL1.HANDSHAKE=off $PFAL,CNF.Set,DEVICE.SERIAL1.HANDSHAKE=soft,80A0000,11,13,1b,20
Get Configuration	$PFAL,Cnf.Get,,DEVICE.SERIAL1.HANDSHAKE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✗

5.1.5. DEVICE.CMD.PFAL.EN

Parameter syntax	DEVICE.CMD.PFAL.EN=<hex_value>[,<show_pfal_executed_events>]

It allows you to enable/disable the supported communication services to the device and showing the events when PFAL commands are executed. The default setting in the firmware 3.1.0_rc20 is "3F", meaning that all types of communication services are enabled. The default setting in the firmware 3.0.0_rc20 is "F". When upgrading your devices to the firmware 3.1.0_rc20 and higher please change this setting to 3F, otherwise you cannot access your device via USB or BLE.

Parameter	Value	Meaning
hex_value	It consists of a hexadecimal number without leading "0x" that determines which communication services will be enabled, the omitted hexadecimal values disable the communication. If it is set to "0", it disables all communication channels in the downstream direction (from the user to device) and no more PFAL commands can be sent Serially, CSD, TCP, SMS, USB and BLE to the AVL device. Following are listed the communication services in hexadecimal value:	
	0x01	Enables serial communication to the device. If this value is omitted, the device blocks PFAL commands received through the Serial ports.
	0x02[1]	Enables CSD communication to the device. If this value is omitted, the device blocks PFAL commands received over the CSD data call.
	0x04	Enables TCP communication to the device. If this value is omitted, the device blocks PFAL commands received over the TCP channel.
	0x08	Enables SMS communication to the device. If this value is omitted, the device blocks PFAL commands received via SMS.
	0x10[2]2	Enables USB communication to the device. If this value is omitted, the device blocks PFAL commands received through the USB port.
	0x20[2]2	Enables BLE (Bluetooth Low Energy) communication to the device.
	0x3F[2]2	(Default) Enables all communication channels. The device can be accessed serially, CSD, TCP, SMS, USB, BLE.
show_pfal_executed_events>	This setting enables or disables showing of events when PFAL commands are executed:	
	0x01	Default value. Disables showing of events when PFAL commands are executed.
	0x02	Enables showing of events when PFAL commands are executed (for more details, see Sys.Device.ePfalExecuted).

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.CMD.PFAL.EN=A,1
Get Configuration	$PFAL,Cnf.Get,DEVICE.CMD.PFAL.EN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓
2	✓	✓	✓	✓

Notes

- In order to enable more than one communication services at once, specify the sum determined by adding the corresponding hex value of each communication services, for example:

```
The hexadecimal value A means: enable SMS+TCP
communication services, while all other services (the
omitted one) keep disabled.
```

- The default setting in the firmware 3.1.0_rc20 is “3F”, all types of communication services are enabled. The default setting in the firmware 3.0.0_rc20 is “F”. When upgrading your devices to the firmware 3.1.0_rc20 and above please change this setting to 3F, otherwise you cannot access your device via USB or BLE.

5.1.6. DEVICE.COMM.<interface>

Parameter syntax	DEVICE.COMM.<interface>=<input_mode>,<output_mode>

This configuration setting should not be changed manually. Please use PFAL,MSG.Mode commands instead.

Warning: ***Changes to these settings may bring the device in a mode, where it can be just reconfigured via SMS. If commands via SMS are disabled, the device cannot be reconfigured anymore. A serial Firmware update (via Serial0) is necessary then for being able to issue commands again.***

```
Compatibility mode:DEVICE.COMM.SERIAL0=cmd,7F
DEVICE.COMM.SERIAL1=cmd,7F
New syntax:
PFAL - Command mode:
DEVICE.COMM.SERIAL0=cmd2,7F
PFAL - Textual command mode:
DEVICE.COMM.SERIAL0=cmd3,7F
```

Parameter	Value	Meaning
<interface>	Defines the interface to be configured. It can be set to:	
	SERIAL0	Serial port 0
	SERIAL1[1]	Serial port 1
	USB[2]	USB port
	CSD[3]	CSD interface
	TCP.CLIENT	TCP interface
	TCP.SERVICE	TCP Service interface (see chapter *5.12.9.*). This interface setting is not supported in firmware 2.13.0_rc27 and later.
<Input_mode>	See PFAL,MSG.Mode command - chapter *4.10.3.1.*	
<output_mode>	See PFAL,MSG.Mode command - chapter *4.10.3.1.*	

How the configuration could be set/requested:

Set Configuration	-
Get Configuration	$PFAL,CNF.Get,DEVICE.COMM.SERIAL0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✗
2	✓	✓	✓	✗
3	✓	✓	✓	✓

5.1.7. DEVICE.SERIAL1.MODE485=<on_off>

Parameter syntax	DEVICE.SERIAL1.MODE485=<on_off>

This parameter can be used to enable or disable communication with other RS-458 devices. There is no special protocol like Modebus implemented in the firmware to control and monitor RS-458 devices. After enabling this interface, use the same events for processing the data that are available for the Serial1.

Note: *After activating and enabling this feature, its setting is saved in RAM and not in flash. To keep this feature enabled after a device restart, use the alarm given below:*

```
$PFAL,Cnf.Set,AL1=Sys.Device.eStart:MSG.Send.RawSerial1,0,"star
tup"&Cnf. Set,DEVICE.SERIAL1.MODE485=on
```

Parameter	Value	Meaning
<on_off>	After activating the premium-feature "RS-485", use one of the following settings to enable or disable communication with other RS-485 devices.	
	ON	Enable communication with external RS-485 connected accessory devices.
	OFF	Disable communication with external RS-485 connected accessory devices.

How the configuration could be set/requested:

Set Configuration	$PFAL,CNF.Set,DEVICE.SERIAL1.MODE485=on
Get Configuration	$PFAL,CNF.Get,DEVICE.SERIAL1.MODE485

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✗

Notes

RS-485 port is a hardware based Premium-feature which is already discontinued, but can be enabled only on request with a minimum order quantity (MOQ) of 1000 pcs.

5.1.8. DEVICE.COMM.BINEVENT<port>

Parameter syntax (1)	DEVICE.COMM.BINEVENT[<port>]=raw,<timeout>
Parameter syntax (2)	DEVICE.COMM.BINEVENT[<port>]=<startbyte(s)>,<stopbyte>,<use_stopbyte>

This setting allows the device to control the start byte(s) and stop byte of the data received on the serial port for occurring the serial data event. The serial port on which the BIN-Events should be generated are defined using the setting <port>.

Parameter	Value	Meaning
<port>	Optional entry. It specifies the serial port on which the BIN-Events should be generated. If this setting is left empty, the BIN-Events will be generated on both serial ports. It can be set to:	
	0	Serial port 0
	1[1]	Serial port 1
<timeout>[2]	This setting is very helpful when receiving data/string/ids on the serial ports of the FOX3 series without fixed end-of-text or termination characters e.g. line feeds. It specifies the number of milliseconds before a time-out occurs when reading data from the specified serial port. By default the read time-out value is set at 100 milliseconds. Setting the timeout to 1000 ms, allows the device to generate the corresponding serial data event every 1 second when serial data is received.	
<startbyte(s)>	Optional entry, which specifies the ASCII code (max. 10 characters in hex form e.g. 0x44) of the start characters that are required to execute the serial data event. If more than one byte is specified in this field, then each hex value must be separated by a comma or space, and complete entry be wrapped in quotation marks (e.g. "0x44,0x30"). If no start character(s) is/are required for the text that comes in, then leave it empty (see first example in table below). In this case, any characters except the stopbyte(s) will be recognized as part of the message. The startbytes of the Example2 message are "AB", so any text starting with "AB" will be recognized as event.	
<stopbyte>	The stop byte is the last byte being received. It can be part of the event or excluded, which is specified at the <use_stopbyte> parameter. Note that in the first example, text events have to be terminated by 0x0D only - not CRLF (0x0D 0x0A) - otherwise the 0x0A would be recognized as the first byte of the next event within this example. For text events terminated by 0x0D 0x0A, the regular text mode is recommended.	
<use_stopbyte>	Defines whether to include or exclude the stop byte/character from the text received on the serial line.	
	e	Exclude stopbyte) - the stopbyte itself is not part of the event data.
	i	Include stopbyte - the event data ends with the stopbyte.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.COMM.BINEVENT0=,0D,e $PFAL,Cnf.Set,DEVICE.COMM.BINEVENT1="0x41,0x42",43,i $PFAL,Cnf.Set,DEVICE.COMM.BINEVENT0="0x44,0x30",0D,e // for IDs (Tags) that start with D0 $PFAL,Cnf.Set,DEVICE.COMM.BINEVENT1=,0D,i // Send <CR> with the user text $PFAL,Cnf.Set,DEVICE.COMM.BINEVENT0=,0D,e //<CR> will be not sent with the user text $PFAL,Cnf.Set,DEVICE.COMM.BINEVENT1=,0A,i // Send <LF> with the user text $PFAL,Cnf.Set,DEVICE.COMM.BINEVENT1=,0A,e // <LF>will be not sent with the user text
Get Configuration	DEVICE.COMM.BINEVENT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	×
2	✓	✓	✓	✓

Notes

Special care has to be taken when clearing DEVICE.COMM.BINEVENT0 or DEVICE.COMM.BINEVENT1.

- In order to make use of the setting DEVICE.COMM.BINEVENT, the device has to be reset.
- At next power up, serial BIN-Event settings will be loaded from DEVICE.COMM.BINEVENT for Serial0 and Serial1 respectively.
- For textual purposes a stopbyte is usually a CR (0x0D) character, as shown in the first example above.
- The value of a single start/stop byte is always written as hexadecimal value and therefore doesn't need to start with "0x".

5.1.9. DEVICE.BAT.MODE

Parameter syntax	DEVICE.BAT.MODE=<mode>

This setting specifies the current battery mode (its behavior). It is recommended to set up battery mode by using the PFAL command Sys.Bat.Mode (which affects this setting).

Parameter	Value	Meaning
<mode>	AVL device can operate from a battery or an external power source and it is able to switch between the two power sources. This entry determines whether or not the AVL device will continue operating even if the external power source is removed. The default setting is set to "eco".	
	disabled	Battery is never used to power the device. The device runs only with external power. As soon as external power drops, the device stops working immediately.
	auto	Battery is enabled if no external power is applied. The device may also run with external power: ◆ if external power is higher than 9 V, the battery will be disabled. ◆ if external power drops below 8V, the battery will be enabled again. In case of no external power is available, the device runs as long as battery provides enough power. If battery power gets too low, a battery low power event occurs.
	always	The device uses the internal battery as power source regardless of external power. ◆ Disadvantages: Even when charging mode is set to AUTO, the battery can be never fully charged. Therefore operation time without external power is reduced using this feature. ◆ Advantages: The device uses less power from external power source until its internal battery is discharged to approx. 3,7V. This mode is best used if the internal battery is charged completely and the device comes in a situation in which it should use as less external power as possible.
	eco	(Default) It operates as auto mode, but internal battery is charged only if: ◆ Device has been started and the battery is not fully charged ◆ Device is running and the battery is below 3.9V. During regular device operation, battery is not charged in short cycles (which happens in auto mode when battery slightly drops below 4,15V). Charging is stopped as soon as the internal battery is fully charged, and it will be resumed when battery voltage drops below 3.9V.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.BAT.MODE=auto
Get Configuration	$PFAL,Cnf.Get,DEVICE.BAT.MODE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	✓

5.1.10. DEVICE.BAT.CHARGEMODE

Parameter syntax	DEVICE.BAT.CHARGEMODE=<charge_mode>

This setting specifies the operating mode of battery charger. It is recommended to set up battery mode by using the PFAL command Sys.Bat.ChargeMode (which affects this setting).

Parameter	Value	Meaning
<charge_mode>	Defines whether or not to charge the backup battery whenever there is a drop of voltage.	
	disabled	(default). Battery is never charged
	auto	The internal battery is charged if sufficient external power (of at least 9V) is available and temperature range for charging is not exceeded. Charging is stopped as soon as the internal battery is fully charged.
	eco	(Default) It operates as auto mode, but internal battery is charged only if: ◆ Device has been started and the battery is not fully charged ◆ Device is running and the battery is below 3.9V. ◆ During regular device operation, battery is not charged in short cycles (which happens in auto mode when battery slightly drops below 4,15V). ◆ Charging is stopped as soon as the internal battery is fully charged, and it will be resumed when battery voltage drops below 3.9V.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.BAT.CHARGEMODE=auto
Get Configuration	$PFAL,Cnf.Get,DEVICE.BAT.CHARGEMODE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	✓

Notes

In case of firmware update it is recommended to set this parameter to DEVICE.BAT.CHARGE-MODE=auto and then update the firmware as usual.

5.1.11. DEVICE.IGNTIMEOUT

Parameter syntax	DEVICE.IGNTIMEOUT=<time_out>

This setting defines a maximal wait time until entering sleep mode – if this time is expired (i.e. by attempting to send a TCP packet), the device clears all unsent messages and enters sleep mode immediately. This ensures that battery power isn't wasted in case the device has e.g. no GSM/ GPRS coverage.

<time_out>

This timeout defines the maximal duration in milliseconds after which the device enters sleep mode. When entering sleep mode using a PFAL,Sys.Device.Sleep command, the device tries to execute all alarms, sends away TCP messages, SMS etc. and then cancels communication via TCP, GPRS etc..

How the configuration could be set/requested:

Set configuration	$PFAL,Cnf.Set, DEVICE.IGNTIMEOUT=60000
Get configuration	$PFAL,Cnf.Get,DEVICE.IGNTIMEOUT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	✓

Notes

The <time_out> value is required by the system, only if the system has been configured to execute a shutdown process when a condition result is TRUE.

- Before the AVL device goes to sleep or resets, it calls the <time_out> value and set itself into the shutdown mode for the given period of time. Within this time the system tries to release all actions activated from the "Sys.Device.eShutdown" shutdown event. Once the timeout <time_out> is exceeded the device cancels all other processes, even if there are still actions to be released, and performs itself an “emergency” shutdown.
- If the <time_out> value is set to 0 (zero), the system performs immediately a shutdown process.

5.1.12. DEVICE.GSM.STARTUP

Parameter syntax	DEVICE.GSM.STARTUP=<control>

This parameter configuration allows the AVL device to control the startup behaviors of the integrated GSM engine.

Parameter	Value	Meaning
<control>	It determines whether or not the GSM engine will be enabled on the system startup. It can be set to:	
	on	Enables the GSM engine during startup. GSM services like SMS, voice/data calls GPRS (=> TCP) are available while the device is powered.
	off	GSM engine isn’t switched on during startup and remains powered off until it is enabled via PFAL command. Note that all GSM related PFAL commands won’t be executed or might result in error and state queries (including IMEI, SIMID OwnNumber etc.) won’t work if GSM engine isn’t properly initialized and running.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.GSM.STARTUP=on
Get Configuration	$PFAL,Cnf.Get,DEVICE.GSM.STARTUP

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

Note that all GSM related PFAL commands will not be executed or might result errors and state queries (including IMEI, SIMID OwnNumber etc.) will not work, if GSM engine is not properly initialized and running.

5.1.13. DEVICE.GPS.AUTOCORRECT

Parameter syntax	DEVICE.GPS.AUTOCORRECT=<type>,<max_pdop>,<max_speed>,<dist_error>,<dropcount>,[<max_acceleration>],[dynamic=<dynamic_tolerance>,<dynamic_interia>]

This parameter contains a number of specified conditions that can be used to control the use of the incoming GPS positions and calculate a valid GPS fix. The specified values have no influence on how the built-in GPS receiver actually computes its positions. AVL devices use your specified values to filter the incoming GPS position coordinates coming from the GPS receiver by changing those positions from active "A" to invalid "V" if they do not meet your specified condition. In the protocol, a value of "A" (for "active") indicates that the device has currently a GPS fix, whereas a value of "V" (for "inValid") indicates the device does not have currently a GPS fix.

Parameter	Value	Meaning
<type>	Determines whether or not to enable the GPS auto correction. It can be set to:	
	on	Enables filtering of GPS auto correction.
	off	(Default) Disables filtering of GPS auto correction. The followed values can be omitted.
<max_pdop>	It specifies the maximal allowed PDOP value to use incoming GPS positions (broken value may also be entered – e.g. 4.8). If the current PDOP value of an incoming GPS position exceeds your specified PDOP value, the AVL device will change the status of these GPS positions from active "A" to invalid "V".	
<max_speed>	It specifies the maximal speed limit in m/s (meter per second). Any speed exceeding this value will be ignored (i.e. no position will be computed from this GPS record).	
<dist_error>	It specifies the maximal allowed distance error. Former GPS records are analyzed. According to their speed, a distance is computed and compared to the distance of current and last (corrected) GPS record. The difference of both distances may not exceed.	
<dropcount>	This setting specifies how many "wrong" positions will be dropped until the current position is considered as inaccurate. Advantages: Several wrong positions (showing a correct DOP and a fix) during early startup can be eliminated in this way. Disadvantages: Whenever a wrong position is considered as correct, it takes <dropcount> seconds until this error will be detected.	
[<max_acceleration>]	This entry is optional, and it can be used to specify the maximum possible GPS speed that can change from one second to the next. For example, a <max_acceleration> value of 50 rejects a GPS speed of 100 m/s, if the prior speed value has been lower than 50 m/s. The default value is 50 m/s	

Parameter	Value	Meaning
dynamic=<dynamic_tolerance>,<dynamic_interia>]	<dynamic_tolerance>	It specifies the PDOP value to short PDOP changes. This tolerance is a relative value to the last received GPS position that is already considered as correct. Whenever a GPS position is considered as correct, then the next received position will be considered correct if it will have a PDOP value smaller than the last received position with a value PDOP + <dynamic_tolerance>. Example (refer to the example in table below): Maximum PDOP <max_pdop> is set to 10, <dropcount> to 10, dynamic tolerance <dynamic_tolerance> to 3 and dynamic inertia <dynamic_inertia> to 4. Let's suppose the last received GPS positions have had a PDOP value of 5 and those positions are considered as correct due to the specified <max_pdop>. The next incoming GPS positions show a reduced accuracy with a PDOP value of 9. A dynamic tolerance of 4 would allow these positions to be considered as correct. Because the dynamic tolerance is set to 3, these positions will be ignored. The <dynamic_interia> setting covers this case.
	<dynamic_interia>	This setting works together with the dynamic tolerance <dynamic_tolerance> and it specifies how many GPS positions should be ignored if the difference between the PDOP value of the previous and present position data is greater than your specified value <dynamic_tolerance>. When the specified number of positions is ignored, the next incoming GPS position will be considered as correct again, if it has a PDOP value that does not exceed your specified <max_pdop>. The value of the <dynamic_interia> ranges from 1 to <dropcount>-1 and always it must be smaller than <dropcount>.

Advantages and disadvantages of dynamic autocorrect:

If the PDOP values change very often from one position to the next received one, then the dynamic tolerance might reject most of these positions even when their PDOP value is lower than your specified <max_pdop>.

Let's see an example with following settings: dynamic_tolerance=3, dynamic_inertia=3 , max_dop=10;

Let's assume that the incoming positions from the GPS receiver have PDOP values as given in the tables below and all other Autocorrect conditions are meat.

Example 1									
Position Nr.	1	2	3	4	5	6	7	8	9
DOP values	2*	6	6	3*	7	7	2*	7	7

(* positions are considered as correct)

In the example 1, all positions that have a PDOP value 6 and 7 (not marked with asterisk [*] sign) are rejected by the system. That means: the second position is rejected because the PDOP difference between the first (already considered as correct) and second position is greater than the

specified value of the <dynamic_tolerance>=3, but the fourth position is considered as correct because it has a PDOP value smaller than previous (third) position and lower than <max_pdop>. If the 4th position should have had a PDOP value equal or greater than the previous rejected position then that position would be rejected too and the next one, due to the <dynamic_interia>value of 3, would be considered as correct again - as shown in the next example 2.

Example 2										
Position Nr.	1	2	3	4	5	6	7	8	9	10
DOP values	2*	6	6	6	6*	7*	5*	9	8*	8*

(* position considered as correct)

Due to the dynamic_inertia=3, the 5th position is considered as correct, and the followed positions 6 and 7 are also considered as correct because their PDOP value difference to the previous position is within the tolerance and smaller than the specified <max_pdop>.

If the PDOP values of the GPS positions increase generally (i.e. the device is entered into an area with poor GPS signal strength), these positions are also considered as correct.

How the configuration could be set/requested:

Set Configuration	PFAL,Cnf.Set,DEVICE.GPS.AUTOCORRECT=on,7.0,60,50,10 $PFAL,Cnf.Set,DEVICE.GPS.AUTOCORRECT=on,10.0,60,50,10,50,dynamic=3,4 $PFAL,Cnf.Set,DEVICE.GPS.AUTOCORRECT=off
Get Configuration	$PFAL,Cnf.Get,DEVICE.GPS.AUTOCORRECT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

This feature is still under test.

- No invalid RMC protocols will be displayed after the first fix. If GPS becomes invalid, the corresponding event will happen – also the current state can be retrieved. GPS protocols contain the last correct position, a current time but fix-state will be invalid as long as there are no correct GPS positions available.
- RMC protocols will always show the (most recent) correct GPS position (an invalid fix contains zeros during early startup).

5.1.14. DEVICE.GPS.CFG

Parameter syntax	DEVICE.GPS.CFG=<min_req_sat>[,<fast>][,<ColdStart>][,<static_nav>][,<sbas>][,<DR>]

This parameter allows to specify the number of satellites to consider a GPS fix as valid.

Parameter	Value	Meaning
<min_req_sat>	It defines the minimum number of satellites required for a valid GPS fix. By default it is set to 3. The default value allows the AVL device to get a GPS fix even in areas with very bad GPS coverage. However, it can be set to a value from 1 to 10. The higher the value, the higher is the GPS accuracy. Furthermore, it is strongly recommended to specify either 3 or 4 satellites depending on GPS coverage. In that case a protocol sent by the STEPPIII device is considered as "invalid" (depending on user set value) if it has less than 3 or 4 satellites in a sentence.	
<fast>	Optional fast startup mode for starting up in hot fix mode when a fast time to first fix is of high importance. It can either be set to fast or left empty. ◆ GPS fix possible within a few seconds after powering up the system (depends on status of GPS receiver). ◆ Tracking/location is possible even before system start event occurs. ◆ Automatically stores (valid) positions each 3 seconds until GPS start event occurs: ◆ Writes history positions (user text "FS-POS"). ◆ Adds a binary record to TCP storage (user text "FS-POS"). ◆ Uses stored last valid position in order to speed up the GPS TTFF (for devices with µ-blox only). ◆ Stores last valid position automatically before shutting down the system.	
<ColdStart>	Optional entry. It defines which type of start should be used when a GPS reset is executed (i.e. by timeout, command etc.).	
	0	Default setting. Disables a cold start initialization of the GPS receiver, allowing a faster time to first fix (with valid stored GPS information).
	1	Cause a cold start initialization of the GPS receiver, so that all information stored within SRAM will be deleted. This can help the GPS receiver to speed up the time to first fix (TTFT), when it has moved a long distance while being turned off.
[<static_nav>]	This option selects whether or not the static navigation should be enabled. Per default, static navigation is disabled.	
	0	Deactivates static navigation.
	1	Activates static navigation.
[<sbas>]	This option selects whether or not the SBAS operation should be enabled for more accurate fix. Per default, this option is enabled.	
	0	Deactivates SBAS.
	1	Activates automatic SBAS
[<DR>]	This option enables the use of Dead Reckoning function. Per default, this option is disabled. Combining multi-GNSS (GPS, GLONASS, BeiDou, Galileo) with the untethered dead-reckoning technology in our FOX3-3G-DR improves position accuracy even where GNSS signals are partial or completely blocked or reflected, such as in urban canyons, tunnels, mines, underpasses, multi-level roads or parking garages. Applications with untethered dead-reckoning include service vehicles from the airports, port facilities, car-sharing, fire departments that require accurate positioning at all times.	
	0	or empty, disables DR feature.
	1	Enables DR feature.

How the parameter could be sent/ and its setting requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.GPS.CFG=3,0,0,1 $PFAL,Cnf.Set,DEVICE.GPS.CFG=3,fast,0 //quick startup mode with coldstart disabled $PFAL,Cnf.Set,DEVICE.GPS.CFG=3,0 //coldstart disabled
Get Configuration	$PFAL,Cnf.Get,DEVICE.GPS.CFG

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.1.15. DEVICE.GPS.TIMEOUT

Parameter syntax	DEVICE.GPS.TIMEOUT=<enable>,<timeout>,[<hard_reset>]

This setting allows to specify a timeout after which the device performs a GPS reset if it has no valid fix. Such a restart forces the GPS engine to start a new search for visible GPS satellites, it can help to reacquire valid GPS positions in areas with very poor GPS coverage.

Parameter	Value	Meaning
<enable>	It defines whether to reset the GPS engine, when no GPS fix available. It can be set to:	
	0	Disabled (no reset is forced, GPS runs continuously).
	1	Enabled (forces a regular GPS reset after <timeout>
<timeout>	The timeout in minutes. It may range between 1 (2 for optional setting) and 45000 (> 1 month) (You can specify larger numbers too, but this probably won't make sense). The default timeout is set to 30 minutes.	
[<hard_reset>]	It is optional. It can be set to:	
	1	Enabled (forces a hard gps reset after <timeout>
	Using this hard reset, causes the GPS device to perform a complete restart. Under normal conditions it shouldn't be necessary to use this setting. It should be used if a device lost GPS and didn't regain a fix even after one or several <timeout>. If a system restart helped to regain a GPS fix, you should enable the "hard reset".	

This optional setting is still under test.

How the parameter could be sent/ and its setting requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.GPS.TIMEOUT=1,30 $PFAL,Cnf.Set,DEVICE.GPS.TIMEOUT=1,30,1
Get Configuration	$PFAL,Cnf.Get,DEVICE.GPS.TIMEOUT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.1.16. DEVICE.GPS.NAVDISTMUL

Parameter syntax	DEVICE.GPS.NAVDISTMUL=<factor>

This parameter is used as a multiplicator for NavDis feature to adapt it to the ODOmeter of the vehicle.

<factor>

It defines the multiplicator as decimal value for NavDis feature. <factor> 1000 means 100%. If you want to multiply the calculated NavDist by 2%, then set the factor to 1020.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.GPS.NAVDISTMUL=1020
Get Configuration	$PFAL,Cnf.Get,DEVICE.GPS.NAVDISTMUL

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.1.17. DEVICE.PFAL.SEND.FORMAT

Parameter syntax	DEVICE.PFAL.SEND.FORMAT=<start_txt>,<checksum>,<end_txt>,<end_msg>

This parameter allows to freely change/adapt the syntax of the messages that will be sent out with MSG.Send…., making them easier to quickly identify the parts of a message and to select them for further use. It allows to define the start and end characters for each protocol specified within the <protocols> and user text specified by <"text">. It affects the syntax of all messages that will be sent from the AVL device via Serial, CSD, TCP and SMS.

Parameter	Value	Meaning
<start_txt>	It defines the start characters for user text section (max. 10 chars). Each protocol specified by the <protocols> field gets the <start_txt> characters as its first character. Furthermore, the defined <"text"> by MSG.Send gets the <start_txt>, too. By default, this parameter is set to $-character. Keep in mind, the use of the $-character inside an SMS text might be misinterpreted. This depends on the receiving modem.	
<checksum>	By default, this parameter is set to CKSUM, However, it can be set to:	
	CKSUM	Adds a NMEA compatible checksum at the end of text section (=default). Note that, the checksum is computed for text only and not for a possibly defined <start_txt> before. If protocols are specified within the MSG.Send command then all protocols will be sent in the format, for example, $GPRMC....*CS including "$" and checksum "*CS"
	NOCKSUM	No checksum is added after user defined text. If protocols are specified within the MSG.Send command then all protocols will be sent in the format, for example, GPRMC.... excluding "$" and checksum "*CS". This format is suited for SMS communication service as some modems, expect the FOX3, could not receive correctly the $- dollar sign.

Parameter	Value	Meaning
<end_txt>	It defines the end characters for user text section (max. 10 chars). A CRLF will be added automatically at the end of the defined "end characters"	
<end-msg>	It defines the end characters to complete the message (max. 10 chars) (by default it is set to <end> allowing syntax compatibility to the PFAL responses). The CRLF will be added automatically at the end of the defined "end message"	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.PFAL.SEND.FORMAT="$",CKSUM,"","<end>"
Get Configuration	$PFAL,Cnf.Set,DEVICE.PFAL.SEND.FORMAT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

Depending on which report syntax you have selected, the messages that will be dispatched with MSG.Send…. command, including <protocols> and <"text”>, change their syntax as follow.

```
For example, you send or configure the AVL device to output its
position to the serial interface using the PFAL-Command:
$PFAL,MSG.Send.Serial,8,"AVL device outputs its GPS positions"
also you have selected the following format syntax:
$PFAL,Cnf.Set,DEVICE.PFAL.SEND.FORMAT="$",CKSUM,"","<end>"
The message that the system AVL device sends out looks in this order:
$AVL device outputs its GPS positions<CRLF>
$GPRMC....*21<CRLF>
<end><CRLF>
If you select the following format syntax:
$PFAL,Cnf.Set,DEVICE.PFAL.SEND.FORMAT="$",CKSUM,"&","<end>"
then the message that the system AVL device sends out looks in this
order:
$AVL device outputs its GPS positions&<CRLF>
$GPRMC.... *21&<CRLF>
<end><CRLF>
This command syntax which the user selects here, it is applied to all
protocols and user text defined by the command MSG.Send.…
```

5.1.18. DEVICE.CAN.OBD.STARTUP

Parameter syntax	DEVICE.CAN.OBD.STARTUP=<format>,<port>

This parameter saves and defines the frame format to be enabled for reading of OBDII messages on the specified CAN interface.

Parameter	Value	Meaning
<format>	Defines the frame format to be enabled for reading of OBDII messages on the specified CAN interface.	
	Std	Enables the 11-bit identifier (CAN2.0A).
	Ext	Enables the 29-bit identifier (CAN2.0B).
<port>	Specifies the CAN port to enable reading of OBDII messages.	
	0	Enables reading of defined frame format on 1st CAN interface (on main port).
	1	Enables reading of defined frame format on 2nd CANB interface (on IOBOX-CAN).

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.CAN.OBD.STARTUP=std,0 $PFAL,Cnf.Set,DEVICE.CAN.OBD.STARTUP=std,1 $PFAL,Cnf.Set,DEVICE.CAN.OBD.STARTUP=ext,1
Get Configuration	$PFAL,Cnf.Get,DEVICE.CAN.OBD.STARTUP

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	✖

5.1.19. DEVICE.CAN.FMS.STARTUP

Parameter syntax	DEVICE.CAN.FMS.STARTUP=<format>,<port>

This parameter saves and enables or disables reading of FMS messages on the specified CAN interface.

Parameter	Value	Meaning
<format>	Defines whether to enable or disable reading of FMS messages on the specified CAN interface.	
	off	Disable FMS functionality on the specified CAN interface.
	on	Enables FMS functionality on the specified CAN interface.
<port>	Specifies the CAN port to enable reading of FMS messages.	
	0	Enables reading of defined frame format on 1st CAN interface (on main port).
	1	Enables reading of defined frame format on 2nd CANB interface (on IOBOX-CAN).

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.CAN.FMS.STARTUP=on,0 $PFAL,Cnf.Set,DEVICE.CAN.FMS.STARTUP=off,1
Get Configuration	$PFAL,Cnf.Get,DEVICE.CAN.FMS.STARTUP

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.1.20. DEVICE.CAN.STARTUP

Parameter syntax	DEVICE.CAN.STARTUP<can_port>=<on_off>,<baud>,<mode>

This parameter saves and enables or disables reading of CAN messages on the specified CAN interface using the user specified baudrate.

Parameter	Value	Meaning
<can_port>	Defines the interface to activate reading of CAN messages.	
	0	Enables reading on 1st CAN interface (on main port).
	1	Enables reading on 2nd CANB interface (on IOBOX-CAN)
<on_off>	Defines whether to enable or disable that CAN port for reading of CAN messages.	
	off	Disable CAN functionality on the specified CAN port.
	on	Enables CAN functionality on the specified CAN port.
<baud>	Defines the baudrate settings of the specified CAN port. Note that, after changing the baudrate with activated CAN, to activate the new user-specified settings a device reset is required. Following values are available:	
	10K	CAN interface operates at 10 Kbits/s.
	20K	CAN interface operates at 20 Kbits/s.
	33K3	CAN interface operates at 33.3 Kbits/s.
	50K	CAN interface operates at 50 Kbits/s.
	83K3	CAN interface operates at 83.3 Kbits/s.
	95K2	CAN interface operates at 95.2 Kbits/s .
	100k	CAN interface operates at 100 Kbits/s .
	125K	CAN interface operates at 125 Kbits/s.
	250K	CAN interface operates at 250 Kbits/s.
	500K	AN interface operates at 500 Kbits/s.
	666K6	CAN interface operates at 666.6 Kbits/s .
	800K	CAN interface operates at 800 Kbits/s.
	1M	CAN interface operates at 1024 Kbits/s.

Parameter	Value	Meaning
<mode>		
	RO	Read Only mode (Silent mode). CANB interface only listens incoming CAN packets. It does not accept any packets or sends any data over the bus. This is the recommended setting, as it does not interfere with other communication at the bus.
	RW	Read Write mode (Running mode). CANB interface accepts received packets and can send CANB messages if requested. Note that the CAN message has to be acknowledged by the CAN bus, otherwise the device keeps repeating this message until an acknowledgement is received. This setting should be used with caution, as it influences and interferes with the communication at the connected CAN bus.
	LB	This mode is provided for self-test function. In Loop Back Mode, the interface treats its own transmitted messages as received messages (if they pass acceptance filtering).
	SLB	This mode is to simulate messages without having access to vehicle CAN-Bus. This mode can be used for a “Hot Selftest”, meaning the interface can be tested like in Loop Back mode but without affecting a running CAN system connected to the interface pins.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.CAN.STARTUP=on,0 $PFAL,Cnf.Set,DEVICE.CAN.STARTUP=off,1
Get Configuration	$PFAL,Cnf.Get,DEVICE.CAN.STARTUP

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.1.21. DEVICE.CAN.DTCOFMS.STARTUP

Parameter syntax	DEVICE.CAN.DTCOFMS.STARTUP=<mode>,<port>

This parameter saves and enables or disables reading of tachograph data via FMS interface on the specified CAN interface.

Parameter	Value	Meaning
<mode>	Defines whether to enable or disable reading of tachograph data via FMS interface.	
	off	Disable reading of tachograph data.
	on	Enables reading of tachograph data.
<port>	Specifies the CAN port to enable reading of FMS messages.	
	0	Enables reading of tachograph data on 1st CAN interface (on main port).
	1	Enables reading of tachograph data on 2nd CANB interface (on IOBOX-CAN)

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.CAN.DTCOFMS.STARTUP=on,0 $PFAL,Cnf.Set,DEVICE.CAN.DTCOFMS.STARTUP=off,1
Get Configuration	$PFAL,Cnf.Get,DEVICE.CAN.DTCOFMS.STARTUP

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.1.22. DEVICE.CAN.ERR.EVENTS

Parameter syntax	DEVICE.CAN.ERR.EVENTS=<mode>

This parameter saves and enables or disables error CAN messages on the both CAN interfaces.

Parameter	Value	Meaning
<on_off>	Defines whether to enable or disable error CAN messages.	
	off	Disables error CAN messages.
	on	Enables error CAN messages.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.CAN.ERR.EVENTS=on
Get Configuration	$PFAL,Cnf.Get,DEVICE.CAN.ERR.EVENTS

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.1.23. DEVICE.CAN.EVENT_<slot>

Parameter syntax	DEVICE.CAN.EVENT_<slot>=<standard>,<msg_id>,<event_cnt>[,eventX>]

This parameter can be used to generate events out of changed CAN variables without checking them periodically for changes. AVL devices provide the possibility to analyze 10 slots allowing 10 different CAN messages to be analyzed at the same time. CAN Messages will be checked for changes approx. every 128 ms (8 times per second).

Parameter	Value	Meaning
<slot>	It specifies the CAN event slot index, in the range from 0 to 9.	

Parameter	Value	Meaning
<standard>	It specifies the CAN message standard. Following protocols are supported by Lantronix AVL devices:	
	OBDII	It is a generic term referring to the self-diagnostic and reporting capability of a vehicle.
	FMS	It is a standard interface to vehicle data of commercial vehicles. The amount of data is dependent on the manufacturer and model of the vehicle and might be different.
	J1939	It is the vehicle bus standard used for communication and diagnostics among vehicle components, originally by the car and heavy duty truck industry.
<msg_id>	It defines the protocol (message) ID of corresponding CAN message standard. Format depends on chosen standard value: If OBD-II is specified in the <standard> entry, then specify here a hexadecimal number of desired OBD message (i.e. C = RPM). If FMS or J1939 is specified in the <standard> entry, then specify here a variable name of specific Message. i.e. BRAKE_SWITCH. Names of implemented FMS/J1939 variables can be found within dynamic protocol table - see chapter 7. Note that J1939 is only partially supported: J1939 is supported for all periodic messages. Depending on the used CAN interface, some J1939 messages might not be sent periodically but have to be requested. These request packets are currently not sent automatically (but can be sent by configuring an alarm). Due to CAN message licensing restrictions, no support can be given for command examples or command syntax lists).	
<event_cnt>	Number of events configured for this CAN message (several events can be configured for one CAN message - i.e. various ranges of RPM speeds etc..{<eventX>} Optional. It defines the list of events (number must match <event_cnt>). Configuration of the specific event. Following syntaxes can be used (refer to the table below): <eventX>=<s_type><event_edge><s_info_id> or <eventX>=<s_type><s_info_id>,<rst_Lmax>,<minval>,<maxval>,<rst_Hmin>	

Parameter	Value	Meaning	
<s_type>	Sub-index type. Variable type and possible values:		
	V	Value	
	B	Bit (value 0 or 1)	
	Value	**Sub-entry**	**Sub-Value and Description**
	If <s_type>=**B** then:	1st Syntax: <eventX>=<s_type><event_edge><s_info_id>	
		<event_edge>	**H** Bit change from 0->1 causes an event **L** Bit change from 1->0 causes the event
		<s_info_id>	Sub-index ID (ID of info within CAN message). If a single message contains just one piece of information, this index is 0. However, a message may contain several units of information. Within the FMS/OBDII/J1939 specification they are listed. <s_info_id> identifies which information is used to create the event . Example: **BH2** Bit change 0->1 within 3rd information inside configured message causes an event. **BL0** Bit change 1->0 within 1st information inside configured message causes an event.
		Example:	<eventX>=BH2
	If <s_type>=**V** then:	2nd Syntax: <eventX>=<s_type><s_info_id>,<rst_Lmax>,<minval>,<maxval>,<rst_Hmin>	
	<s_info_id>	Sub-index ID (ID of info within CAN message). A single message may contain several pieces of information. Within the FMS/OBDII/J1939 specification they are listed. <s_info_id> identifies which information is used to create the event (i.e. Event-Ranges of the specified CAN information: <rst_Lmax>,<minval>,<maxval>,<rst_Hmin>)	
	<rst_Lmax>	Maximum value which resets the event condition (-> after reaching this value or a lower value, the event can be triggered again).	
	<minval>	Minimum value which triggers the event.	
	<maxval>	Maximum value which can still trigger the event.	
	<rst_Hmin>	Minimum value which resets the event condition (-> after reaching this value or a higher value, the event can be triggered again).	
	Example:	<eventX>=V0,990,1000,2000,2010	
	Info: Using these parameters it is possible to define a kind of hysteresis. **For example engine RPM:** an event can be defined which is launched between 1000 and 2000 RPM. In order to prevent events arising very often when RPM changes around 1995 and 2005 RPM, it is possible to set minimum and maximum levels which have to be exceeded in order to cause another event of the same type. For this scenario ranges could be defined as: <rst_Lmax>: 990; <maxval>: 2000; <minval>: 1000; <rst_Hmin>: 2010		

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.CAN.EVENT_0=OBDII.C,1,V0,-1,0,1387,1388 //generates single event when OBDII-RPM is between 0 and 1387 (4 x rpm) $PFAL,Cnf.Set,DEVICE.CAN.EVENT_1=FMS.BREAK_SWITCH,2,BL0,BH1 //generates 2 events when break switch state changes (pressed & released)
Get Configuration	$PFAL,Cnf.Set,DEVICE.CAN.EVENT_1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.1.24. DEVICE.DTCO.D8

Parameter syntax	DEVICE.DTCO.D8=B0,<format> // enables the D8 input DEVICE.DTCO.D8=off // disables the D8 input

This parameter enables or disables the IN4 (PIN 8) on the Lantronix IOBOX-CAN accessory device to use it either for tachograph functionality or digital input purposes and specifies the format of the data to be received from the D8 interface.

The pin 8 of the connector “D” on the back of the digital tachograph delivers serial data in the several proprietary formats, e.g. VDO and Stoneridge. The IN4 (PIN 8) on the Lantronix IOBOX-CAN accessory device must be connected to D8 pin of the tachograph. See chapter *4.4.1.* to identify this on the 16pin connector of the IOBOX-CAN.

Parameter	Value	Meaning
<format>	Defines the format of the data to be received.	
	VDO	VDO data format.
	SRE	Stoneridge data format.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.DTCO.D8=B0,vdo $PFAL,Cnf.Set,DEVICE.DTCO.D8=off
Get Configuration	$PFAL,Cnf.Get,DEVICE.DTCO.D8

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.1.25. DEVICE.LOWPOWER

Parameter syntax	DEVICE.LOWPOWER=<mode>,<on_time>,<off_time>

This parameter sets the device into a low power mode.

Parameter	Value	Meaning
<mode>	This setting defines the power mode in the device.	
	0	Device remains always in full power operation.
	1	Device switches on power save mode periodically every user-specified time if no event occurs within this time.
	2	System enters doze mode whenever the defined on-time expires. This mode means: processor and GPS receiver go sleeping and GSM keeps running in stand-by state.
<"on_time">	Defines the amount of time, in seconds, that elapses before the system enters the auto or doze mode.	
<"off_time">	Define the amount of time, in seconds, the system will stay sleeping before it wakes up from the auto or doze mode.	

How the configuration could be set/requested:

Set Configuration	$PFAL,CNF.Set,DEVICE.LOWPOWER=0,0,0 //disabled $PFAL,CNF.Set,DEVICE.LOWPOWER=1,5,10 //"auto" mode $PFAL,CNF.Set,DEVICE.LOWPOWER=2,10,20 //"doze" mode
Get Configuration	$PFAL,Cnf.Get,DEVICE.LOWPOWER

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.1.26. DEVICE.GPS.JAMMINGLEVEL=<min>,<max>

Parameter syntax	DEVICE.GPS.JAMMINGLEVEL=<min>,<max>

This parameter uses noise profile to detect abnormalities during operation e.g. the possible presence of GPS jammers, interference, noise, jamming, after re-acquisition, wake-up of the inside GPS-Receiver. This can be used, for example of vehicle theft. If jamming is detected the corresponding event occurs. You can use this event to activate an alarm or alert signal and send the values of Cell-ID and LAC (local area code) with dynamic variables &(CellID) and &(LAC) as a backup positioning based on GSM cellular positioning.

Parameter	Value	Meaning
<min>	Defines the minimum jamming/interference level (unit = dB).	
	0...255	0 = no continuous wave jamming, 255 = strong continuous wave jamming.
<max>	Defines the maximum jamming/interference level (unit = dB).	
	0...255	0 = no continuous wave jamming, 255 = strong continuous wave jamming.

How the configuration could be set/requested:

Set Configuration	$PFAL,CNF.Set,DEVICE.GPS.JAMMINGLEVEL=6,200
Get Configuration	$PFAL,Cnf.Get,DEVICE.GPS.JAMMINGLEVEL

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.1.27. DEVICE.WLAN.STARTUP

Parameter syntax	DEVICE.WLAN.STARTUP= <state>,<auto_reconnect>,<tcp_timeout>

This parameter is used to set up the general configuration of the IOBOX-WLAN.

Parameter	Value	Meaning
<state>	Defines the state of the IOBOX-WLAN. Following values can be used:	
	On	The module stays on and ready to scan for new WLAN networks.
	Off	The module shuts down after initialization.
<auto_reconnect>	Defines whether to enable or disable the auto-reconnect to the WLAN network	
	0	Disables auto-reconnect.
	60000	Enables auto-reconnect. The amount of time to wait before attempting to reconnect.
<tcp_timeout>	Defines whether to enable or disable the auto-reconnect to the remote server.	
	0	Disables auto-reconnect.
	60000	Enables auto-reconnect. The amount of time to wait before attempting to reconnect to the server.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.WLAN.STARTUP=on,60000,60000
Get Configuration	$PFAL,Cnf.Get,DEVICE.WLAN.STARTUP

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.1.28. DEVICE.NFC

Parameter syntax	DEVICE.NFC=<port>
	DEVICE.NFC=Serial0

This parameter enables or disables connection to a Lantronix NFC reader on a serial port.

Parameter	Value	Meaning
<port>	Defines the format of the data to be received.	
	Off	Deactivates connection to an Lantronix NFC reader.
	Serial0	Activates connection to an Lantronix NFC reader on serial port on 8pin connector.
	Serial1[1]	Activates connection to an Lantronix NFC reader on serial port on 6pin connector.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.NFC=Serial0
Get Configuration	$PFAL,Cnf.Get,DEVICE.NFC

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	?
1	?	?	?	✗

5.1.29. DEVICE.VIN

Parameter syntax	DEVICE.VIN=<VIN>

This parameter saves the 17-character vehicle identification number connection and reports it using dynamic entry &(VIN).

<VIN>

Defines the 17-character vehicle identification number of your car.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.VIN=12232352352355678
Get Configuration	$PFAL,Cnf.Get,DEVICE.VIN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.1.30. DEVICE.SERIAL1.MODE485

Parameter syntax	DEVICE.SERIAL1.MODE485=<on/off>

This setting enables, disables the RS485 mode of the Serial1 interface.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DEVICE.SERIAL1.MODE485=on $PFAL,Cnf.Set,DEVICE.SERIAL1.MODE485=off
Get Configuration	$PFAL,Cnf.Get,DEVICE.SERIAL1.MODE485

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✗

5.2. REPLACE

5.2.1. REPLACE<index>

Parameter syntax	REPLACE<index>=<replace_string>

Additionally replacement marks can be set within any AL<alarms>. Each replacement mark belongs to a replacement configuration setting. The mark will be replaced automatically with its contents when an alarm is configured.

This is useful to e.g. store phone numbers within replacement configuration settings. Instead of using this number within an alarm, you can simply add the proper replacement mark. Doing so allows you to quickly change the replacement configuration later – that changes all occurrences within your alarm configuration. So you don't have to search and replace the whole alarm configuration anymore.

Of course this comes in handy only if you use e.g. a phone number very often within the alarm configuration. Changing this number now requires just the change of one replacement configuration – not each configured alarm AL.

Example:

```
For example, instead of writing alarm such as the following:
AL0=IO.e0=redge:GSM.SMS.Send,"+1234567",8,"I00 was switched on"
AL1=IO.e1=redge:GSM.SMS.Send,"+1234567",8,"I01 was switched on"
AL2=IO.e2=redge:GSM.SMS.Send,"+1234567",8,"I02 was switched on"
Your alarm would look as follows:
AL0=IO.e0=redge:GSM.SMS.Send,"(REPLACE0)",8,"I00 was switched on"
AL1=IO.e1=redge:GSM.SMS.Send,"(REPLACE0)",8,"I01 was switched on"
AL2=IO.e2=redge:GSM.SMS.Send,"(REPLACE0)",8,"I02 was switched on"
REPLACE0=+1234567// the phone number to which all SMS have to be
sent to.
```

Now, let's assume you want to change this phone number. Without replacement marks, you would have to enter this number within each alarm è you would have to change 3 alarms. Using replacement marks, you can simply change the REPLACE0 setting – e.g. from REPLACE0=+1234567 to REPLACE0=+7654321. After next restart of the device, all alarm settings will be using this new number.

Parameter	Value	Meaning
<index>	It specifies the replace index. It is a number ranging from 0 through 9 (including 10 different replacements). These settings contain text that is inserted into alarm configuration settings (AL<x>=....) at any occurrence of (REPLACE<x>).	
<replace_string>	It specifies the text to be replaced – with a maximum length of 200 characters .	

Set Configuration	$PFAL,Cnf.Set,REPLACE0=+1234567
Get Configuration	$PFAL,Cnf.Get, REPLACE0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

- When changing the replace configuration, a system restart is necessary to update all alarms.
- If you configure new alarms after you set up the replacements, these new alarms will already use the new settings – so there is no need to restart in order to update the values.
- Replacement marks may be placed JUST within alarm configurations (i.e. each configuration which starts with "AL<x>="

5.3. USER

5.3.1. USER<index>=<user_text>

Parameter syntax	USER<index>=<user_text>

This parameter allows user-specific settings to be stored in the configuration, e.g. version of the configuration, special device settings and so on.

Parameter	Value	Meaning
<index>	It specifies the index for the specified user text. It can be set to:	
	1	First user text.
	2	Second user text.
	3	Third user text.
	4	Fourth user text.
	5	Fifth user text.
	6	Sixth user text.
	7	Seventh user text.
	8	Eighth user text.
	9	Ninth user text.
<user_text>	It specifies the user text.	

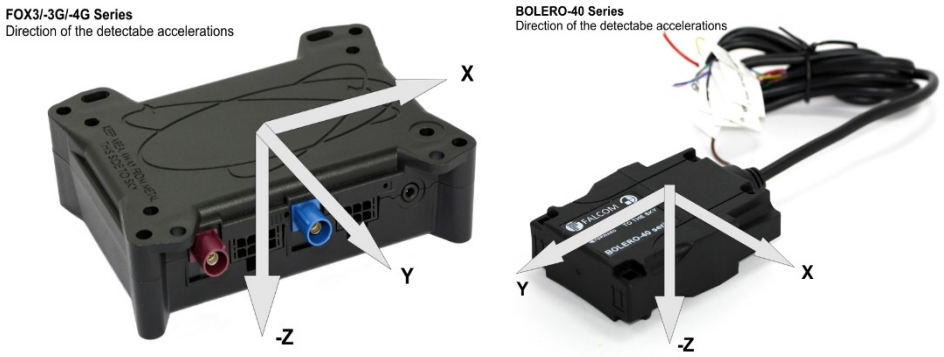
How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,USER1=Config_2.31.1
Get Configuration	$PFAL,Cnf.Get, USER1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.4. MOTION

Information about axis details/mounting directions can be found in the application note, ***App Note: In-Vehicle Installation Guidelines for FOX3 and BOLERO40 Series.*** See *1.3. Related documents*.



5.4.1. MOTION.FILTER

Parameter syntax	MOTION.FILTER=<a_coe>,<g_coe>,<max_dforce>,<min_dforce>

This parameter defines the configuration settings for filtering conditions of device motions. Based on the filter values, system raises events when device moves and device stops – use figure below as a reference. This configuration settings will also be used to wake up the AVL device from the Motion sleep mode.

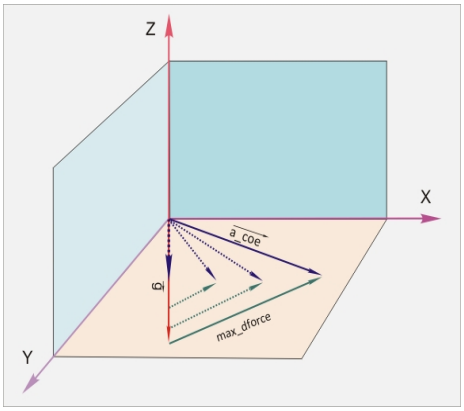
Parameter	Value	Meaning
<a_coe>	Set a value in a range of 1...20 which determines the sensitivity on which an average acceleration vector will be calculated each second. The lower the set value, the more sensitive is the motion of the device.	
<g_coe>	Set a value in a range of 20...60 that determines the sensitivity on which an average gravitation vector will be calculated each second. The lower the set value, the more sensitive is the motion of the device.	
<max_dforce>	Set a maximum difference value between the gravitation vector and acceleration vector, force in mg smaller (<) 8000, that will be accepted as device moving (the set value is used to detect whether the device is moving). Any other value greater than the set value counts as a disagreement (system raises the moving event [IO.Motion.eMoving]). ***Note:*** *A high value reduces the sensitivity of motion detection. In contrast a very low value may detect undesired "motion" caused by smallest vibrations of the device.*	
<min_dforce>	Set a minimum value, ranging from 39 ... <max_dforce>, that will be accepted as device standing (the value is used to detect whether the device is standing). Any other value less than the minimum set value counts as a disagreement (system raises the standing event [IO.Motion.eStanding]). ***Note:*** *A higher value detects a standing condition earlier, with the possibility of misinterpreting „low g-force impacts" as a standing condition (it could also be a very smooth deceleration (rolling out the car until it stands). In contrast: a very low value may prevent a standing condition due to a continuous small force applied to the device (i.e. motor vibrations or simply the noise of the sensor itself).*	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,MOTION.FILTER=4,20,312,273 // Supported values in the firmware version avl_2.12.0. To get the new values from the old firmware version multiply the old last two values by 39. $PFAL,Cnf.Set,MOTION.FILTER=2,20,8,7 // Supported values in the prior firmware versions
Get Configuration	$PFAL,Cnf.Get,MOTION.FILTER

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Figure 5-1 Filtering in motion



5.4.2. MOTION.BEARING

Parameter syntax	MOTION.BEARING<index>=<min_x>,<min_y>,<min_z>,<max_x>,<max_y>,<max_z>,<timeout>

This parameter is used to define the configuration settings to filter bearings of device motions. Based on the filter values, system raises the event IO.Bearing.e<> when the device exceeds the specified min. and max. values. In order to define the correct values for the device motion you have to test the AVL locally and find out the values that your application requires. The current bearing values can be read out via the serial communication ports or USB by activating the (protocol) with $PFAL,Cnf.Set,PROT.3DP=1 for examples:

```
$GP3DP,-52,-62,1100,-60,-68,1096,-48,-52,1108,3*05
$GP3DP,-72,971,-51,-80,960,-68,-68,976,-36,107*32
```

The 3DP protocol represents in decimal values the position of the device for all axes. Negative values have a minus (-) sign in front of that value (e.g. -52). To define the correct values into the settings of this parameter and generate the correct bearing events, just turn/rotate the device in that position your application requires and copy the values from the $GP3DP output and past them into corresponding min, max values below and send it to the device.

Parameter	Value	Meaning
<index>	Specify the index for storing the motion bearings values in range from 0 to 4.	
<min_x>	Specify the minimum value in mG (1G/1000) in range from -5000 to +5000 applied to the x-axis. Whenever the thresholds are exceeded (current value is greater than specified maximum OR current value is smaller than specified minimum), a GPEVENT:IO.Bearing.eX event is triggered.	
<min_y>	Specify the minimum value in mG (1G/1000) in range from -5000 to +5000 applied to the y-axis. Whenever the thresholds are exceeded (current value is greater than specified maximum OR current value is smaller than specified minimum), a GPEVENT:IO.Bearing.eX event is triggered.	

Parameter	Value	Meaning
<min_z>	Specify the minimum value in mG (1G/1000) in range from -5000 to +5000 applied to the z-axis. Whenever the thresholds are exceeded (current value is greater than specified maximum OR current value is smaller than specified minimum), a GPEVENT:IO.Bearing.eX event is triggered.	
<max_x>	Specify the maximum value in mG (1G/1000) in range from -5000 to +5000 applied to the z-axis. Whenever the thresholds are exceeded (current value is greater than specified maximum OR current value is smaller than specified minimum), a GPEVENT:IO.Bearing.eX event is triggered.	
<max_y>	Specify the maximum value in mG (1G/1000) in range from -5000 to +5000 applied to the z-axis. Whenever the thresholds are exceeded (current value is greater than specified maximum OR current value is smaller than specified minimum), a GPEVENT:IO.Bearing.eX event is triggered.	
<max_z>	Specify the maximum value in mG (1G/1000) in range from -5000 to +5000 applied to the z-axis. Whenever the thresholds are exceeded (current value is greater than specified maximum OR current value is smaller than specified minimum), a GPEVENT:IO.Bearing.eX event is triggered.	
<timeout>	Define the period of time on which the device checks the user-defined values to generate an event if they are fallen below or exceeded. The examples in table below are given for turn angles of approx. 45° and 90°. The threshold must be constantly exceeded during the defined delay (i.e. 5 seconds) in order to trigger the event.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,MOTION.BEARING0=-500,-5000,-5000,5000,5000,5000,1 // tilted right by (x = + 45°) $PFAL,Cnf.Set,MOTION.BEARING1=-5000,-500,-5000,5000,5000,5000,1 // tilted backwards by (y = -45°) $PFAL,Cnf.Set,MOTION.BEARING2=-5000,-5000,-5000,500,5000,5000,1 // tilted left by (x = -45°) $PFAL,Cnf.Set,MOTION.BEARING3=-5000,-5000,-500,500,5000,5000,1 // normal position (Z-Axis points downwards) (z = 180°) $PFAL,Cnf.Set,MOTION.BEARING4=-5000,-5000,-5000,5000,5000,500,1 // back position (Z-Axis points upwards with device label upwards) (Z = -180°)
Get Configuration	$PFAL,Cnf.Get,MOTION.BEARING0

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.4.3. MOTION.FORCE

Parameter syntax	MOTION.FORCE=<on_off>,<force_level>,<XYZ>

This parameter can be used to raise the Force events whenever one of the specified axis of the attitude sensor exceeds the specified acceleration. Each axis can be enabled or disabled separately, which allows to detect a force in a specific direction.

Parameter	Value	Meaning
<on_off>	Enables or disables the force events. It can be set to:	
	on	Enables the force events.
	off	Default. Disables the force events. If it is disabled then further parameters are optional and so they can be omitted.
<force_level>	Set a value in a range of 0 ... +8000 (1000=1g) that determines the level to occur the force event. If the devices detects that the force level is exceeded by one or several enabled axis then the device occurs the event IO.Motion.eForce.	
<XYZ>	Set the axis that will be monitored to enable occurring of the force event. To enable occurring of the force event on three-axes then set this entry to xyz or XYZ (case-insensitive). If you want to enable, for example, only the force event for the x-axis set this entry to x and to enable the event only for x-axis and z-axis then set this entry to xz. Note that, the order xyz must be considered.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,MOTION.FORCE=on,1500,xy
Get Configuration	$PFAL,Cnf.Get,MOTION.FORCE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.4.4. MOTION.3DFORCE

Parameter syntax	MOTION.3DFORCE=<-X>,<+X>,<-Y>,<+Y>,<-Z>,<+Z>

This parameter can be used to raise Force events whenever one of the specified axis direction of the attitude sensor exceeds the specified acceleration. Each axis direction can be enabled or disabled separately, which allows to detect a force in a specific direction. Information about axis details/mounting directions can be found in the application note, ***App Note: In-Vehicle Installation Guidelines for FOX3 and BOLERO40 Series.*** See *1.3. Related documents*.

Parameter	Value	Meaning
<-X>	nables or disables the X- direction of the 3DForce event. It can be set to:	
	0 ... 8000	Decimal value in milli-G, defining the threshold causing a 3DForce event.
	off	Disables the -X direction of the 3DForce event.
<+X>	Enables or disables the X+ direction of the 3DForce event. It can be set to:	
	0 ... 8000	Decimal value in milli-G, defining the threshold causing a 3DForce event.
	off	Disables the +X direction of the 3DForce event.

Parameter	Value	Meaning
<-Y>	Enables or disables the Y- direction of the 3DForce event. It can be set to:	
	0 ... 8000	Decimal value in milli-G, defining the threshold causing a 3DForce event.
	off	Disables the -Y direction of the 3DForce event.
<+Y>	Enables or disables the Y+ direction of the 3DForce event. It can be set to:	
	0 ... 8000	Decimal value in milli-G, defining the threshold causing a 3DForce event.
	off	off
<-Z>	Enables or disables the Z- direction of the 3DForce event. It can be set to:	
	0 ... 8000	Decimal value in milli-G, defining the threshold causing a 3DForce event.
	off	Disables the -Z direction of the 3DForce event.
<+Z>	Enables or disables the +Z direction of the 3DForce event. It can be set to:	
	0 ... 8000	Decimal value in milli-G, defining the threshold causing a 3DForce event.
	off	Disables the +Z direction of the 3DForce event.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,MOTION.3DFORCE=off,1500,2500,off,off,2000
Get Configuration	$PFAL,Cnf.Get,MOTION.3DFORCE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.5. ALIAS

5.5.1. ALIAS.<type>

Parameter syntax	ALIAS.<type_index_subindex>=<alias_name>

This parameter allows you to define an alias name for each command and use it further in the configuration with the defined alias name instead of its default name.

Parameter	Value	Meaning
<type_index_subindex>	Specifies the command to be otherwise called.	
<alias_name>	Specifies the alias name which will correspond to the last <type_index_subindex> specified command.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,ALIAS.SYS=System $PFAL,Cnf.Set,ALIAS.CNF=Config $PFAL,CNF.Set,ALIAS.IO11=_GSM $PFAL,CNF.Set,ALIAS.IO12=_STATUS $PFAL,CNF.Set,ALIAS.IO13=_GPS

Get Configuration	$PFAL,Cnf.Get,ALIAS.SYS $PFAL,Cnf.Get,ALIAS.CNF $PFAL,CNF.Get,ALIAS.IO11 $PFAL,CNF.Get,ALIAS.IO12 $PFAL,CNF.Get,ALIAS.IO13

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

The "**$PFAL,Cnf.Set,ALIAS.IO11=_GSM**" input message sent to the AVL device does not mean that the "**IO11**" is equal to "**_GSM**", but the alias name for index "11" is "**_GSM**". An example how to declare alias names if they are already defined.

`$PFAL,CNF.Set,AL1=SYS.TIMER.e2&GSM.sOpInvalid:IO_GSM.Set=hpulse,500`

5.6. DBG

5.6.1. DBG.EN

Parameter syntax	DBG.EN=<enable>,<interface>

It enables a special debugging mode that displays the data being sent to the AVL device and responses received. The integrated debugger which helps you to track down the initialization/ execution of the firmware/ your application, to monitor the specified values and settings and runtime errors, etc. can be monitored from a terminal program connected to the AVL device. This configuration parameter is managed by the system rather than by the user. This parameter can be used only to read out the validity of the system debugging.

Parameter	Value	Meaning
<enable>	It specifies the value of the system-debugging mode. It can be set to:	
	0	Disables debugging mode.
	1[1]	Enables debugging mode.
<interface>	It is optional. It specifies the interface for outputting debug messages. If omitted the debug messages are outputted on the serial0.	
	Serial0	Fist serial port (default)
	Serial1[1]	Second serial port
	TCP	TCP interface

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,DBG.EN=1
Get Configuration	$PFAL,Cnf.Get,DBG.EN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
1	✓	✓	✓	×

5.7. PROT

The configuration of the protocols in this chapter is intended to provide real time navigation data to the host device. The AVL device uses the satellite signals to calculate its exact current location by calculating its distance from the satellites. The position data within the device is then converted into latitude and longitude coordinates, that are usually provided in the geodetic datum on which the GPS is based (WGS84) and the configured protocols are transmitted via serial interface to the connected host device (PC, laptop).

This configuration is intended only for the Serial Interface and it does **NOT** match the configuration of protocols to be transferred via SMS, CSD and TCP applications. To transfer/receive GPS data via SMS, CSD or TCP, please, refer to chapter *4.5.1.*

In addition to the configuration protocol, if a last valid position has been already saved from the prior operation and there is no GPS fix currently available, for example during system startup procedure, then the last valid position will be shown in the RMC and GGA output protocols (with state: invalid).

Whenever the GPS fix gets lost, the last valid position is still displayed (just the state changes to invalid), which works even if a last valid position has not yet been saved.

Note: *The saved last valid position is displayed during system startup, if the AVL device resides within an area without GPS coverage or whenever the GPS fix fails.*

5.7.1. PROT.<message_id>

Parameter syntax	PROT.<message_id>=<value>

This parameter not only allows certain messages to be enabled or disabled but also specifies the rate at which they are sent to the serial interface.

Parameter	Value	Meaning
<message_id>	Specifies the message identifier to be enabled or disabled. Following are the identifiers corresponding to the message supported by the BOLERO device software.	
<value>	Specifies the rate at which the selected message is sent to the serial interface.	
	1 to **255**	Enables the selected message.
	0	Disables the selected message.

Protocol	Meaning
GGA	Enables or disables the $ message (GPS NMEA)
GSA	Enables or disables the $ message (GPS NMEA
GSV	Enables or disables the $ message (GPS NMEA)
RMC	Enables or disables the $ message (GPS NMEA)
GLL	Enables or disables the $ message (GPS NMEA)
VTG	Enables or disables the $ message (GPS NMEA)
GLGSA	Enables or disables the (GLONASS)

Protocol	Meaning
GLGSV	Enables or disables the (GLONASS)
IOP	Enables or disables the message (Lantronix)
GSM	Enables or disables the message (Lantronix)
AREA	Enables or disables the message (Lantronix)
3DP	Enables or disables the $GP3DP message (Lantronix)
BIN	Enables or disables the message (Lantronix)

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,PROT.GGA=1 $PFAL,Cnf.Set,PROT.GSA=1 $PFAL,Cnf.Set,PROT.GSV=1 $PFAL,Cnf.Set,PROT.GLGSA=1 $PFAL,Cnf.Set,PROT.GLGSV=1 $PFAL,Cnf.Set,PROT.RMC=1 $PFAL,Cnf.Set,PROT.GLL=1 $PFAL,Cnf.Set,PROT.VTG=1 $PFAL,Cnf.Set,PROT.IOP=1 $PFAL,Cnf.Set,PROT.GSM=1 $PFAL,Cnf.Set,PROT.AREA=1 $PFAL,Cnf.Set,PROT.3DP=1 $PFAL,Cnf.Set,PROT.BIN=1
Get Configuration	$PFAL,Cnf.Get,PROT.GGA

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.7.2. PROT.START.BIN

Parameter syntax	PROT.START.BIN=$<value>

It allows you to specify your own protocol. It consists of the user specified text which will be attached at begin of the BIN protocol.

Parameter	Value	Meaning
<value>	String type. It consists of 18 Bytes. If it is specified, the value will be displayed at begin of the BIN protocol, including the dollar sign "$".	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,PROT.START.BIN=$user protocol
Get Configuration	$PFAL,Cnf.Get,PROT.START.BIN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.7.3. PROT.ERR

Parameter syntax	PROT.ERR=<"value ">

It allows you to define a value as a string when a read command returns error (currently for CAN FMS only)

Parameter	Value	Meaning
<value>	Enclosed in parentheses, it specifies your own text. This text will be added to the corresponding dynamic variable of the read command. By default it is set to “err”.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,PROT.ERR="err"
Get Configuration	$PFAL,Cnf.Get,PROT.ERR

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.7.4. PROT.NA

Parameter syntax	PROT.NA=<"value ">

It allows you to define a value as a string when the source of the value is not available (currently for CAN FMS only)

Parameter	Value	Meaning
<value>	Enclosed in parentheses, it specifies your own text. This text will be added to the corresponding dynamic variable of the read command. By default it is set to “n/a”.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,PROT.NA="n/a""
Get Configuration	$PFAL,Cnf.Get, PROT.NA

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.7.5. PROT.EMPTY

Parameter syntax	PROT.EMPTY=<"value ">

It allows you to define a value as a string when the read variable is empty yet (currently for CAN FMS only)

Parameter	Value	Meaning
<value>	Enclosed in parentheses, it specifies your own text. This text will be added to the corresponding dynamic variable of the read command. By default it is set to “n/a”.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,PROT.EMPTY="n/a"
Get Configuration	$PFAL,Cnf.Get,PROT.EMPTY

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.8. ECODRIVE

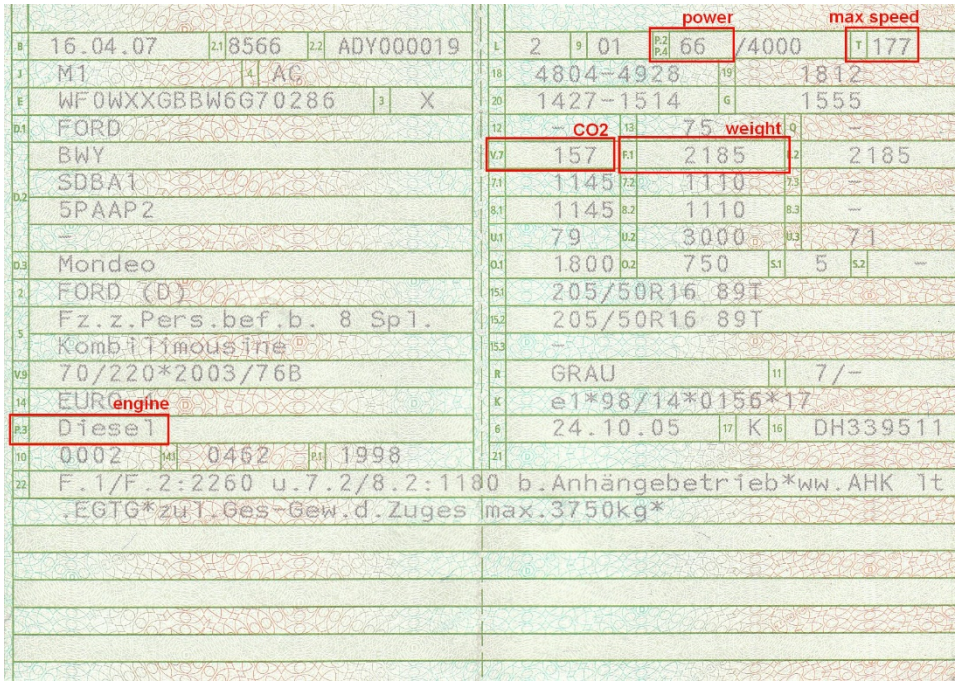
For more details about the ECO-Drive applications, please refer ***App Note: Eco-Drive-GPS Premium Features in AVL Firmware 2.11.0 and Above.*** See *1.3. Related documents*.

5.8.1. ECODRIVE.CAR

Parameter syntax	ECODRIVE.CAR=<"Name”>,<Weight>,<Engine>,<Max_speed>,<Power>,<Loss>[,<Speed>],<Fuel_CO2>

This configuration is used to define the car parameters when controlling fuel consumption. The software computes the fuel consumption of your car in detail and provides you with a complete overview of all costs involved in maintaining the vehicle.

This configuration is used to define the car parameters based on the data given on the car registration certificate.



Parameter	Value	Meaning
<"Name">	Enclosed in parentheses, it specifies the name of vehicle (e.g. "Car01").	
<Weight>	It specifies the weight of vehicle, refer to the registration certificate of the car to be configured (e.g. "2185").	
<Engine>	It specifies the Motortyp (P)etrol,(D)iesel,(G)as o. (E)lectro of the vehicle, refer to the registration certificate of the car to be configured (e.g. "Diesel").	
<Max_speed>	It specifies the maximum allowed speed in Km/h, refer to the registration certificate of the car to be configured (e.g. "177").	
<Power>	It specifies the nominal power expressed in kW, refer to the registration certificate of the car to be configured (e.g. "66").	
<Loss>	It is an internal correction factor for the calculations, at the moment it can be set fixed to 3.3 which will fit most cars).	
<Speed>	It specifies the speed in Km/h (e.g. "100"). This setting should be set only if the <Fuel_CO2> specifies the fuel consumption.	
<Fuel_CO2>	This setting specifies either fuel consumption in 1/100L or fuel emission (CO2) in g/km.If you are going to control the fuel consumption then specify here the fuel consumption of the vehicle given from the manufacturer for the specified <"Speed">. If you are going to control the fuel emission, then specify here the fuel emission in g/km given in the car registration certificate. The <speed> setting is not required. How the configuration could be set/requested:	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,ECODRIVE.CAR="Car01",2185,D,177,66,3.3,100,8 // for fuel consumption $PFAL,Cnf.Set,ECODRIVE.CAR="Car01",2185,D,177,66,3.3,157 // for fuel emission
Get Configuration	$PFAL,Cnf.Get,ECODRIVE.CAR

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Notes

Specify the vehicle parameters used for the definition of the functions ECO-DRIVE-GPS.

5.8.2. ECODRIVE.LIMITS

Parameter syntax	ECODRIVE.LIMITS=<Max_Brake>,<Max_ACC_City>,<Max_ACC_Country>[,<angle_slow>,<angle_high>]

This configuration parameter is used to define the braking deceleration and acceleration limits for city and country road topologies.

Parameter	Value	Meaning
<Max_Brake>	It specifies the limit in m/s² of the braking deceleration (e.g. 2).	
<Max_ACC_City>	It specifies the acceleration limit in m/s2 for the city driving (e.g. 2).	
<Max_ACC_Country>	It specifies the acceleration limit in m/s2 for the country driving (e.g. 1). [<angle_slow>,<angle_high>]	
[<angle_slow>,<angle_high>]	Optional settings. The events EcoDrive.eHarshTurn will be generated between 30 km/h and the max. Speed Country. These events are generated when the vehicle makes a change of direction in one second that is greater than a user-defined angle. The angle <angle_slow> is used for the min. speed and <angle_high> for the max. speed in the Country Speed. The limit value that is used now is the interpolation of the two angles <angle_slow> and <angle_high> and speeds and the current speed.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,ECODRIVE.LIMITS=2,2,1,11,20
Get Configuration	$PFAL,Cnf.Get,ECODRIVE.LIMITS

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Notes

The limits are used for the generation of the braking deceleration and acceleration events in city and country.

5.8.3. ECODRIVE.TOPOLOGY

Parameter syntax	ECODRIVE.TOPOLOGY=<City_Speed>,<Time>,<Country_Speed>,<Highway_Speed>,<Time>

This configuration parameter is used to define the limits for the detection of the different road types and times for switching between the different topologies.

Parameter	Value	Meaning
<City_Speed>	It specifies the speed limit in km/h for driving in the city (e.g. 60).	
<Time>	It specifies the minimum time out in seconds for changing the topology from City to Country (e.g. 10). If the vehicle is travelling too fast compared to the specified <City_Speed> speed limit and the time out runs out, the system reports an event and an alarm with detailed driving information of the current or last trip can be send to the server.	
<Country_Speed>	It specifies the speed limit in km/h for driving in the country (e.g. 110).	
<Highway_Speed>	It specifies the speed limit in km/h for driving in the highway (e.g. 140).	
<Time>	It specifies the minimum time out in seconds for changing the topologies Country and Highway (e.g. 20). If the vehicle is travelling too fast compared to the specified <Country_Speed> or <Highway_Speed> speed limits and the time out runs out, the system reports an event and an alarm with detailed driving information of the current or last trip can be send to the remote server.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,ECODRIVE.TOPOLOGY=60,10,110,140,20
Get Configuration	$PFAL,Cnf.Get,ECODRIVE.TOPOLOGY

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

5.8.4. ECODRIVE.AUTOSTART

Parameter syntax	ECODRIVE.AUTOSTART=<Type>,<Speed>,<Timeout>

This configuration parameter is used to define the trigger conditions and limits for an automatic start and stop of an Eco-Drive trip. If you use this parameter do not try to start and stop an EcoDrive trip with the *$PFAL,EcoDrive.TripStart* and *$PFAL,EcoDrive.TripStop* commands. If an EcoDrive trip will automatically be started and stopped, then executing of both commands return ERROR.

Parameter	Value	Meaning
<Type>	It specifies the trigger to be started. It can be set to:	
	IGN	It starts and stops the trip when powering the vehicle Ignition on and off correspondingly.
	GPS	It starts the trip when the vehicle drives faster than the defined <speed> for the specified time out <Timeout> and stops the trip when driving slower than the defined <speed> for the specified time out <Timeout>.
<Speed>	It specifies the minimum speed in km/h for starting and stopping an ECO-Drive trip with autostart type <GPS>.	
<Timeout>	It specifies the minimum timeout in seconds for starting and stopping an EcoDrive trip with autostart type <GPS>.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,ECODRIVE.AUTOSTART=GPS,10,120 $PFAL,Cnf.Set,ECODRIVE.AUTOSTART=IGN
Get Configuration	$PFAL,Cnf.Get,ECODRIVE.AUTOSTART

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

5.9. GSM

5.9.1. GSM.PIN

Parameter syntax	GSM.PIN=<new_pin>

The parameter lets the AVL device store the entered PIN <new_pin> of the used SIM card.

Parameter	Value	Meaning
<new_pin>	It specifies the PIN number of the used SIM card. This may be for example the SIM PIN to register onto the GSM network, or the SIM PIN to replace the current PIN number with a new one.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GSM.PIN=1321
Get Configuration	$PFAL,Cnf.Get,GSM.PIN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	🗝	🗝	🗝	🗝

Notes

- Successful PIN authentication only confirms that the entered PIN was recognized and correct. The PIN acceptance does not necessarily imply that the AVL device is registered to the

desired network. Typical example: PIN was entered and accepted, but the AVL device fails to register to the network. This may be due to missing network coverage, denied network access with currently used SIM card, no valid roaming agreement between home network and currently available operators etc.

- To verify the present status of network registration, please refer to the LED states in the hardware manual of the AVL device. The next way to verify if it is available, establish remotely a voice or data call.
- No PIN request is more pending, if the PIN number of used SIM card once has been specified and it is sent to the AVL device. The AVL device stores that specified PIN and uses it upon request of the GSM part. No more PIN entry is required from your side, as long as the used SIM card is not replaced with a new one. A PIN message sent without value deletes the existing entry, in this case, the command in chapter *4.7.1.* can be used to insert the SIM PIN.

5.9.2. GSM.BALANCE.DIAL

Parameter syntax	GSM.BALANCE.DIAL=<dial_text>

This setting might have to be modified only for some operators, which do not rely on this standard dial number.

Parameter	Value	Meaning
<dial_text>	It specifies the GSM dial number for retrieving balance information.	

Set Configuration	$PFAL,Cnf.Set,GSM.BALANCE.DIAL=*100#
Get Configuration	$PFAL,Cnf.Get,GSM.BALANCE.DIAL

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.9.3. GSM.CALLID.EN

Parameter syntax	GSM.CALLID.EN=<enable>

It allows the configuration of the caller identification. This feature allows you to identify incoming calls for accepting or rejecting them by using the events accordingly. This parameter refers to the GSM supplementary services CLIP (Calling Line Identification Presentation) and CLIR (Calling Line Identification Restriction). The CLIP enables a called subscriber to get the calling line identity (CLI) of the calling party. The CLIR determines if your participant will see or not your phone number.

Parameter	Value	Meaning
<enable>	It allows you to enable and disable the caller identification. Following values can be set:	
	0	Disables caller identification (CLIP + CLIR)
	1	Enables caller identification (CLIP + CLIR)

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GSM.CALLID.EN=1
Get Configuration	$PFAL,Cnf.Get,GSM.CALLID.EN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.9.4. GSM.OPERATOR.BLACKLIST

Parameter syntax	GSM.OPERATOR.BLACKLIST=<value>

This parameter configuration allows creating a customized blacklist containing the GSM operator names that will be considered unacceptably during GSM registration attempts.

Parameter	Value	Meaning
<value>	It allows the system to create/disable a customized blacklist. Following values can be set:	
	disable	Disables the blacklist addresses. If the GSM.OPERATOR.SELECTION mode is set to any or auto, this setting speeds up the first GPRS attachment after powering on the device. By default, it is set to disable.
	<"operator_1">,…..,<"operator_20">	Specifies the operator name or the operator ID for roaming. Up to 40 either operator names or Operator IDs separated by commas can be specified. (e.g. T-Mobile, E-Plus, D2, 26201 etc.). Each specified operator name or operator ID will not be used while GSM registration prevent roaming or direct GSM connection to those operators. Each entry must be enclosed in quotes (" "). See also the example in the table below.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GSM.OPERATOR.BLACKLIST=disable $PFAL,Cnf.Set,GSM.OPERATOR.BLACKLIST="E-Plus","D2"
Get Configuration	$PFAL,Cnf.Get,GSM.OPERATOR.BLACKLIST

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.9.5. GSM.OPERATOR.SELECTION

Parameter syntax	GSM.OPERATOR.SELECTION=<value>

This setting specifies which operator selection mode should be used. It is strongly recommended not to change operator selection mode during normal operation (i.e. by setting alarms). The device should be restarted after this mode has been changed. Take also special care when configuring operator selection modes which restrict the usage of certain networks. In worst case, GPRS and Data/Voice/SMS service may become unavailable in specific situations (i.e. when being in foreign countries and using **home_country** as operator selection mode). The device will wait at least 5 minutes before performing a new operator search.

Parameter	Value	Meaning
<value>	It allows the system allows determining the mode of operator selection. Following values can be set:	
	home_op	The device first searches for a home network service provider and enables GPRS section as soon as it finds a GSM home operator. When the device is registered at a foreign network (roaming), the GPRS section is disabled. If the registration to the current GSM operator is lost (i.e. when driving through tunnels etc.), the GPRS section is kept active until the GPRS section timeout occurs. After the GPRS times out, the device will close that GPRS section. This allows to re-use the current GPRS session when GSM signal is weak or not available for a short time.
	any	The device search automatically for any GSM operator. If an operator is found that it is not in the blacklist, the device registers to that operator. If no network is found, the device goes on searching process and it remains unregistered. Note that, the GPRS startup might be delayed for a time when using the blacklist. Voice calls and SMS are anytime available and might cause additional costs due to roaming.
	manual,<"operator">	Forces AVL device to select and register itself to any GSM operator found in the operator list. If no GSM operator found, the device will also register to the Roaming Networks (if available), but it stays GPRS detached all the time until one of the listed operators is found. This option is not recommended for security-based applications, as it is possible to configure the system in a way which prevents Voice/Datacall/SMS and GPRS connections. A configured blacklist will be considered. Note that GPRS startup might be delayed a bit if the blacklist feature is used. A whitelist is optional and may be used to specify which operators HAVE to be used for registration. It is possible that no operators can be found and the device keeps searching for them Voice calls and SMS are available when the device can register to a GSM operator. Foreign countries Voice calls/SMS might cause additional costs due to roaming.

Parameter	Value	Meaning
	<"operator">	Specifies the whitelist containing the GSM operator names which are considered acceptably during the GSM registration attempts. Up to 50 operator names comma-separated can be specified. (e.g. T-Mobile, E-Plus, D2, etc.). Do not specify any GSM operator name that is already listed in the GSM.OPERATOR.BLACKLIST. Each entry must be enclosed in quotes (" "). See also the example in the table below. If your application requires more than 50 operator names, than use the parameter "GSM.EXT.WHITELIST" for other 50 operator names.
	GPRS	Forces the AVL device to select only GSM network operators that offer GPRS service. For each cell changes the AVL device performs GPRS cell selection and re-selection processes. If the new cell does not support GPRS services, the current operator will be changed. If no GSM operator with GPRS enabled is found, the AVL device remains unregistered for that time, but it is still searching.
	auto	Forces the AVL device to select and register itself to the any GSM network operator that is currently available. Whenever a GPRS or PPP initialisation fails, the <value> will be switched automatically to "GPRS".
	restrictive_manual,{<"desired_op">,<"desired_op1">,...}	his option is not recommended for security-based applications, as it is possible to configure the system in a way which prevents voice/datacall/SMS and GPRS connections. Same functionality as manual selection additional, no voice/data calls are possible and no outgoing SMS may be sent if no desired operator can be found.
	no_roaming	GSM starts up with any available operator but performs GPRS only if no roaming operator has been found. If within roaming, voice calls and SMS are available.
	restrictive_no_roaming	Like no_roaming, except that no voice/data calls are possible and no outgoing SMS may be sent if no desired operator can be found.
	home_country,<mcc>:	GPRS connection is allowed only for operators within the configured home country. Roaming in foreign country networks can be prevented this way.
	<mcc>:	Specify the mobile country code for the desired country. This is usually a 3-digit integral number - i.e. 262 for Germany. In order to get a mobile country code for another country, simply set operation mode to any. If the device is within the desired country and finds a GSM operator there, the PFAL command PFAL,GSM.MMC can be used to read out the current mobile country code.
	restrictive_home_country,<mcc>:	Like home_country, except that no voice/data calls are possible, and no outgoing SMS may be sent when being registered to foreign networks.
	fixed,<operator_id>:	GSM engine starts with automatic operator search. If a different operator than the specified one is found, the GSM engine will manually register to the specified fixed operator. In case this operator is not visible, GSM keeps searching and it does not provide GSM service in this situation.
	<operator_id>:	Defines a numeric operator ID of a specific provider e.g. 26203

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GSM.OPERATOR.SELECTION=any $PFAL,Cnf.Set,GSM.OPERATOR.SELECTION=manual,"D1","O2" $PFAL,Cnf.Set,GSM.OPERATOR.SELECTION=GPRS
Get Configuration	$PFAL,Cnf.Get,GSM.OPERATOR.SELECTION

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

- The GPRS attachment might be delayed for a few minutes if the *5.9.4.* is used.
- Do NOT change the **GSM.OPERATOR.SELECTION** mode during normal operation. The device will restart after changing the value.
- Keep in main, that during a voice call no operator selection will be performed.

5.9.6. GSM.EXT.WHITELIST

Parameter syntax	GSM.EXT.WHITELIST=<"operatorID">, <"operatorID">,<"operatorID">,....,<"operatorID">

This setting specifies which operator IDs should additionally be used to the list of operator IDs added in the parameter *5.9.5.*. After the equal sign "=" a comma "," is required. Please note that if your application requires more than 50 operator names than:

```
First configure the GSM.EXT.WHITELIST as below:
$PFAL,Cnf.Set,GSM.EXT.WHITELIST=,"24701","24702","24801
","24802","25001".
and then enter to the list GSM.OPERATOR.SELECTION the
other operator IDs as below:
$PFAL,Cnf.Set,GSM.OPERATOR.SELECTION=manual,"20201","20
210","20416",....
```

<"operatorID">

Specifies the whitelist containing the GSM operator IDs which are considered acceptably during the GSM registration attempts. Up to 50 operator IDs comma-separated can be specified. (e.g. "24701", "24702","24801" etc.).

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GSM.EXT.WHITELIST=,"12345","67890"
Get Configuration	$PFAL,Cnf.Get, GSM.EXT.WHITELIST

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.9.7. GSM.OPERATOR.GPRSCHECK

Parameter syntax	GSM.OPERATOR.GPRSCHECK=<enable>

This setting enables or disables the check for GSM base cells whether they provide GPRS service or not. By default these tests are disabled now, which allows to use GPRS service of GSM cells which do not correctly set up their “GPRS availability” information. As such cells were discovered in specific areas in Europe, it is recommended to leave the default setting untouched in order to improve connectivity of the AVL devices.

Parameter	Value	Meaning
<enable>	Following settings are available:	
	disabled	Disables the check for GSM base cells whether they provide GPRS service.
	enabled	Enables the check for GSM base cells whether they provide GPRS service.

How the configuration could be set/requested:

Set Configuration	$PFAL,CNF.Set,GSM.OPERATOR.GPRSCHECK=enabled
Get Configuration	$PFAL,Cnf.Get,GSM.OPERATOR.GPRSCHECK

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.9.8. GSM.OPLOST.RESTART

Parameter syntax	GSM.OPLOST.RESTART=<enable>,<interval>,[<fieldstrength>]

This setting is needed when the device enters or is being in areas with bad GSM coverage. If the GSM operator is lost, the AVL device reinitializes the GSM engine periodically (at specified intervals of time), until a GSM operator found.

Parameter	Value	Meaning
<enable>	It allows the system to re-initialize the GSM engine, in event of bad GSM coverage. Following values can be set:	
	0	Disables GSM re-initialization, when there is no GSM operator available.
	1	Enables GSM re-initialization. This setting reinitializes GSM engine if no valid GSM operator has been found within the user-specified <interval>.
<interval>	It allows you to define the amount of time, in milliseconds, on which the system reinitializes the GSM engine, if no GSM operator found. Following values can be set:	
	0 .. **2147483647**	The amount of time in milliseconds after which the GSM engine will be reinitialized, if no GSM operator found.

Parameter	Value	Meaning
[<fieldstrength>]	This value is **optional**, it can also be omitted. If it is used, it allows you to define the minimum GSM field strength between **1** and **20**. System checks additionally whether GSM signal strength drops below the specified minimal field strength during the defined amount of time <interval>. ***Note:*** *If field strength drops just for a short time, the GSM engine will not perform a restart process.*	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GSM.OPLOST.RESTART=1,300000 $PFAL,Cnf.Set,GSM.OPLOST.RESTART=1,300000,7
Get Configuration	$PFAL,Cnf.Get,GSM.OPLOST.RESTART

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.9.9. GSM.SMS.RESPONSE

Parameter syntax	GSM.SMS.RESPONSE=<enable>,<amount>

Whenever a PFAL command is sent via SMS to the device, its answer will be sent back to the origin number. The maximum number of SMS sent back may be configured. Warning: the maximum number may not exceed the maximal number of SMS output slots *(else this value will be taken)*. If too fewer output slots are free when a SMS-PFAL command is detected, the answer will be queued inside the remaining output slots. Additional SMS will be deleted if there are no more free slots available. *(Worst case: all output buffers are filled, so no SMS answer can be sent at the moment. Retry later)*.

Parameter	Value	Meaning
<enable>	It allows you to enable SMS responses of PFAL commands to the sender (by SMS communication only). Following values can be set:	
	0	Disables SMS responses
	1	Enables SMS responses
<amount>	It specifies the maximum number of SMS sent back to the SMS sender. Following values can be set:	
	1...5	The number of SMS responses.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GSM.SMS.RESPONSE=1,5
Get Configuration	$PFAL,Cnf.Get,GSM.SMS.RESPONSE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Warning: ***The maximum number of responses may not exceed the maximal number of SMS output slots (else default value will be used). If too fewer output slots are free when a SMS-PFAL command is detected, the response will be queued inside the remaining output slots. Additional SMS will be deleted, if there are no more free slots available. (Worst case: if all output buffers are used up, no SMS responses can be sent at the moment. Retry later).***

5.9.10. GSM.MODEPREF

Parameter syntax	GSM.MODEPREF=<network>

This setting Forces the selection of the mobile network operator and preferences for the FOX3 (2G)/3G devices. Dual or tri mode determine which network is selected first (preferred) if both or all three are available.

Parameter	Value	Meaning
<network>	One of the following modes can be set.	
	AUTO[1]	Depending on the network availability, the device automatically switches between the 2G / 3G / 4G modes
	GSM	Enables GSM/2G (single mode) only
	UMTS[2]	Enables 3G (single mode) only
	LTE[3]	Enables LTE (single mode) only
	DUAL_GSM	Enables 3G and GSM/2G, preferred use of 2G (dual mode)
	DUAL_UMTS[2]	Enables 3G and GSM/2G, preferred use of 3G (dual mode)
	DUAL_LTE[3]	Enables LTE and GSM/2G, preferred use of LTE (dual mode)
	TRI_LTE[4]	Enables LTE, 3G and GSM/2G, preferred use of LTE (tri mode)

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GSM.MODEPREF=DUAL_GSM
Get Configuration	$PFAL,Cnf.Get,GSM.MODEPREF

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓
2	×	✓	✓	×
3	×	×	✓	×

Exception: UMTS mode not supported on BOLERO41.

5.9.11. GSM.SIMSLOT

Parameter syntax	GSM.SIMSLOT=<slot>

This setting allows to select the used SIM card slot in the Lantronix device.

Parameter	Value	Meaning
<slot>	One of the following modes can be set:	
	1	First SIM card slot (for BOLERO40 Series only).
	2	Second SIM card slot (for BOLERO40 Series only).

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GSM.SIMSLOT=1 $PFAL,Cnf.Set,GSM.SIMSLOT=2
Get Configuration	$PFAL,Cnf.Get,GSM.SIMSLOT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	✓

5.10. GPRS

5.10.1. GPRS.APN

Parameter syntax	GPRS.APN=<value>

This parameter specifies the APN Access Point Name (text string). The APN is logical name that is used to select the GGSN or the external packet data network. In other words, the APN name that your network operator has provided to connect the AVL device to the Internet. The APN specifies the external network that an AVL device can access. It also defines the type of IP address to be utilized, security mechanisms to invoke, available value added services, redundancy, and fixed-end connections.

<value>

It specifies the Access Point Name (APN). The logical name that will be used to select the GGSN or the external packet data network. Consult your Network Operator for the correct APN settings).

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GPRS.APN=internet.t-d1.de
Get Configuration	$PFAL,Cnf.Get,GPRS.APN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

If the value is omitted, then it is set to the default value.

5.10.2. GPRS.APN2

Parameter syntax	GPRS.APN2=<value>

This parameter specifies the APN Access Point Name (text string) for the 2nd SIM card slot in the BOLERO40 series. The APN is logical name that is used to select the GGSN or the external packet data network. In other words, the APN name that your network operator has provided to connect the BOLERO40 device to the Internet. The APN specifies the external network that a BOLERO40 device can access. It also defines the type of IP address to be utilized, security mechanisms to invoke, available value added services, redundancy, and fixed-end connections.

<value>

It specifies the Access Point Name (APN). The logical name that will be used to select the GGSN or the external packet data network. Consult your Network Operator for the correct APN settings).

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GPRS.APN2=internet.t-d1.de
Get Configuration	$PFAL,Cnf.Get,GPRS.APN2

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	✓

Notes

If the value is omitted, then it is set to the default value.

5.10.3. GPRS.APN.ALTERNATIVE

Parameter syntax	GPRS.APN.ALTERNATIVE=<value>

This parameter is optional. If it is configured, the device will try to connect to the alternative APN server after two failed connection attempts (network errors/APN not reachable etc.) to the first specified APN server. If both servers cannot be reached, the operator selection is informed and may choose a different operator (depending on operator selection strategy and current network visibility).

```
GPRS.QOS=3,4,3,0,0quality of service
GPRS.DIAL=ATD*99***1#dialup text for GPRS *
GPRS.QOSMIN=0,0,0,0,0minimal quality of service
GPRS.AUTOSTART=0enable autostart of GPRS(1=enable)
* this setting needs usually not to be changed as most
regions use the default dialup text
```

It is STRONGLY recommended to set GPRS.AUTOSTART=1 when using the GPRS service.

`GPRS.TIMEOUT=1,600000`

- Enable GPRS timeout (GPRS will be restarted if there is no TCP communication for longer than the specified timespan)
- **Do NOT** use timeout values below the configured TCP.CLIENT.TIMEOUT value. Else the GPRS connection will be closed (and might be restarted) periodically, which may lead to a unreachable device (during GPRS connect, disconnect no commands/data can be transmitted to/from the device)
- In the rare case of network environment prevents closing the GPRS connection, device performs a hard reset in order to maintain data transfer via GPRS (accessibility from server).

- default: enabled, 10 minutes timeout (which means: restart GPRS connection every 10 minutes if no TCP communication is possible).
- PPP AUTOPING can be disabled when using this configuration setting

<value>

It specifies the alternative Access Point Name (APN). The logical name that will be used to select the GGSN or the external packet data network. Consult your Network Operator for the correct APN settings).

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GPRS.APN.ALTERNATIVE=internet.t-d1.de
Get Configuration	$PFAL,Cnf.Get,GPRS.APN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

If the value is omitted, then it is set to the default value.

5.10.4. GPRS.AUTOSTART

Parameter syntax	GPRS.AUTOSTART=<value>

It allows you to specify the start-up mode of the GPRS connection.

Parameter	Value	Meaning
<value>	It specifies the start-up mode of the GPRS connection. It can be set to:	
	0	Disables the automatic attachments to the GPRS network.
	1	Enables the automatic attachments to the GPRS network. If the GPRS network connection gets lost, it tries to reconnect automatically as soon as the network is available again.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GPRS.AUTOSTART=1
Get Configuration	$PFAL,Cnf.Get,GPRS.AUTOSTART

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.10.5. GPRS.DIAL

Parameter syntax	GPRS.DIAL=<dial_text>

This configuration causes the AVL device to establish a communication to the external PDN (Public Data Network). The V.250 'D' (Dial) command causes the AVL device to enter the V.250 online data state.

<dial_text>

It specifies the GPRS dial text. Please contact your network provider to specify the correct dial text.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GPRS.DIAL=ATD*99***1#
Get Configuration	$PFAL,Cnf.Get,GPRS.DIAL

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

Usually, this setting needs not to be changed, because most regions use the default dial text (ATD*99***1#).

5.10.6. GPRS.TIMEOUT

Parameter syntax	GPRS.TIMEOUT=<enable>,<G_timeout>

This parameter is used to detach the device from GPRS network if there is no TCP communication available when the timeout has passed.

Parameter	Value	Meaning
<enable>	Define whether or not the timeout <G_timeout> has to be utilized. By default, this value is set to **1**. However, it can be set to:	
	0	Disables the GPRS timeout.
	1	Enables the GPRS timeout. If the **GPRS.TIMEOUT** is enabled you have to manually disable the **PPP.AUTOPING** parameter.
<G_timeout>	Specify the period of time *(in milliseconds)* within which the target device re-initializes a GPRS connection, if there is no TCP communication established within the user-specified timeout. Please note that, the specified value <G_timeout> must be greater than the timeout value <C-timeout> defined by the **TCP.CLIENT.TIMEOUT** parameter, otherwise the GPRS connection will be closed (and might be restarted) periodically, which may lead to an unreachable device. By default, this value is set to 600000. It means, GPRS restarts every 10 minutes if there is no TCP communication available.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GPRS.TIMEOUT=1,600000
Get Configuration	$PFAL,Cnf.Get,GPRS.TIMEOUT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.10.7. GPRS.QOSMIN

Parameter syntax	GPRS.QOSMIN=<precedence>,<delay>,<reliability>,<pea>,<mean>

This parameter name allows the AVL device to specify a minimum acceptable profile, which is

checked by the AVL device against the negotiated profile returned in the Activate PDP Context Accept message.

Parameter	Value	Meaning	
<precedence>	Precedence class (numeric)		
	[0]	Network subscribed value.	
	1	High Priority Service commitments shall be maintained ahead of precedence classes 2 and 3.	
	2	Normal Priority Service commitments shall be maintained ahead of precedence class 3.	
	3	Low Priority Service commitments shall be maintained.	
<delay>	Delay class (numeric): The Delay parameter defines end-to-end transfer delay incurred in the transmission of SDUs through GPRS network(s).		
	[0]	Network subscribed value.	
SDU size:128 octets:			
	Delay Class	**Mean Transfer Delay**	**95 percentile Delay**
	1 (Predictive)	<0.5	<1.5
	2 (Predictive)	<5	<25
	3 (Predictive)	<50	<250
	4 (Best Effort)	Unspecified	
SDU size: 1024 octets:			
	Delay Class	**Mean Transfer Delay**	**95 percentile Delay**
	1 (Predictive)	<0.5	<1.5
	2 (Predictive)	<5	<25
	3 (Predictive)	<50	<250
	4 (Best Effort)	Unspecified	
<reliability>	Reliability class (numeric)		
	[0]	network subscribed value	
	1	Non real-time traffic, error-sensitive application that cannot cope with data loss	
	2	Non real-time traffic, error-sensitive application that can cope with infrequent data loss	
	3	Non real-time traffic, error-sensitive application that can cope with data loss	
	4	Real-time traffic, error-sensitive application that can cope with data loss	
	5	Real-time traffic, error non-sensitive application that can cope with data loss	
<peak>	(numeric) Peak throughput class (in octets per second)		
	[0]	network subscribed value	

Peak Throughput Class	Peak Throughput (in octets per second)
1	Up to 1 000 (8 kbit/s).
2	Up to 2 000 (16 kbit/s).
3	Up to 4 000 (32 kbit/s).
4	Up to 8 000 (64 kbit/s).
5	Up to 16 000 (128 kbit/s).
6	Up to 32 000 (256 kbit/s).
7	Up to 64 000 (512 kbit/s).
8	Up to 128 000 (1 024 kbit/s).
9	Up to 256 000 (2 048 kbit/s).

Parameter	Value	Meaning
<mean>	(numeric) Mean throughput class.	
	[**0**]	network subscribed value

Mean Throughput Class	Mean Throughput (in octets per hour)
1	100 (~0.22 bit/s)
2	200 (~0.44 bit/s)
3	500 (~1.11 bit/s
4	1 000 (~2.2 bit/s)
5	2 000 (~4.4 bit/s)
6	5 000 (~11.1 bit/s)
7	10 000 (~22 bit/s)
8	20 000 (~44 bit/s)
9	50 000 (~111 bit/s)
10	100 000 (~0.22 kbit/s)
11	200 000 (~0.44 kbit/s)
12	500 000 (~1.11 kbit/s)
13	1 000 000 (~2.2 kbit/s)
14	2 000 000 (~4.4 kbit/s)
15	5 000 000 (~11.1 kbit/s)
16	10 000 000 (~22 kbit/s)
17	20 000 000 (~44 kbit/s)
18	50 000 000 (~111 kbit/s)
31	best effort.

How the configuration could be set/requested:

Set configuration	$PFAL,Cnf.Set,GPRS.QOSMIN=3,4,3,0,0
Get configuration	$PFAL,Cnf.Get,GPRS.QOSMIN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

The value is set to the default, if it is omitted.

5.10.8. GPRS.QOS

Parameter syntax	GPRS.QOS=<precedence>,<delay>,<reliability>,<peak>,<mean>

This parameter name allows the AVL device to specify a Quality of Service Profile that is used when the AVL device sends an Activate PDP Context Request message to the network.

Parameter	Value	Meaning	
<precedence>	Precedence class (numeric)		
	[0]	Network subscribed value.	
	1	High Priority Service commitments shall be maintained ahead of precedence classes 2 and 3.	
	2	Normal Priority Service commitments shall be maintained ahead of precedence class 3.	
	3	Low Priority Service commitments shall be maintained.	
<delay>	Delay class (numeric): The Delay parameter defines end-to-end transfer delay incurred in the transmission of SDUs through GPRS network(s).		
	[0]	Network subscribed value.	
SDU size:128 octets:			
	Delay Class	**Mean Transfer Delay**	**95 percentile Delay**
	1 (Predictive)	<0.5	<1.5
	2 (Predictive)	<5	<25
	3 (Predictive)	<50	<250
	4 (Best Effort)	Unspecified	
SDU size: 1024 octets:			
	Delay Class	**Mean Transfer Delay**	**95 percentile Delay**
	1 (Predictive)	<0.5	<1.5
	2 (Predictive)	<5	<25
	3 (Predictive)	<50	<250
	4 (Best Effort)	Unspecified	

Parameter	Value	Meaning
<reliability>	Reliability class (numeric)	
	[0]	network subscribed value
	1	Non real-time traffic, error-sensitive application that cannot cope with data loss
	2	Non real-time traffic, error-sensitive application that can cope with infrequent data loss
	3	Non real-time traffic, error-sensitive application that can cope with data loss
	4	Real-time traffic, error-sensitive application that can cope with data loss
	5	Real-time traffic, error non-sensitive application that can cope with data loss
<peak>	(numeric) Peak throughput class (in octets per second)	
	[0]	network subscribed value

Peak Throughput Class	Peak Throughput (in octets per second)
1	Up to 1 000 (8 kbit/s).
2	Up to 2 000 (16 kbit/s).
3	Up to 4 000 (32 kbit/s).
4	Up to 8 000 (64 kbit/s).
5	Up to 16 000 (128 kbit/s).
6	Up to 32 000 (256 kbit/s).
7	Up to 64 000 (512 kbit/s).
8	Up to 128 000 (1 024 kbit/s).
9	Up to 256 000 (2 048 kbit/s).

Parameter	Value	Meaning
<mean>	(numeric) Mean throughput class.	
	[0]	network subscribed value

Mean Throughput Class	Mean Throughput (in octets per hour)
1	100 (~0.22 bit/s)
2	200 (~0.44 bit/s)
3	500 (~1.11 bit/s
4	1 000 (~2.2 bit/s)
5	2 000 (~4.4 bit/s)
6	5 000 (~11.1 bit/s)
7	10 000 (~22 bit/s)
8	20 000 (~44 bit/s)
9	50 000 (~111 bit/s)

Mean Throughput Class	Mean Throughput (in octets per hour)
10	100 000 (~0.22 kbit/s)
11	200 000 (~0.44 kbit/s)
12	500 000 (~1.11 kbit/s)
13	1 000 000 (~2.2 kbit/s)
14	2 000 000 (~4.4 kbit/s)
15	5 000 000 (~11.1 kbit/s)
16	10 000 000 (~22 kbit/s)
17	20 000 000 (~44 kbit/s)
18	50 000 000 (~111 kbit/s)
31	best effort.

<How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GPRS.QOS=3,4,3,0,0
Get Configuration	$PFAL,Cnf.Get,GPRS.QOS

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

If the value is omitted, it is set to the default value.

5.11. PPP

5.11.1. PPP.USERNAME

Parameter syntax	PPP.USERNAME=<value>

This parameter allows the AVL device to specify user name that will be used to log (attach) itself into the GPRS network.

<value>

String type supplied by your GPRS provider. By default, it is set to "none". A string is required for the Chap and Pap authentication methods over PPP.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,PPP.USERNAME=t-d1
Get Configuration	$PFAL,Cnf.Get,PPP.USERNAME

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

Most providers do not require the username, however, if it is required your GPRS provider should have provided the details with your GPRS subscription.

5.11.2. PPP.USERNAME2

Parameter syntax	PPP.USERNAME2=<value>

This parameter allows the BOLERO40 device to specify user name for the 2nd SIM card that will be used to log (attach) itself into the GPRS network.

<value>

String type supplied by your GPRS provider. By default, it is set to "none". A string is required for the Chap and Pap authentication methods over PPP.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,PPP.USERNAME2=t-d1
Get Configuration	$PFAL,Cnf.Get,PPP.USERNAME2

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✗	✗	✗	✓

Notes

Most providers do not require the username, however, if it is required your GPRS provider should have provided the details with your GPRS subscription.

5.11.3. PPP.PASSWORD

Parameter syntax	PPP.PASSWORD=<value>

This parameter allows the terminal to specify password that will be used to log (attach) into the GPRS network.

<value>

String type supplied by your GPRS provider. By default, it is set to "**none**". A string is required for the Chap and Pap authentication methods over PPP.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,PPP.PASSWORD=gprs
Get Configuration	$PFAL,Cnf.Get,PPP.PASSWORD

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.11.4. PPP.PASSWORD2

Parameter syntax	PPP.PASSWORD2=<value>

This parameter allows the BOLERO40 device to specify password for the 2nd SIM card that will be used to log (attach) into the GPRS network.

<value>

String type supplied by your GPRS provider. By default, it is set to "**none**". A string is required for

the Chap and Pap authentication methods over PPP.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,PPP.PASSWORD2=gprs
Get Configuration	$PFAL,Cnf.Get,PPP.PASSWORD2

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.11.5. PPP.AUTOPING

Parameter syntax	PPP.AUTOPING=<type>,<time>

It allows you to enable/disable the maximal idle time until the next ping will send to the GPRS network for keeping the GPRS connection alive.

Parameter	Value	Meaning
<type>	By default, it is set to **0** (disabled). However, it can be set to:	
	0	Disables sending of pings to the GPRS network.
	1	Enables sending of pings to the GPRS network for keeping the GPRS connection alive. It is recommended to enable it if the **GPRS.TIMEOUT** remains disabled, otherwise set it to **0**.
<time>	It specifies the amount of time, in milliseconds, on which a ping will be sent to the GPRS network.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,PPP.AUTOPING=1,180000
Get Configuration	$PFAL,Cnf.Get,PPP.AUTOPING

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

The **GPRS.TIMEOUT** parameter replaces the functionality of the **PPP.AUTOPING**.

5.11.6. PPP.AUTH

Parameter syntax	PPP.AUTH=<auth_type>

This parameter allows you to define the authentication method to be used over PPP. By default, the authentication over PPP is selected automatically. However, in some GSM networks it might be required to define the authentication method manually.

Parameter	Value	Meaning
<auth_type>	Specifies the authentication method over PPP. It can be set to:	
	none	No authentication is done (**NOT** recommended to use this setting).
	auto	authentication method is selected automatically.
	pap	Only PAP authorization method is used.
	chap	Only CHAP authorization method is used.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,PPP.AUTH=auto
Get Configuration	$PFAL,Cnf.Get,PPP.AUTH

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.12. TCP

Our AVL devices connected to the Internet via GPRS use a protocol called TCP/IP to communicate with each other. When an AVL device wants to send data to a remote server, it must know the destination IP address for sending the data to. That data is sent either via TCP or UDP.

TCP stands for *Transmission Control Protocol*. Using a TCP connection, the AVL device sending the data connects directly to that computer/server that should read/receive this data, and stay connected during the data transfer. This transmission can guarantee that the data an AVL device sends will reach its destination safely and correctly and then disconnect the connection.

UDP stands for *User Datagram Protocol*. Using a UDP connection, the AVL device sending the data does not connect directly to that computer/server that should read/receive this data as TCP does, but the data is published into the network with the hopes getting to the right place. This transmission does not guarantee that the data an AVL device sends will ever reach its destination.

5.12.1. TCP.CLIENT.CONNECT

Parameter syntax	TCP.CLIENT.CONNECT=<s_enable>,<ip_address>,<tcp_port>

This parameter specifies the connection type, IP-address and port that your application will use to connect to the remote server. AVL device actively initiates a connection to a remote server when GPRS state is online. Firstly, set the IP-address and Port number and then execute manually (if <s_enable> is set to 0) the "**TCP.Client.Connect**" command to connect to the server, as shown in chapter *4.8.1.1.* When the connection has been established successfully, the "TCP.Client.eConnected" event occurs. If the remote server rejected the connection, an Error event occurs. After a connection has been established, use the **TCP.Client.Send,<protocols>,<"text">** command to stream data to a remote server. A "**TCP.Client.ePacketSent**" and a "T**CP.Client.eReceived**" event occurs when there is outgoing and incoming data accordingly. If the <s_enable>=0, use the "**TCP.Client.Disconnect**" command

to terminate the connection, otherwise set the <s_enable>=0 before using the "**TCP.Client.Disconnect**" command.

Parameter	Value	Meaning
<s_enable>	Activates/deactivates automatically connection to the remote server. Following values can be used:	
	0	Deactivates automatically connection to the remote server.
	1	Activates automatically connection to the remote server. If the TCP connection to the remote server gets lost, the AVL device will automatically attempt to reconnect as soon as the remote server will be available again.
<ip_address>	IP address in dotted-four-byte format. It is the IP address to which the AVL device will be registered and send its data. The format of this address is "xxx.xxx.xxx.xxx". See also chapter *2.3.* It is also possible to specify a DNS address, max. 255 chars, in the hostname form (for example **www.lantronix.com**) instead of a dynamic or static IP-address. In such a case it is not required to set up any DNS provider, it is automatically done.	
<tcp_port>	Specifies the TCP port number used for communication between the AVL device and remote server.	

Notes

- The port number has to be specified in ANY case.
- Using DNS can cause much traffic if the specified domain does not exist (the device keeps requesting of the IP-address until one is returned)
- The AVL device clears automatically the implemented DNS cache (inside the unit) if the server refuses that connection immediately (it deletes an old DNS entry that maybe not up to date. During the next reconnection the AVL device retrieves automatically a new IP-address and can continue normal operation)
- Note that, if the server login fails (i.e. the server closes the TCP connection without requiring any acknowledge), DNS query process will be restarted during the next reconnection.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT.CONNECT=1,217.119.194.35,2222
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT.CONNECT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

- All values are separated by comma ",".
- Contact your network administrator to specify a correct server IP address and port number.

5.12.2. TCP.CLIENT.ALTERNATIVE

Parameter syntax	TCP.CLIENT.ALTERNATIVE=<a_enable>,<a_ip_address>,<tcp_port>,<timeout>

This parameter defines an alternative server in case the primary server fails. This allows to define a fallback server which can be used in case the primary server gets disconnected/doesn't respond anymore. If both servers aren't available, the device continues to attempt a connection (alternately to defined "primary" and "alternative" server). A wait time between 2 connection attempts can be specified using the setting **TCP.CLIENT.TIMEOUT.**

Parameter	Value	Meaning
<a_enable>	Enables/disables automatically connection to the remote server. Following values can be used:	
	0	Disables automatically connection to the alternative server (**default**).
	1	This Server will be used for each second connection attempt.
<a_ip_address>	IP address in dotted-four-byte format. It is the IP address to which the AVL device will be registered and send its data. The format of this address is "xxx.xxx.xxx.xxx". See also chapter *2.3.* It is also possible to specify a DNS address in the hostname form (for example **www.lantronix.com**) instead of a static IP-address. In such a case it is not required to set up any DNS provider, it is automatically done.	
<tcp_port>	Specifies the port number used for communication between the AVL device and alternative remote server.	
<timeout>	Optional. Defines an independent TCP timeout for the alternative server. If no timeout is specified, then ***TCP.CLIENT.TIMEOUT*** setting is used.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT.ALTERNATIVE=0,217.119.194.35,2222 $PFAL,Cnf.Set,TCP.CLIENT.ALTERNATIVE=1,www.lantronix.com,2222,300000
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT.ALTERNATIVE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

All values are separated by commas ","

Contact your network administrator to specify a correct server IP address and port number.

5.12.3. TCP.CLIENT.PING

Parameter syntax	TCP.CLIENT.PING=<type>,<time>

It allows you to activate/deactivate the maximum idle time until the next ping must be sent to the remote server.

Parameter	Value	Meaning
<type>	It can be set to:	
	0	Deactivates sending of pings to the remote server.
	1	Activates sending of pings to the remote server (it is recommended). These pings are used to ensure the remote server that the AVL device is still present in an active TCP connection.
<time>	It specifies the amount of time, in milliseconds, on which a ping will be sent to the remote server.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT.PING=1,120000
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT.PING

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.12.4. TCP.CLIENT.TIMEOUT

Parameter syntax	TCP.CLIENT.TIMEOUT=<C-timeout>,<wait_time>

This parameter is used to define the period of time the device will wait for a response and between two connection attempts when the TCP connection fails.

Parameter	Value	Meaning
<C-timeout>	Specifies the period of time, in milliseconds, the target device will wait for a response *(an acknowledgement)* from the remote server about the received data and the next data transmission.	
<wait_time>	Specifies the length of time, in milliseconds, the AVL device waits between two connection attempts. Please note, the **TCP** <C-timeout> value must always be smaller than the **GPRS** <G_timeout> value, otherwise the AVL device closes the GPRS connection before trying to perform a TCP connection to the remote server.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT.TIMEOUT=240000,60000
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT.TIMEOUT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

In general, the timeout should never exceed 10 days (10*24*3600*1000).

The standard specification for a TCP timeout is 5 minutes. The timeout above 15 minutes are not recommended.

5.12.5. TCP.CLIENT.DNS.TIMEOUT

Parameter syntax	TCP.CLIENT.DNS.TIMEOUT=<dns_cache_timeout>

AVL device has the feature to cache the DNS records for a fixed period of time. This parameter allows to control caching of DNS records. The DNS cache consists of a fixed portion of SRAM memory. The DNS cache is lost whenever the AVL device is switched off and no user-backup-battery is already connected.

<dns_cache_timeout>

Specifies the DNS cache timeout. The length of time (in seconds) to keep the DNS cache valid. After the time expires, the DNS memory cache will be updated with new data. It can be set to a value from **0** to **2147483647**.

Setting the DNS cache timeout to **0** results that the AVL device always performs a new DNS query process.

Setting the DNS cache timeout to **2147483647** results that AVL device each **49.7** days (converted from seconds to days) performs a new DNS query process. By default, it is set to **86400** resulting each **1** day the AVL device performs a new DNS query process.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT.DNS.TIMEOUT=86400
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT.DNS.TIMEOUT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

However, a DNS query process will automatically be performed, whenever the AVL device tries to perform a TCP connection and the TCP connection establishment fails.

After a system restart the TCP messages in the non-volatile buffer will only be lost if the device configuration is set to TCP.CLIENT.SENDMODE=0. To retain these messages in the non-volatile buffer you have to set the device configuration to TCP.CLIENT.SENDMODE=2.

5.12.6. TCP.CLIENT.LOGIN

Parameter syntax	TCP.CLIENT.LOGIN=<login>,[<security_mode>]

It allows you to activate sending of login data automatically to the used remote server after the TCP connection is established.

Parameter	Value	Meaning
<login>	It can be set to:	
	0	Deactivates sending of login information to the remote server.
	1	Activates sending of login information to the remote server. It sends the same data as the "" message described in chapter This data is sent after the AVL device has successfully established a TCP connection to the remote server. The login data can then be used to ensure the connection of the AVL device to the remote server.
	2	Activates sending of login information to the remote server in a new format (shrink format) that is supported by the firmware release 2.13.0 and above. It sends the same data as the "" message described in chapter This data is sent after the AVL device has successfully established a TCP connection to the remote server. The login data can then be used to ensure the connection of the AVL device to the remote server.
[<security_mode>]	PREMIUM-FEATURE “AES_TCP” will be required to enable and use this setting. Current implementation without PREMIUM-FEATURE contains fixed AES key (128bit 0-bits) and 2 modes of operation. If no parameter is specified the default setting will be used (Default = no AES).	
	1[1]	AES encryption (ECB mode)
	2[2]	AES encryption (CBC mode)
	3	No AES encryption
	0	No AES encryption

How the configuration could be set/requested:

Set Configuration	$PFAL,CNF.Set,TCP.CLIENT.LOGIN=1,0 $<MSG.Info.ServerLogin> $DeviceName=Unnamed AVL $Security=0 $Software=avl_3.0.0 (BxBGT1gzIHJIdjowMy1OVUNIAgEA) $Hardware=FOX3 rev:03-NUCH $LastValidPosition=$GPRMC,143445.000,A,5040.4096,N,01058.8542,E,0.01,0.00,040315,, $IMEI=353816054739497 $PhoneNumber=+491773456789 $LocalIP=10.208.151.168 $CmdVersion=2 $SUCCESS $<end> $PFAL,CNF.Set,TCP.CLIENT.LOGIN=2,0 $<MSG.Info.ServerLogin> $DeviceName=Unnamed AVL $Software=avl_3.0.0_rc20 $Hardware=FOX3 rev:03-NUCH $IMEI=353816054739497 $PhoneNumber=+491773456789 $Position=$GPRMC,143545.000,A,5040.4086,N,01058.8543,E,0.00,0.00,040315,, $SUCCESS $<end> $PFAL,CNF.Set,TCP.CLIENT.LOGIN=2,2 $<MSG.Info.ServerLogin> $DeviceName=Unnamed AVL $Software=avl_3.0.0_rc20 $Hardware=FOX3 rev:03-NUCH $IMEI=353816054739497 $PhoneNumber=+491773456789 $Position=$GPRMC,144217.000,A,5040.4086,N,01058.8543,E,0.00,0.00,040315,, $Security=2 $SUCCESS $<end>
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT.LOGIN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	🗝	🗝	🗝	🗝

5.12.7. TCP.CLIENT.LOGIN.EXT

Parameter syntax	TCP.CLIENT.LOGIN.EXT=<"text ">

It allows you to add additional information to the regular login data that the device will send to the remote server after establishing a TCP connection.

<"text">

Wrapped in quotation marks (" "), specify up to 100 characters. The specified string can be used as an extension to the login data that the AVL device will send to the TCP server after it is connected.

```
The text sent to the server will look like:
$<MSG.Info.ServerLogin>
$DeviceName=unnamed FOX3
$Ext=example extension text
$Software= ..........
```

The user may also specify any dynamic protocol. Each dynamic variable is separated by ampersand "&" without spaces and enclosed in parentheses "( )". Dynamic variables are listed in chapter 7.

```
For example: $PFAL,Cnf.Set,TCP.CLIENT.LOGIN.EXT="&(OwnNumber)"
```

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT.LOGIN.EXT="example extension text"
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT.LOGIN.EXT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.12.8. TCP.CLIENT.SENDMODE

Parameter syntax	TCP.CLIENT.SENDMODE=<mode>

This parameter configuration allows to select between fast and safe TCP transmissions.

Note: *It is strongly NOT recommended to change this setting when alarms are active (during regular device operation). Restart the device after this mode is changed!Otherwise loss of stored data is possible (i.e. volatile or non-volatile stored packets). Furthermore, it may cause data to be transmitted partial.*

Parameter	Value	Meaning
<mode>	It specifies how fast TCP packets should be sent. It can be set to:	
	0	**Safe and volatile transmission mode (default).** Safe transmission prevents losing TCP packets during TCP/GPRS reconnects. Only **one** single packet is sent at a time. Outgoing TCP packets sent with ***MSG.Send.TCP, TCP.Client.Send*** and any data transferred using data transfer mode from the serial port are stored inside volatile memory (40KB size) and will be erased by system when it performs a reset and/or power down. When the server has confirmed a packet with "ack sent", the next packet is sent. Even if a TCP timeout occurs no packets can get lost. PFAL commands can be executed when being inside the command mode and answers are appended to the volatile buffer and sent after any data inside the buffer is sent.
	1	Fast and volatile transmission mode. It shall NOT be used (use for testing purposes only). - robust communication cannot be guaranteed - stop of communication has been observed on some networks (stuck communication which requires manual device reset); might be solved by future firmware versions - several packets are sent at a time - if a TCP/GPRS timeout happens, sent but not yet acknowledged packets could get lost - packets are stored inside volatile memory and will be erased by system resets and/or system power down.
	2[,<buffer_level>]	Safe and non-volatile transmission mode. Outgoing messages sent with *MSG.Send.TCP*, *TCP.Client.Send* and *TCP Storage*, *System protocols* and data transferred to the *FlashTcpBuffer* (i.e. using data transfer mode from serial port) are stored inside non-volatile buffer. Messages in the non-volatile buffer will not be affected by system resets or power downs. The size of the non-volatile is approx. 1MB. The maximum size for one packet: 1KB = 1024 Bytes. If the storage in non-volatile buffer is low, the oldest entries (max. 64KB) will be erased to free the memory. The data transferred to TcpClient is stored inside volatile Buffer (like in Sendmode 0). PFAL commands can be executed when being inside the command mode and their answers are appended to volatile buffer → answers are sent after any data inside volatile buffer is sent, but prior to nonvolatile buffer contents. The device sends only one single packet is at a time; if this packet is confirmed by the server (ack sent), the packet is erased from volatile memory (and a possibly existing next packet can be sent). All packets are transmitted sequentially in the same order they were stored/enqueued (TCP send, TCP Storage, System Protocols). New packets can be written while older ones are transmitted. They will be transmitted when all other packets have been successfully transmitted. <buffer_level> Optional. It determines the number of bytes of data that can be saved in the buffer before being transmitted. Once the <buffer_level> is filled up, the device transfers automatically all the data within the buffer via TCP. Use the command TCP.Client.FlushSendBuffer to transfer the data before the buffer is filled up.
	0...1000000	The number of bytes of data that can be saved.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT.SENDMODE=0 $PFAL,Cnf.Set,TCP.CLIENT.SENDMODE=2,2000
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT.SENDMODE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

When the device resets, all TCP data will be cleared.

It is recommended not to change this setting during a transmission.

5.12.9. TCP.SERVICE.CONNECT

Parameter syntax	TCP.SERVICE.CONNECT=<s_enable>,<ip_address>,<tcp_port>

This setting allows to configure a TCP service connection (2nd server), which allows to run services like remote control, configuration, remote updates, AGPS etc.. without the need to reconfigure a currently used TCP client connection.

Parameter	Value	Meaning
<s_enable>	Refer to chapter 5.12.1. for more details.	
<ip_address>	*Refer to chapter 5.12.1. for more details.*	
<port>	Refer to chapter 5.12.1. for more details.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.SERVICE.CONNECT=1,217.119.194.35,2222
Get Configuration	$PFAL,Cnf.Get,TCP.SERVICE.CONNECT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

This command is not supported in firmware 3.5.0 and later.

Contact your network administrator to specify a correct server IP address and port number.

5.12.10. TCP.SERVICE.TIMEOUT

Parameter syntax	TCP.SERVICE.TIMEOUT=<C-timeout>,<wait_time>

This parameter is used to define the period of time the device will wait for a response and between two connection attempts when the TCP connection fails.

Parameter	Value	Meaning
<C-timeout>	Refer to chapter *5.12.4..* for more details.	
<wait_time>	Refer to chapter *5.12.4.* for more details.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.SERVICE.TIMEOUT=240000,60000
Get Configuration	$PFAL,Cnf.Get,TCP.SERVICE.TIMEOUT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

This command is not supported in firmware 3.5.0 and later.

In general, the timeout should never exceed 10 days (10*24*3600*1000). The standard specification for a TCP timeout is 5 minutes. The timeout above 15 minutes is not recommended.

5.12.11. TCP.STORAGE

Parameter syntax	TCP.STORAGE=size=<value>,dispatch=<mode>

This parameter allows you to specify the size of TCP storage and the operation mode. The data in the TCP storage will be lost after resetting the device.

Parameter	Value	Meaning
<value>	It specifies the size in bytes of the TCP storage to be set. It must be an integer number between **1** and **4096**. The TCP storage holds the data that should be transferred by means of the TCP storage. By means of ***TCP.Storage.Dispatch*** command the contents of data in the TCP storage can be moved to an internal outgoing TCP buffer when the TCP storage is created. If there is no enough memory available to satisfy a TCP storage requirement, AVL device will report an error upon attempting to start ***TCP.Storage.Dispatch*** command.	
<mode>	Currently the TCP storage supports two operation modes:	
	manual	If this mode is selected, the TCP storage has to be dispatched manually (i.e. you have to determine when to send the stored information via TCP).
	auto	This mode allows the system to dispatch automatically the data inside the TCP storage whenever it is used up.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.STORAGE=size=512,dispatch=manual
Get Configuration	$PFAL,Cnf.Get,TCP.STORAGE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.12.12. TCP.SMTP.CONNECT

Parameter syntax	TCP.SMTP.CONNECT=<enable>,<mail_server_address>,<mail_server_port>

This parameter allows you to configure the sending of E-Mail via an Internet mail server. This parameter opens a connection to the SMTP server using the <mail_server_address> and <mail_server_port> values when the AVL device attempts to send E-Mail messages.

Parameter	Value	Meaning
<enable>	Enables/disables the sending of E-Mail via an Internet mail server. Following values can be used:	
	0	Disables sending of E-Mail.
	1	Enables sending of E-Mail.
<mail_server_address>	It specifies a string used to identify the address of the remote computer system. It can contain an IP address in dotted-decimal form, such as "129.3.1.24", or a computer name, such as ***"smtp.mail.yahoo.com"***. Sending email from your AVL device over GPRS will be connecting using the GPRS service of your network operator, and will need to use your ISPs SMTP server, not their SMTP server. As an example, if you use **D1** as your GPRS network operator and **yahoo.com** as your email provider, you will not be able to send email using ***smtp.o2.co.uk***, you would need to use ***smtp.mail.yahoo.com*.**	
<mail_server_port>	It is an Integer value used to identify the port number on the remote computer system in <mail_server_address>.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.SMTP.CONNECT=1,217.119.194.35,2222
Get Configuration	$PFAL,Cnf.Get,TCP.SMTP.CONNECT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

All values are separated by comma ",".

Contact your ISP provider to specify a correct outgoing mail server address and port number.

5.12.13. TCP.SMTP.SUBJECT

Parameter syntax	TCP.SMTP.SUBJECT=<subject_text>

This parameter allows you to identity the authentication for the SMTP session when sending E-Mail to an Internet mail server. Set the Username and Password values before performing the ***TCP.SMTP.Send*** command. If the Username or Password is invalid, or blank and the Username and Password are required, the authentication to the remote mail server fails.

<subject_text>

It specifies the user text as a string (recommendation: don't exceed 200 characters) which may also contain dynamic variables.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.SMTP.SUBJECT ="Hallo"
Get Configuration	$PFAL,Cnf.Get,TCP.SMTP.SUBJECT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

This setting is optional. In case it is defined, user text will be used to create all sent email subjects. If the firmware default setting is used, the subject is extended to grant backward compatibility. It will have the format „Message from <hardware_type> rev:<hw_revision> Name=&(DeviceName), IMEI=&(IMEI)“.

<**hardware_type**>: Hardware type of the device (i.e. FOX3).
<**hw_revision**>: Hardware revision of the device.

5.12.14. TCP.SMTP.LOGIN

Parameter syntax	TCP.SMTP.LOGIN=<"domain">,<timeout>,<"username">,<"password">

This parameter allows you to identity the authentication for the SMTP session when sending E-Mail to an Internet mail server. Set the Username and Password values before performing the ***TCP.SMTP.Send*** command. If the Username or Password is invalid, or blank and the Username and Password are required, the authentication to the remote mail server fails.

Parameter	Value	Meaning
<“domain”>	String type. It specifies the domain name from the Internet address. For example ***"fal.de"***.	
<timeout>	Timeout specifies the amount of time (in seconds) to wait for a response from the socket before the current operation is aborted.	
<“username”>	Username specifies a string of your email account that contains the authentication identity provided when using the authentication mechanisms for the SMTP client in the authenticate process.	
<“password”>	Password specifies a string (max. 20 characters) of your email account that contains the authentication credentials provided when using the authentication mechanisms for the SMTP in the authenticate process.	

Show the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.SMTP.LOGIN ="fal.de",30,"UserID","Password"

Get Configuration	$PFAL,Cnf.Get,TCP.SMTP.LOGIN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.12.15. TCP.SMTP.FROM

Parameter syntax	TCP.SMTP.FROM=<"fromaddress">

This parameter allows you to configure the E-Mail address of the sender of the message.

<"fromaddress">

String email formatted. It specifies the E-Mail address of the sender of the message. Please, enter a valid email address. If your email address is incorrect or invalid your massage may be cancelled.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.SMTP.FROM="user4@yahoo.com"
Get Configuration	$PFAL,Cnf.Get,TCP.SMTP.FROM

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.12.16. TCP.SLA

Parameter syntax	TCP.SLA=<mode>

This setting activates or deactivates the SLA (Service License Agreements) connections to the Lantronix SLA-Server. This SLA applies to the following services:

- Firmware update on request
- Activation of PREMIUM-FEATURES

This feature is available since firmware version 2.13.0_rc11. If the SLA connection is activated, the AVL device will try to connect to the Lantronix SLA-Server only during an existing GPRS connection. The transfer data is kept as small as possible approximately 80 bytes per connection and the transferred data is protected by AES encryption.

Note: *It is strongly recommended not to change this setting, as this prevents Lantronix to provide maintenance and/or services for this device. If the customer would like to have*

this setting disabled, then this setting must be mentioned in the CPS document (Customer-Product-Specification) as: TCP.SLA=0.

Parameter	Value	Meaning
<mode>	Defines the SLA transmission mode.	
	0	Disables SLA connection.
	1	Enables every 30 days SLA connections (**default**).
	2	Enables every 24 hours SLA connections.
	3	Enables only one SLA connection (used for the activation only).
	4[1]	Enables permanent SLA connection.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.SLA=1
Get Configuration	$PFAL,Cnf.Get,TCP.SLA

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	✓	✓	✓	✓

5.12.17. TCP.CLIENT2.CONNECT

Parameter syntax	TCP.CLIENT2.CONNECT=<s_enable>,<ip_addr>,<port>

This parameter specifies a second TCP port setup. Includes the connection type, IP-address and port that your application will use to connect to the remote server. AVL device actively initiates a connection to a remote server when GPRS state is online. Firstly, set the IP-address and Port number and then execute manually (if <s_enable> is set to 0) the "**TCP.Client2.Connect**" command to connect to the server, as shown in chapter *4.8.4.4.* When the connection has been established successfully, the "TCP.Client2.eConnected" event occurs. If the remote server rejected the connection, an Error event occurs. After a connection has been established, use the **TCP.Client2.Send,<protocols>,<"text">** command to stream data to a remote server. A "**TCP.Client2.ePacketSent**" and a "TCP.Client2.eReceived" event occurs when there is outgoing and incoming data accordingly. If the <s_enable>=0, use the "TCP.Client2.Disconnect" command to terminate the connection, otherwise set the <s_enable>=0 before using the "TCP.Client2.Disconnect" command.

Parameter	Value	Meaning
<s_enable>	Activates/deactivates automatically connection to the remote server. Following values can be used:	
	0	Deactivates automatically connection to the remote server.
	1	Activates automatically connection to the remote server. If the TCP connection to the remote server gets lost, the AVL device will automatically attempt to reconnect as soon as the remote server will be available again.
<ip_addr>	IP address in dotted-four-byte format. It is the IP address to which the AVL device will be registered and send its data. The format of this address is "xxx.xxx.xxx.xxx". It is also possible to specify a DNS address, max. 255 chars, in the hostname form (for example **www.lantronix.com**) instead of a dynamic or static IP-address. In such a case it is not required to set up any DNS provider, it is automatically done.	

Notes

- The port number has to be specified in ANY case.
- Using DNS can cause much traffic if the specified domain does not exist (the device keeps requesting of the IP-address until one is returned)
- The AVL device clears automatically the implemented DNS cache (inside the unit) if the server refuses that connection immediately (it deletes an old DNS entry that maybe not up to date. During the next reconnection the AVL device retrieves automatically a new IP-address and can continue normal operation)
- Note that, if the server login fails (i.e. the server closes the TCP connection without requiring any acknowledge), DNS query process will be restarted during the next reconnection.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT2.CONNECT=1,217.119.194.35,2222
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT2.CONNECT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

All values are separated by comma ",".

Contact your network administrator to specify a correct server IP address and port number.

5.12.18. TCP.CLIENT2.ALTERNATIVE

Parameter syntax	TCP.CLIENT2.ALTERNATIVE=<a_enable>,<ip_addr>,<port>,<timeout>

This parameter defines an alternative TCP port for the second TCP connection in case the primary server fails. This allows to define a fallback server which can be used in case the primary server gets disconnected/doesn't respond anymore. If both servers aren't available, the device continues

to attempt a connection (alternately to defined "primary" and "alternative" server). A wait time between 2 connection attempts can be specified using the setting **TCP.CLIENT2.TIMEOUT**.

Parameter	Value	Meaning
<a_enable>	Enables/disables automatically connection to the remote server. Following values can be used:	
	0	Disables automatically connection to the alternative server (**default**).
	1	This Server will be used for each second connection attempt.
<ip_addr>	IP address in dotted-four-byte format. It is the IP address to which the AVL device will be registered and send its data. The format of this address is "xxx.xxx.xxx.xxx". See also chapter *2.3.* It is also possible to specify a DNS address in the hostname form (for example **www.lantronix.com**) instead of a static IP-address. In such a case it is not required to set up any DNS provider, it is automatically done.	
<port>	Specifies the port number used for communication between the AVL device and alternative remote server. See also chapter *2.3.*	
<timeout>	Optional. Defines an independent TCP timeout for the alternative server. If no timeout is specified, then ***TCP.CLIENT2.TIMEOUT*** setting is used.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT2.ALTERNATIVE=0,217.119.194.35,2222 $PFAL,Cnf.Set,TCP.CLIENT2.ALTERNATIVE=1,www.lantronix.com,2222,300000
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT2.ALTERNATIVE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

All values are separated by commas ","

Contact your network administrator to specify a correct server IP address and port number.

5.12.19. TCP.CLIENT2.SENDMODE

Parameter syntax	TCP.CLIENT2.SENDMODE=<mode>[,<buflevel>]

This parameter configuration allows to select between fast and safe TCP transmissions for the second TCP connection.

Note: *It is strongly NOT recommended to change this setting when alarms are active (during regular device operation). Restart the device after this mode is changed!*

Otherwise loss of stored data is possible (i.e. volatile or non-volatile stored packets).

Furthermore, it may cause data to be transmitted partial.

Only one TCP channel can be configured with SENDMODE=2.

TCP.CLIENT.SENDMODE=2

or

TCP.CLIENT2.SENDMODE=2

Parameter	Value	Meaning
<mode>	It specifies how fast TCP packets should be sent. It can be set to:	
	0	**Safe and volatile transmission mode (default).** Safe transmission prevents losing TCP packets during TCP/GPRS reconnects. Only **one** single packet is sent at a time. Outgoing TCP packets sent with ***MSG.Send.TCP, TCP.Client2.Send*** and any data transferred using data transfer mode from the serial port are stored inside volatile memory (40KB size) and will be erased by system when it performs a reset and/or power down. When the server has confirmed a packet with "ack sent", the next packet is sent. Even if a TCP timeout occurs no packets can get lost. PFAL commands can be executed when being inside the command mode and answers are appended to the volatile buffer and sent after any data inside the buffer is sent.
	1	Fast and volatile transmission mode. It shall NOT be used (use for testing purposes only). - robust communication cannot be guaranteed - stop of communication has been observed on some networks (stuck communication which requires manual device reset); might be solved by future firmware versions - several packets are sent at a time - if a TCP/GPRS timeout happens, sent but not yet acknowledged packets could get lost - packets are stored inside volatile memory and will be erased by system resets and/or system power down.
	2[,<buffer_level>]	Safe and non-volatile transmission mode. Outgoing messages sent with *MSG.Send.TCP*, *TCP.Client2.Send* and *TCP Storage*, *System protocols* and data transferred to the *FlashTcpBuffer* (i.e. using data transfer mode from serial port) are stored inside non-volatile buffer. Messages in the non-volatile buffer will not be affected by system resets or power downs. The size of the non-volatile is approx. 1MB. The maximum size for one packet: 1KB = 1024 Bytes. If the storage in non-volatile buffer is low, the oldest entries (max. 64KB) will be erased to free the memory. The data transferred to TcpClient is stored inside volatile Buffer (like in Sendmode 0). PFAL commands can be executed when being inside the command mode and their answers are appended to volatile buffer → answers are sent after any data inside volatile buffer is sent, but prior to nonvolatile buffer contents. The device sends only one single packet is at a time; if this packet is confirmed by the server (ack sent), the packet is erased from volatile memory (and a possibly existing next packet can be sent). All packets are transmitted sequentially in the same order they were stored/enqueued (TCP send, TCP Storage, System Protocols). New packets can be written while older ones are transmitted. They will be transmitted when all other packets have been successfully transmitted. <buffervel> Optional. It determines the number of bytes of data that can be saved in the buffer before being transmitted. Once the <bufferevel> is filled up, the device transfers automatically all the data within the buffer via TCP. Use the command TCP.Client2.FlushSendBuffer to transfer the data before the buffer is filled up.
	0...1000000	The number of bytes of data that can be saved.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT2.SENDMODE=0 $PFAL,Cnf.Set,TCP.CLIENT2.SENDMODE=2,2000
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT2.SENDMODE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

When the device resets, all TCP data will be cleared.

It is recommended not to change this setting during a transmission.

5.12.20. TCP.CLIENT2.PING

Parameter syntax	TCP.CLIENT2.PING=<type>,<time>

It allows you to activate/deactivate the maximum idle time until the next ping has to be sent to the remote server.

Parameter	Value	Meaning
<type>	It can be set to:	
	0	Deactivates sending of pings to the remote server.
	1	Activates sending of pings to the remote server (it is recommended). These pings are used to ensure the remote server that the AVL device is still present in an active TCP connection.
<time>	It specifies the amount of time, in milliseconds, on which a ping will be sent to the remote server.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT.PING=1,120000
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT.PING

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.12.21. TCP.CLIENT2.TIMEOUT

Parameter syntax	TCP.CLIENT2.TIMEOUT=<timeout>,<retry_wait>

This parameter is used to define the connect and reconnect timeouts for the second TCP connection. The timeout is the period of time the device will wait for a response and between two connection attempts when the TCP connection fails.

Parameter	Value	Meaning
<timeout>	Specifies the period of time, in milliseconds, the target device will wait for a response *(an acknowledgement)* from the remote server about the received data and the next data transmission.	
<retry_wait>	Specifies the length of time, in milliseconds, the AVL device waits between two connection attempts. Please note, the **TCP** <timeout> value must always be smaller than the **GPRS** <G_timeout> value, otherwise the AVL device closes the GPRS connection before trying to perform a TCP connection to the remote server.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT2.TIMEOUT=240000,60000
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT2.TIMEOUT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

In general, the timeout should never exceed 10 days (10*24*3600*1000).

The standard specification for a TCP timeout is 5 minutes. The timeout above 15 minutes are not recommended.

5.12.22. TCP.CLIENT2.LOGIN.EXT

Parameter syntax	TCP.CLIENT2.LOGIN.EXT=<"text ">

It allows you to add additional information to the regular login data that the device will send to the remote server after establishing the second TCP connection.

Parameter	Value	Meaning
<text>	Wrapped in quotation marks (" "), specify up to 100 characters. The specified string can be used as an extension to the login data that the AVL device will send to the TCP server after it is connected. The text sent to the server will look like: `$<MSG.Info.ServerLogin>` `$DeviceName=unnamed FOX3` `$Ext=example extension text` `$Software= ..........` The user may also specify any dynamic protocol. Each dynamic variable is separated by ampersand "&" without spaces and enclosed in parentheses "( )". Dynamic variables are listed in chapter 7. For example: $PFAL,Cnf.Set,TCP.CLIENT.LOGIN.EXT="&(OwnNumber)"	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT2.LOGIN.EXT="example extension text"
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT2.LOGIN.EXT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.12.23. TCP.CLIENT2.LOGIN

Parameter syntax	TCP.CLIENT.LOGIN=<login>,[<security>]

It allows you to activate sending of login data automatically to the used remote server after the TCP connection is established for the second TCP connection.

Parameter	Value	Meaning
<login>	It can be set to:	
	0	Deactivates sending of login information to the remote server.
	1	Activates sending of login information to the remote server. It sends the same data as the "" message described in chapter *4.8.1.*This data is sent after the AVL device has successfully established a TCP connection to the remote server. The login data can then be used to ensure the connection of the AVL device to the remote server.
	2	Activates sending of login information to the remote server in a new format (shrink format) that is supported by the firmware release 2.13.0 and above. It sends the same data as the "" message described in chapter *4.8.1.*This data is sent after the AVL device has successfully established a TCP connection to the remote server. The login data can then be used to ensure the connection of the AVL device to the remote server.
[<security>]	PREMIUM-FEATURE AES_TCP will be required to enable and use this setting. Current implementation without PREMIUM-FEATURE contains fixed AES key (128bit 0-bits) and 2 modes of operation. If no parameter is specified the default setting will be used (Default = no AES).	
	1	AES encryption (ECB mode)
	2	AES encryption (CBC mode)
	3	No AES encryption
	0	No AES encryption

How the configuration could be set/requested:

Set Configuration	$PFAL,CNF.Set,TCP.CLIENT2.LOGIN=1,0 $<MSG.Info.ServerLogin> $DeviceName=Unnamed AVL $Security=0 $Software=avl_3.0.0 (BxBGT1gzIHJldjowMy1OVUNIAgEA) $Hardware=FOX3 rev:03-NUCH $LastValidPosition=$GPRMC,143445.000,A,5040.4096,N,01058.8542,E,0.01,0.00,040315,, $IMEI=353816054739497 $PhoneNumber=+491773456789 $LocalIP=10.208.151.168 $CmdVersion=2 $SUCCESS $<end> $PFAL,CNF.Set,TCP.CLIENT2.LOGIN=2,0 $<MSG.Info.ServerLogin> $DeviceName=Unnamed AVL $Software=avl_3.0.0_rc20 $Hardware=FOX3 rev:03-NUCH $IMEI=353816054739497 $PhoneNumber=+491773456789 $Position=$GPRMC,143545.000,A,5040.4086,N,01058.8543,E,0.00,0.00,040315,, $SUCCESS $<end> $PFAL,CNF.Set,TCP.CLIENT2.LOGIN=2,2 $<MSG.Info.ServerLogin> $DeviceName=Unnamed AVL $Software=avl_3.0.0_rc20 $Hardware=FOX3 rev:03-NUCH $IMEI=353816054739497 $PhoneNumber=+491773456789 $Position=$GPRMC,144217.000,A,5040.4086,N,01058.8543,E,0.00,0.00,040315,, $Security=2 $SUCCESS $<end>
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT2.LOGIN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.12.24. TCP.CLIENT2.DNS.TIMEOUT

Parameter syntax	TCP.CLIENT2.DNS.TIMEOUT=<dns_cache_timeout>

AVL device has the feature to cache the DNS records for a fixed period of time. This parameter allows to control caching of DNS records for the second TCP connection. The DNS cache consists of a fixed portion of SRAM memory. The DNS cache is lost whenever the AVL device is switched off and no user-backup-battery is already connected.

<dns_cache_timeout>

Specifies the DNS cache timeout. The length of time (in seconds) to keep the DNS cache valid. After the time expires, the DNS memory cache will be updated with new data. It can be set to a value from **0** to **2147483647**.

Setting the DNS cache timeout to **0** results that the AVL device always performs a new DNS query process.

Setting the DNS cache timeout to **2147483647** results that AVL device each **49.7** days (converted from seconds to days) performs a new DNS query process. By default, it is set to **86400** resulting each **1** day the AVL device performs a new DNS query process.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,TCP.CLIENT2.DNS.TIMEOUT=86400
Get Configuration	$PFAL,Cnf.Get,TCP.CLIENT2.DNS.TIMEOUT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

However, a DNS query process will automatically be performed, whenever the AVL device tries to perform a TCP connection and the TCP connection establishment fails.

After a system restart the TCP messages in the non-volatile buffer will only be lost if the device configuration is set to TCP.CLIENT2.SENDMODE=0. To retain these messages in the non-volatile buffer you have to set the device configuration to TCP.CLIENT2.SENDMODE=2.

5.12.25. MQTT.CLIENT.CONNECT

Parameter syntax	MQTT.CLIENT.CONNECT=<s_enable>,<ip_address>,<tcp_port>

This parameter specifies the connection type, IP-address and port that your application will use to connect to the MQTT server/broker. AVL device actively initiates a connection to a server when GPRS state is online. Firstly, set the IP-address and Port number and then execute manually (if <s_enable> is set to 0) the "TCP.MQTT.Connect" command to connect to the server, as shown in chapter *4.8.5.1.* When the connection has been established successfully, the "MQTT.Client.eConnected" event occurs. If the server rejected the connection, an Error event occurs. After a connection has been established, use the TCP.MQTT.Send,"<topic>@<message>" command to send messages to a server. A "MQTT.Client.ePacketSent" event occurs when there is outgoing data accordingly.

Parameter	Value	Meaning
<s_enable>	If the <s_enable>=0, use the "TCP.MQTT.Disconnect" command to terminate the connection, otherwise set the <s_enable>=0 before using the "TCP.MQTT.Disconnect" command. It can be set to:	
	0	Deactivates automatically connection to the server.
	1	Activates automatically connection to the server. If theTCP connection to the server gets lost, the AVL device will automatically attempt to reconnect as soon as the server will be available again.

Parameter	Value	Meaning
<ip_address>	IP address in dotted-four-byte format. It is the IP address to which the AVL device will be registered and send its data. The format of this address is "xxx.xxx.xxx.xxx". Seealso chapter 2.3.. It is also possible to specify a DNS address, max. 255 chars, in the hostname form (for example www.lantronix.com) instead of a dynamic or static IP-address. In such a case it is not required to set up any DNSprovider, it is automatically done.	
<tcp_port>	Specifies the TCP port number used for communication between the AVL device and remote server. See also See also chapter 2.3..	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,MQTT.CLIENT.CONNECT=1,217.119.194.35,2222
Get Configuration	$PFAL,Cnf.Get,MQTT.CLIENT.CONNECT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.12.26. MQTT.CLIENT.PING

Parameter syntax	MQTT.CLIENT.PING=<time>,<topic>@<message>

This parameter allows you to activate/deactivate the maximum idle time until the next ping is sent to the server.

Parameter	Value	Meaning
<time>	Specifies the amount of time, in milliseconds, on which a ping is sent to the server. If this value is 0 (zero) no ping message is sent.	
<topic>@<message>	Defines the subscription and the message that is sent to the server in case of inactivity.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,MQTT.CLIENT.PING=30000,$aws/things/&(ThingID)/shadow/update@{\'state\':{\'reported\':{\'latlng\':\'&(LatLon)\'}}}
Get Configuration	$PFAL,Cnf.Get,MQTT.CLIENT.PING

5.12.27. MQTT.CLIENT.TIMEOUT

Parameter syntax	MQTT.CLIENT.TIMEOUT=<conn_timeout>,<wait_time>

This parameter is used to define the period of time the device will wait for a response and between two connection attempts when the TCP connection fails.

Parameter	Value	Meaning
<conn_timeout>	Specifies the period of time in milliseconds, the target device will wait for a response (an acknowledgement) from the server about the received data and the next data transmission.	

Parameter	Value	Meaning
<wait_time>	Specifies the length of time, in milliseconds, the AVL device waits between two connection attempts. Please note, the TCP <C-timeout> value must always be smaller than the GPRS <G_timeout> value, otherwise the AVL device closes the GPRS connection before trying to perform a TCP connection to the remote server.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,MQTT.CLIENT.TIMEOUT=240000,60000
Get Configuration	$PFAL,Cnf.Get,MQTT.CLIENT.TIMEOUT

5.12.28. MQTT.CLIENT.SENDMODE

Parameter syntax	MQTT.CLIENT.SENDMODE=<mode>

This parameter allows you to select between fast and safe TCP transmissions.

Note: *It is NOT recommended that you change this setting when alarms are active (during regular device operation). Reconnect the device after this mode is changed. Otherwise, loss of stored data can occur (volatile or non-volatile stored packets). Further, it may cause data to be transmitted partially.*

Parameter	Value	Meaning
<mode>	It specifies how fast TCP packets should be sent. It can be set to:	
	0|1	Volatile transmission mode (default): Safe transmission prevents loss of TCP packets during TCP/GPRS reconnects. A single packet is sent at a time. Outgoing TCP packets sent with MSG.Send.MQTT, MQTT.Client.Send and any data transferred using data transfer mode from the serial port is stored inside volatile memory (32KB size) and is erased by the system when it performs a reset and/or power down. When the server confirms a packet with "ack sent", the next packet is sent.
	2	Non-volatile transmission mode: Outgoing messages are sent with MSG.Send.MQTT, MQTT.Client.Send and TCP Storage, system protocols and data transferred to the FlashTcpBuffer are stored inside non-volatile buffer. Messages in the non-volatile buffer will not be affected by system resets or power downs. The size of the non-volatile buffer is approximately 256 KB. The maximum size for one packet: 1KB = 1024 Bytes. If the storage in non-volatile buffer is low, the oldest entries (max. 64KB) are erased to free memory. The device sends a single packet at a time; if this packet is confirmed by the server ("ack sent"), the packet is erased from volatile memory (and a possibly existing next packet can be sent). All packets are transmitted sequentially in the same order they were stored/enqueued (TCP.MQTT.Send). New packets can be written while older ones are transmitted. They will be transmitted when all other packets have been successfully transmitted.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,MQTT.CLIENT.SENDMODE=0
Get Configuration	$PFAL,Cnf.Get,MQTT.CLIENT.SENDMODE

5.12.29. MQTT.CLIENT.WELCOME

Parameter syntax	WLANMQTT.CLIENT.WELCOME=<topic>@<message>

Allows you to send a welcome message to the server after a successful connection.

<topic>@<message>

Defines the subscription and the message that is sent to the server after successful connection.

Set Configuration	$PFAL,Cnf.Set,MQTT.CLIENT.WELCOME=$aws/things/&(ThingID)/shadow/update@{\'state\':{\'reported\':{\'tcxn\':{\'connection_status\':2,\'imei\':\'&(IMEI)\'}}}}
Get Configuration	$PFAL,Cnf.Get,MQTT.CLIENT.WELCOME

5.12.30. MQTT.CLIENT.LASTWILL

Parameter syntax	MQTT.CLIENT.LASTWILL=<topic>@<message>

In MQTT, you can use the "Last Will and Testament" feature to notify other clients about an ungracefully disconnected client. This configuration allows you to set the last will message for an unexpected device disconnect.

<topic>@<message>

Defines the last will subscription and message that will be sent to the server.

5.12.31. MQTT.CLIENT.ID

Parameter syntax	MQTT.CLIENT.ID=<ThingID>

In MQTT, devices are registered and accessible through a thing ID.

<ThingID>

Defines the registered thing ID of the device.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,MQTT.CLIENT.ID=5640
Get Configuration	$PFAL,Cnf.Get,MQTT.CLIENT.ID

5.12.32. MQTT.CLIENT.USER

Parameter syntax	MQTT.CLIENT.USER=<user[:pwd]>

Enter a user name and password for client authentication.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,MQTT.CLIENT.USER=t76543210/user0:password0
Get Configuration	$PFAL,Cnf.Get,MQTT.CLIENT.USER

5.13. WLAN

Please refer to the chapter *4.2.17.* for more details about the WLAN.

The following parameters are used to initialize a "known networks list". All of the five parameters must be set for a specific index. If one of the parameters is missing, the configuration for this network will be initialized as invalid. If an entry is changed at runtime the internal list will be

updated. If all entries for one network get valid, the network will be used at a network scan. Maximum 5 WLAN networks can be configured (id 0-4).

5.13.1. WLAN.MODE

Parameter syntax	WLAN.MODE=<mode>

This parameter is used to set the mode to connect to a WLAN access point.

Parameter	Value	Meaning
<mode>	Specifies the mode to be used for WLAN connections:	
	off	Disables connections to WLAN access points.
	scan	Enables connections to WLAN access points with command WLAN.Scan.
	Connect,<index>	Enables connections to a WLAN access point with command WLAN.Connect,<index>

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,WLAN.MODE=Connect,1
Get Configuration	$PFAL,Cnf.Get,WLAN.MODE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.13.2. WLAN.CLIENT.LOGIN

Parameter syntax	WLAN.CLIENT.LOGIN=<state>

This parameter is used to set the login mode when connecting to a TCP server via WLAN.

Parameter	Value	Meaning
<state>	Enables/disables sending of ServerLogin when the connection to the TCP server is established. Following values can be used:	
	0	Disables sending of ServerLogin.
	1	Enables sending of ServerLogin.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,WLAN.CLIENT.LOGIN=1
Get Configuration	$PFAL,Cnf.Get,WLAN.CLIENT.LOGIN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.13.3. WLAN.NET<id>.SSID

Parameter syntax	WLAN.NET<id>.SSID=<ssid>

This parameter is used to set the SSID (real name) of the access point that IOBOX-WLAN will connect to, even when the access point is hidden (meaning, its SSID isn't broadcast publicly).

Parameter	Value	Meaning
<id>	Specifies the index to save the WLAN configuration profile.	
	0..4	Available indices
<ssid>	String with ASCII printable characters with a max. length of 32 characters. Defines the SSID of the access points.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,WLAN.NET0.SSID=AP_Intern $PFAL,Cnf.Set,WLAN.NET1.SSID=AP_Drivers
Get Configuration	$PFAL,Cnf.Get,WLAN.NET0.SSID

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.13.4. WLAN.NET<id>.TYPE

Parameter syntax	WLAN.NET<id>.TYPE=dhcp
Parameter syntax	WLAN.NET<id>.TYPE=static,<own_ip>,<netmask>,<gateway_ip>,<dns_ip>

This parameter is used to set the type of connection to access the IOBOX-WLAN to a wireless access point by either using DHCP or a static IP address depending on the configuration of that access point. For instructions on how to setup this connection type, please ask the network administrator for the appropriate IP settings.

Parameter	Value	Meaning
<id>	Specifies the index to save the WLAN configuration profile.	
	0..4	Available indices
<own_ip>	Sets the static IP address within the subnet of the access point's IP. For example, the IP of access point is 192.168.0.2, an acceptable value for an IP address that has to be set in <own_ip> would be between 192.168.0.3 – 192.168.0.254. Only the last number changes and can be any value between 3 and 254 (e.g. 192.168.0.200)	
<netmask>	Sets the subnet mask of the access point (e.g. 255.255.255.0).	
<gateway_ip>	Sets the gateway IP address of the access point (e.g. 192.168.0.1)	
<dns_ip>	Sets the DNS (Domain Name Server) IP address of the access point (e.g. 192.168.0.1)	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,WLAN.NET0.TYPE=dhcp $PFAL,Cnf.Set,WLAN.NET1.TYPE=static,192.168.0.200,255.255.255.0,192.168.0.1,192.168.0.1
Get Configuration	$PFAL,Cnf.Get,WLAN.NET0.TYPE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.13.5. WLAN.NET<id>.PSK

Parameter syntax	WLAN.NET<id>.PSK=<psk>

This parameter is used to set the WLAN network pre-shared key (PSK) to connect to.

Parameter	Value	Meaning
<id>	Specifies the index to save the WLAN configuration profile.	
	0..4	Available indices
<psk>	Sets the pre-shared key (PSK) of the WLAN network to connect to with a max. length of 63 characters.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,WLAN.NET0.SECURITY=123456789
Get Configuration	$PFAL,Cnf.Get,WLAN.NET0.SECURITY

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.13.6. WLAN.NET<id>.SECURITY

Parameter syntax	WLAN.NET<id>.SECURITY=<security_level>

This parameter is used to set the encryption method of data transmission between IOBOX-WLAN and WLAN network.

Parameter	Value	Meaning
<id>	Specifies the index to save the WLAN configuration profile.	
	0..4	Available indices
<security_level>	Sets the security protocol used to authenticate the IOBOX-WLAN on that network. Following options are available:	
	WEP	Wired Equivalent Privacy, a wireless network security standard.
	WPA	Wireless Protected Access, a wireless network security standard.
	WPA2	Second version of the WPA standard.
	WPA2 MIX	WPA2 mixed mode.
	None	No Security.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,WLAN.NET0.SECURITY=WPA2
Get Configuration	$PFAL,Cnf.Get,WLAN.NET0.SECURITY

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.13.7. WLAN.NET<id>.IP

Parameter syntax	WLAN.NET<id>.IP=<ip>

This parameter is used to set the IP address of the remote server that the IOBOX-WLAN will use to connect to after successful connection to the WLAN network.

Parameter	Value	Meaning
<id>	Specifies the index to save the WLAN configuration profile.	
	0..4	Available indices
<ip>	IP address in dotted-four-byte ("xxx.xxx.xxx.xxx") format. It is the IP address to which the device will connect and send its data.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,WLAN.NET0.IP=217.119.194.35
Get Configuration	$PFAL,Cnf.Get,WLAN.NET0.IP

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.13.8. WLAN.NET<id>.PORT

Parameter syntax	WLAN.NET<id>.PORT=<port>

This parameter is used to set the TCP port of the remote server that the IOBOX-WLAN will use to send the data after successful connection to the WLAN network.

Parameter	Value	Meaning
<id>	Specifies the index to save the WLAN configuration profile.	
	0..4	Available indices
<port>	Specifies the port number that is used to send the data.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,WLAN.NET0.PORT=2222
Get Configuration	$PFAL,Cnf.Get,WLAN.NET0.PORT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.13.9. WLAN.RSSIMIN

Parameter syntax	WLAN.RSSIMIN=<level>

This parameter is used to scan for networks with RSSI levels greater than the user defined <level>.

Parameter	Value	Meaning
<level>	Defines the min. RSSI level for a stable WLAN connection. WLAN networks with RSSI levels greater than this will be scanned and listed.	
	0..255	RSSI level

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,WLAN.RSSIMIN=35
Get Configuration	$PFAL,Cnf.Get,WLAN.RSSIMIN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.13.10. WLAN.MODE

Parameter syntax	WLAN.MODE=<mode>

This parameter is used to define and store the startup connection mode of the IOBOX WLAN.

Parameter	Value	Meaning
<mode>	Defines the automatic connection mode.	
	off	Automatic connection mode is turned off.
	Scan	Automatic connection mode after the scan.
	Connect<id>	Automatic connection mode with profile <id>.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,WLAN.MODE=Connect,1
Get Configuration	$PFAL,Cnf.Get,WLAN.MODE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	?	?	?	×

5.14. BLE

5.14.1. BLE.ADVNAME

Parameter syntax	BLE.ADVNAME=<Advertized_friendly_name>

This parameter is used to set up the name of the FOX3-3G-BLE devices for using BLE features.

<Advertized_friendly_name>

Defines the name of the FOX3-3G-BLE used for BLE applications. Default assigned name is FOX3-3G-BLE.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,BLE.ADVNAME=FOX3-3G-AVL
Get Configuration	$PFAL,Cnf.Get, BLE.ADVNAME

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only in FOX3-3G BLE variant.

5.14.2. BLE.WHITELIST

Parameter syntax	BLE.WHITELIST=<attribute>

This parameter is used to set the mode of the BLE whitelist after scanning the iBeacon sensors.

Parameter	Value	Meaning
<attribute>	Defines the mode of the whitelist after scanning the iBeacon sensors. It can be set to:	
	None	Disables checking the attributes of iBeacon sensors on the whitelist
	Public	After scan it shows only iBeacon sensors advertising their name(s).
	Name	After scan it shows all iBeacon sensors that advertise and not advertise their names. iBeacon Sensors that do not advertise their names are shown as “unnamed”
	MAC	After scan it shows the advertised MAC address(es) of scanned iBeacon sensors.

How the configuration could be set/requested

Set Configuration	$PFAL,Cnf.Set,BLE.WHITELIST=Name
Get Configuration	$PFAL,Cnf.Get,BLE.WHITELIST

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only in FOX3-3G BLE variant.

5.14.3. BLE.SCANDURATION

Parameter syntax	BLE.SCANDURATION=<duration>

This parameter is used to set the duration in which the FOX3-3G-BLE device stays in the scanning state to scan for nearby BLE iBeacons that are advertising. Every time a device is discovered during the FOX3-3G-BLE is scanning, the event SYS.BLE.eRegister=<"id_name"> occurs and when a BLE iBeacon is lost, the event SYS.BLE.eRelease=<"id_name"> occurs.

Parameter	Value	Meaning
<duration>	Defines the duration time in seconds in which the FOX3-3G-BLE device stays in the scanning state. It can be set to:	
	5…60	The duration time in seconds.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,BLE.SCANDURATION=10
Get Configuration	$PFAL,Cnf.Get,BLE.SCANDURATION

* Starting with firmware version 3.1.0_rc33

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only in FOX3-3G BLE variant.

5.15. BLUEID

5.15.1. BLUEID.DEVID

Parameter syntax	BLUEID.DEVID=<"friendly_name">,<"soid">

This parameter is used to set the advertised Name for a BLE device with integrated BlueID Library.

Parameter	Value	Meaning
<"friendly_name">	Defines the name of the FOX3-3G-BLE used for BLE applications. Default assigned name is FOX3-3G-BLE.	
<"soid">	The Secured Object ID is a result of registration of a device at the BlueID secured server backend required for creating a BlueID ticket.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,BLUEID.DEVID="FOX3-3G-BLE-BID","BlueID ticket"
Get Configuration	$PFAL,Cnf.Get,BLUEID.DEVID

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only in FOX3-3G BLE variant.

5.15.2. BLUE.KEY1

Parameter syntax	BLUEID.KEY1=<"public_key">

This parameter is used to set the BlueID public key for decrypting operations inside the BlueID Library.

<"public_key">

It is a result of registration a device at the BlueID secured server backend.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,BLUEID.KEY1="BlueID public key"
Get Configuration	$PFAL,Cnf.Get,BLUEID.KEY1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

Exception: This command is supported only in FOX3-3G BLE variant.

5.15.3. BLUE.KEY2

Parameter syntax	BLUEID.KEY2=<"private_key">

This parameter is used to set the BlueID private key for decrypting operations inside the BlueID Library.

<"private_key">

It is a result of registration a device at the BlueID secured server backend

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,BLUEID.KEY2="BlueID private key"
Get Configuration	$PFAL,Cnf.Get,BLUEID.KEY2

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	×	×	×	×

5.16. UDP

If you do not have read the differences between the two Internet protocols UDP and TCP, please refer to the chapter *5.12.*

5.16.1. UDP.CLIENT.CONNECT

Parameter syntax	UDP.CLIENT.CONNECT=<s_enable>,<ip_address>,<tcp_port>

This parameter specifies the type, IP-address and port that your application will use to connect to the remote server via UDP protocol. An AVL device actively initiate a connection to a remote server only when it is GPRS attached.

Parameter	Value	Meaning
<s_enable>	Specifies the index for the WLAN configuration settings.	
	0	Deactivates automatically connection to the remote server.
	1	Activates automatically connection to the remote server. If the UDP connection to the remote server gets lost, the AVL device will automatically attempt to reconnect as soon as the remote server will be available again.
<ip_address>	Defines either a static IP address in dotted-four-byte format or a dynamic DNS address. It is the address to which the AVL device will send its data. The format of a static IP address is "xxx.xxx.xxx.xxx". See also chapter *2.3.*. It is also possible to specify a DNS address in the hostname form (for example ***www.lantronix.com***).	

Parameter	Value	Meaning
<port>	Specifies the port number used for communication between the AVL device and remote server. See also See also chapter 2.3..	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,UDP.CLIENT.CONNECT=1,217.119.194.35 ,2222
Get Configuration	$PFAL,Cnf.Get,UDP.CLIENT.CONNECT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

- The port number has to be specified in ANY case.
- Using DNS can cause much traffic if the specified domain does not exist (the device keeps requesting of the IP-address until one is returned).
- The AVL device clears automatically the implemented DNS cache (inside the unit) if the server refuses that connection immediately (it deletes an old DNS entry that maybe not up to date. During the next reconnection the AVL device retrieves automatically a new IP-address and can continue normal operation).

5.16.2. UDP.CLIENT.TIMEOUT

Parameter syntax	UDP.CLIENT.TIMEOUT=<U_timeout>

This parameter specifies the allowed timeout between reconnections to the remote server, when the TCP connection fails.

<U_timeout>

Specifies the period of time, in milliseconds, the target device will wait for a connection response from the remote server. Please note, the ***TCP*** <U_timeout> value must always be smaller than the ***GPRS*** <G_timeout> value, otherwise the AVL device closes the GPRS connection before trying to perform a TCP connection to the remote server.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,UDP.CLIENT.TIMEOUT=240000
Get Configuration	$PFAL,Cnf.Get,UDP.CLIENT.TIMEOUT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

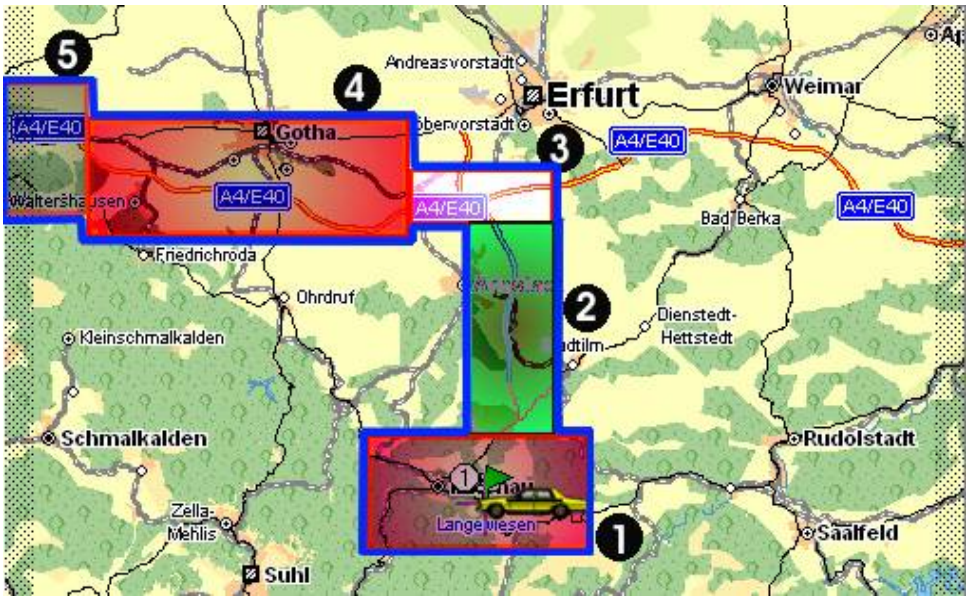
5.17. GF

This section describes how the Geofence works and how to set up a Geofence to the AVL device. It is assumed that the users have a basic understanding of conditional logic and geographic coordinates.

How to do Geofence with the AVL devices:

The Geofence is a term used to describe an event when the vehicle fitted with a GSM/GPS unit places an electronic rectangle coordinates or a circle around your vehicle. Once a geo-fence is established, users can be automatically notified, as a result of event occurring, if a vehicle enters and/or leaves the user pre-defined area(s)/Geofences. This functionality can be used for territory management, route verification, arrival/departure notification and prohibited locations. Exception reporting can also be applied to a wide variety of additional events, such as arrivals, departures, deliveries, pick-ups, illegal entries, unauthorized movement, and more. The AVL device based on the GPS system recognizes if the vehicle crosses a user-defined geographic boundary, therefore, a SMS alert is issued or other services can be used for notification. The constructed form of geographic boundary zones (Geofences) may be either a rectangular or circular one in different sizes but the smallest size is recommended 20 x 20 meters. The figure below shows possibilities of defining Geofences within an area (the area's boarder is colored blue).

Figure 5-2 Possibilities of defining Geofences within an area



Determine the Zone's Grid Coordinates:

The AVL device firmware supports one kind of coordinate format for latitude and longitude (decimal format). The supported format is based on the output of NMEA protocol format.

- Latitude, longitude (in decimal format).

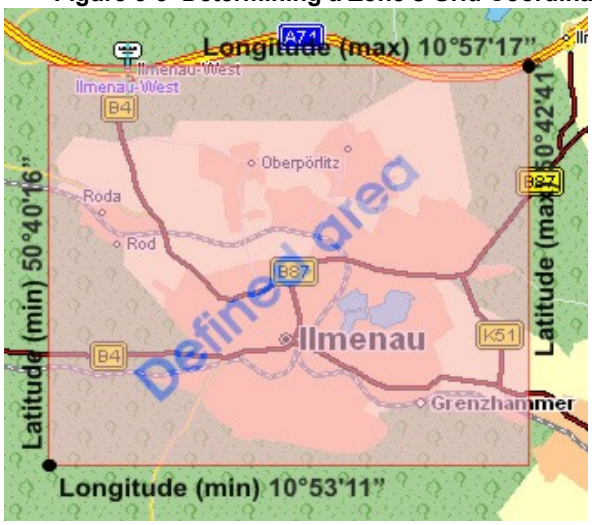
$GPRMC,141128.638,A,50.712222,N,10.881944,E,0.07,103.22,280104,,*3E

If you use a mapping software (Map&Guide or another one) to determine the rectangle coordinates of the Geofences to be set, then open it and locate the coordinate text that displays the geographic coordinates of your cursor location on the map:

For the Map&Guide: by moving the mouse cursor on the map the current coordinates (Longitude/Latitude in degrees, minute and second) are displayed at the bottom corner on the first panel of the viewer window.

Locate the zone you want to define on the map, and note down the **min** longitude/Latitude and the **max** longitude/Latitude coordinates of a rectangle that defines the zone/Geofence, as shown in the figure below.

Figure 5-3 Determining a Zone's Grid Coordinates



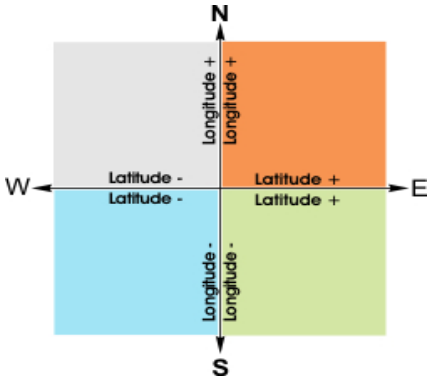
In our example (in degrees, minutes, seconds):

```
Longitude (min) = 10°53’11" E
Latitude (min) = 50°40’16" N
Longitude (max) = 10°57’17" E
Latitude (max) = 50°42’41" N
```

In order to convert coordinates from degrees, minutes, seconds format into decimal format, refer to chapter *11.4.*

After you have determined the grid coordinates for your rectangular zone/Geofence, one last step before setting up the configuration is to determine the sign of the coordinate values. Please remember that, Latitude North and Longitude East are positive and Latitude South and Longitude West are negative (*see also figure below*):

Figure 5-4 Determining the sign of coordinate values



Set up the Geofencing zones and areas:

Set up the configuration as normal based on the parameter syntax, and enter the parameters as indicated in chapter *5.17.3.* The set values of **GF3** are obtained from chapter .

```
$PFAL,Cnf.Set,GF.Area4="Ilmenau-City"
$PFAL,Cnf.Set,GF3=
area<flag>,"name",R,<La_Min>,<Lo_Min>,<La_Max>,<Lo_Max>
$PFAL,Cnf.Set,GF3=
area<flag>,"Ilmenau",R,50.6711,10.8863,50.7113,10.9547
```

An established Geofence always generates two different events:

```
When the vehicle enters into the pre-defined Geofence boundaries.
When the vehicle leaves the pre-defined Geofence boundaries.
```

An area supports also two different events:

```
When the vehicle enters into the area boundaries*.
When the vehicle leaves the area boundaries*.
* Geographic boundaries are automatically generated from a
collection of set Geofences.
```

For our example:

"*GPS.Geofence.e1=outside*" and **"*GPS.AREA.e4=outside*"** events occur when the Latitude of the vehicle's current position is bigger than 10°57'17" AND smaller than 10°53'11" AND the Longitude of the vehicle's current position is bigger than 50°42'41" AND smaller than 50°40'16".

"*GPS.Geofence.e1=inside*" and **"*GPS.AREA.e4=inside*"** events occur when the Latitude of the vehicle's current position is smaller than 10°57'17" AND bigger than 10°53'11" AND the Longitude of the vehicle's current position is smaller than 50°42'41" AND bigger than 50°40'16".

Note: *The AVL device can store up to 100 Geofences (and up to 3000 with Premium feature "Geofence-Extended" activated), and up to 32 areas where one of them is parking area. The parking area is pre-defined and is not configurable; it can only be activated using the command in chapter 4.5.3.1. he ID-number of the parking area is "0".*

5.17.1. GF.CONFIG

Parameter syntax	GF.CONFIG=maxdop=<max_dop>,parkradius=<park_radius>,changes=<changes>

This parameter manages the global configuration of the Geofence functionalities. Based on this configuration all events occurred within Geofence range are controlled.

Parameter	Value	Meaning
<max_dop>	This setting specifies the allowed maximal PDOP (GPS position accuracy error) value for Geofencing. If this value is exceeded, no Geofence events will be generated. The background is: If DOP exceeds the defined limit, positioning errors increase more and more, which usually result in wrong geofence or area events. Therefore, Geofence checks are performed only when the PDOP value is inside the defined range (i.e. 5 means approx. 50 meters maximal error). This means only if GGA is inside range, updated geofence states are available. If e.g. a geofence is configured while PDOP is out of range, its status will be set initially to outside.	
<park_radius>	It defines the radius in meter of the circular area to be used as Geofence park area. The center point is the AVL location. It places the AVL device (vehicle) into a restricted circle zone (park radius), where the current position (including Latitude and Longitude) of the AVL device is the center of circle and the <park_radius> is the radius of circle.	

Parameter	Value	Meaning
<changes>	This parameter may be set to „on“ or „off“, which can be used to prevent occurring of any Geo-fence (**GF.e<*ID*>**) event. The area events will be displayed always.	
	on	Enables occurring of the Geofence events (**GF.e<ID>**).
	off	Disables occurring of the Geofence events (**GF.e<ID>**). (It does not affect the AREA events. These events always will be occurred)

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GF.CONFIG=maxdop=5,parkradius=200,changes=on
Get Configuration	$PFAL,Cnf.Get,GF.CONFIG

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.17.2. GF.AREA<id>

Parameter syntax	GF.AREA<ID>=<"text">">

This parameter allows the AVL device to set up to **32** areas in a range of **0** to **31**. An area may be a Geofence or a collection of up to 100 separate Geofences. Area boundaries are automatically increased or decreased in size according to the set of the size of setup Geofences. As reference, see also figure attached in chapter *5.17.3.* Each area consists of two events ("*GPS.Area.e<id>=inside*" and "*GPS.Area.e<id>=outside")* and two states ("*GPS.Area.s<id>=inside*" and "*GPS.Area.s<id>=outside")* that automatically rise whenever the corresponding area<> is entered or left respectively. If an area includes more than one overlapped GeoFences, then no area event occurs while the AVL device is moving within the set GeoFences within that area. GeoFences and their size can be set up and (re-)configured using the "*GF<id*" parameter. Also, several names of set areas can be linked to available ID-numbers of the set areas.

Parameter	Value	Meaning
<ID>	It specifies the ID-number of an area. It is used to separate areas and to group Geofences. Up to 32 areas in range of **0** to **31.** The configuration is stored in the on-board FLASH memory of the FOX3.	
<"text">	It defines the text for an area index that will be displayed together also with each area event. For example, "***AREA.e0=outside***" without quotation marks. The maximal length of string to be specified is limited to **50** characters.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GF.AREA0="park area" $PFAL,Cnf.Set,GF.AREA1="City East" $PFAL,Cnf.Set,GF.AREA2="City West " $PFAL,Cnf.Set,GF.AREA3="City Centre" $PFAL,Cnf.Set,GF.AREA4="City"

Get Configuration	$PFAL,Cnf.Get,GF.AREA0 $PFAL,Cnf.Get,GF.AREA1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

The text of the GF.AREA0="park area" parameter may be redefined, however the GF.AREA0 should not be used as an area for other Geofences, if the parking area is set/activated (see chapter *4.5.3.1.* for more details).

5.17.3. GF<id>

Parameter syntax	GF<id>=area<flag>,<"name">",<shape>

This parameter allows you to set up to 100 GeoFences in arrange of 0 to 99, to define the form of GeoFences and place them to a specific area(s) (*5.17.2.*). The purpose of GeoFences is that all entries and exits of the target object equipped with AVL device unit into or out of a preset area trigger events and based on these events the user can be notified. The AVL device can hold up to 100 GeoFences setup either as a rectangle or as a circle. The coordinates of each GeoFences can be obtained from any map software, and then converted (if needed) into the format supported by the AVL device *(please, refer to chapter 11.4. for more details how to convert them)*. The AVL device stores the user specified configuration into the on board FLASH memory, and this configuration will be available until overwritten. Each Geofence consists of two events and two states that automatically rise whenever the corresponding Geofence is entered or left. Several GeoFences can be set up into a single area, which allow you to define and cover up a very complex moving direction. In this way the AVL device can move through several GeoFences, but within a single area, without occurring of any area events. Additionally, if within an area are placed several GeoFences, then a part of each set GeoFences must overlap its neighbor in the moving direction, otherwise area events will rise when a Geofence is entered or left. Overlapping of these GeoFences prevent occurring of area events or alarm notification while the AVL device moves within the set GeoFences of that area. Several GeoFences may also be added to several areas *(if certain regions in the map belong to different areas, see* Example 2 *for more details)*.

Parameter	Value	Meaning
<ID>	It specifies the ID-number of a Geofence. It is used to separate GeoFences. Up to 100 Geofences, in range of **0** to **99**. A Geofence<> can be set/attached to several **areas<**ID**>** if the setup conditions allow it. GeoFences are automatically stored for use in the on-board FLASH memory of the AVL device and these will be available all the time until manually updated or deleted.	

Parameter	Value	Meaning
<flag>	The <flag> is used to place/attach specific **Geofence<**ID**>** to one or several **areas<**ID**>**. The string "area" preceding the <flag> is predefined and currently (in this version) must be written in small letters. GeoFences could not work alone; they work only in connection with areas. That means, if you specify a Geofence you have to determine the area where this Geofence will be placed. The <flag> consists of a 32 bit (4 byte) hexadecimal number in the range 0x**0 …** 0x**FFFFFFFF**. Each bit in this value stands for a single area (0 – 31). For example, if you configure an area ***GF.AREA10="Street 21",*** the <ID> 10 mean that bit **10** from **32** is set high and this value corresponds to hexadecimal value of **0x400 (**^=10000000000$_2$).	
	Format	Range Value in binary format (32-bit)
	BIN	0000 0000 0000 0000 0000 0000 0000 0000$_2$ … 1111 1111 1111 1111 1111 1111 1111 1111$_2$
	Each bit shown in table above consists of a single area (0 – 31). Before you start the configuration process of GeoFences, make sure you have done the configuration of the areas that will cover up these GeoFences. After the user has set up the configuration into the AVL device, it starts comparing its current location with the location of set areas and GeoFences based on the **area<**flag**>** value. If there are disagreements with the user set configuration alarm, you will be automatically notified when alarms are executed. Examples (Example1 and Example 2) will help you have a better understanding of areas and GeoFences configuration settings.	
<"name">	String type, up to **30** characters, it specifies the name of a Geofence. The specified name must be wrapped in quotation marks (" ").	
<shape>	It is used to define and create the shape for GeoFences. The shape can be either a square/rectangle or a point with radius zone (circle). The easiest way to obtain the coordinates of shapes is to use a mapping software. As reference, see chapter *5.17.* ***Note:*** *The number of fractional digits for the Longitude/Latitude value is limited to 5 digits, like "**50.67340**".*	
	C,<lat>,<lon>,<alt>,<radius>	Creates a circle shape <lat> Specifies the latitude, in decimal format. <lon> Specifies the longitude, in decimal format. <alt> Specifies the altitude, in meter. It corresponds to the center of the circle. <radius> Specifies the radius of the circle, in meter. The smallest radius size is recommended 200 meters. When the device enters into such Geofencing zones, the firmware compares the calculated angle between the center of the specified circle (longitude, latitude) and altitude with the current position of the device rather than the given radius value. That means if the altitude is given wrong and does not match the altitude where the circle is set, results a wrong calculation of the radius, so false alarms may be activated.

Parameter	Value	Meaning
	C2 <lat>,<lon>,<radius>	Creates a circle shape. <lat> Specifies the latitude, in decimal format. <lon> Specifies the longitude, in decimal format. <radius> Specifies the radius of the circle, in meter. The smallest radius size is recommended 200 meters. When the device enters into such Geofencing zones, the firmware compares the calculated angle between the center of the specified circle (longitude, latitude) with the current position of the device rather than the given radius value. That means if the altitude is given wrong and does not match the altitude where the circle is set, results a wrong calculation of the radius, so false alarms may be activated.
	R<la_LL>,<lo_LL>,<la_UR>,<lo_UR>	Creates a square/rectangle shape. The smallest size is recommended 20 x 20 meters. Based on the longitude and latitude it allows you to restrict a zone you want to determine on the earth. The upper left corner (LL) and the lower right corner (UR) coordinates of the rectangle are required to define the zone. function Supports <la_LL> Latitude, in decimal format. Specifies the Lower Left corner of value the rectangle. <lo_LL> Longitude, in decimal format. Specifies the Lower Left corner value of the rectangle. <la_UR> Latitude, in decimal format. Specifies the Upper Right corner value of the rectangle. <lo_UR> Longitude, in decimal format. Specifies the Upper Right corner value of the rectangle.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GF1=area8000,"Lantronix Office",R,50.67340,10.98060,50.67350,10.98070 $PFAL,Cnf.Set,GF1=area8000,"Lantronix Office",C,50.67340,10.98060,482,100
Get Configuration	$PFAL,Cnf.Get,GF1

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Example 1:

This example represents how to cover up a driving route using several Geofences within just an area. The purpose of such configurations is to notify users whenever the AVL device enters/exists a Geofence or only when it enters/exits an area - features for arrival/departure notifications. As reference use the diagram attached to this example.

For example you have configure just one area (e.g. GF.AREA4) with following settings:

```
GF.AREA4="City" // corresponds to hex value 10 (1403020100 - 100002)
```

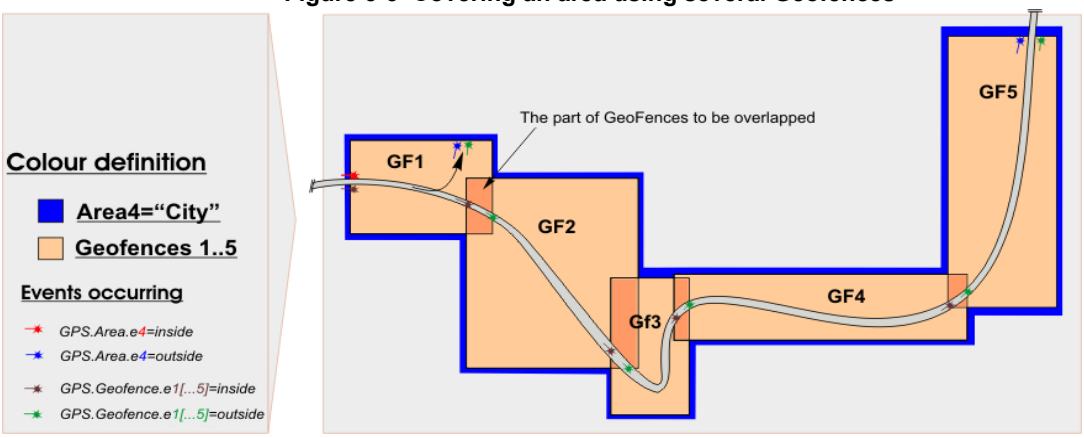
Later on, based on the area settings you may configure the GeoFences that will be attached to.

The defined **GF.AREA4="City"** means that bit **4** is set to high (1) all other bits are low ($\mathbf{1^4 0^3 0^2 0^1 0^0}$). So the Binary value of $\mathbf{10000_2}$ converted to the hexadecimal value is **10,** which will be entered to the **area<**flag**>**.

```
GF1=area10,"B88",R,50.67340,10.98060,50.67350,10.98070
GF2=area10,"Street 1",R,….
GF3=area10,"Street 5",R,….
GF4=area10," B104",R,….
GF5=area10,"Street 256",R,….
```

Setting the value of **area<**flag**>** to **10,** means that all GeoFences **GF1, GF2, GF3, GF4** and **GF5** belong to the single area **GF.AREA4**.

Figure 5-5 Covering an area using several Geofences



The GF.AREA4 boundaries are outlined in the diagram below. Its color definition is given on the left side of the diagram. While the device is moving in or out the GF.AREA4 and GF1 to GF 5 the corresponding in/out events are also occurred.

Once the AVL device unit enters into the first geofence (**GF1)** it enters also into the **GF.AREA4,** so two events and two states are automatically generated "*GPS.Geofence.e1=inside*", "*GPS.AREA.e4=inside*" and "*GPS.Geofence.s1=inside*" and "*GPS.AREA.s4=inside*" respectively. You may connect these events and states to alarm configuration parameter, so that when the AVL device enters into the first geofence (**GF1)** and also into the **GF.AREA4** your application executes the action in the configuration (e.g. SMS notifications, TCP packets, history entries etc.). In the same way you may be notified whenever the AVL device enters/exists a Geofence or only when it enters/exits an area.

Example 2:

This example represents how to cover up a driving route using several Geofences allocated in different areas. The purpose of such configurations is to store the GPS position data in internal memory when the area events rise - features for route verification.

As reference use the diagram attached to this example. Supposed you have configured four areas (*using* GF.AREA*parameter*) with following settings (*see also the figure attached below*).

```
GF.AREA1="City East"// corresponds to hex value 2 (0403021100 -
000102),
GF.AREA2="City West" // corresponds to hex value 4 (0403120100 -
001002),
GF.AREA3="City Centre" // corresponds to hex value 8 (0413020100
- 010002).
GF.AREA4="City" // corresponds to hex value 10 (1403020100 -
100002)
```

Based on the set configuration of the areas you can start the configuration process of GeoFences. For example (some of coordinates are not given):

```
GF1=areaE,"B88",R,50.67340,10.98060,50.67350,10.98070
GF2=areaE,"Street 1",R,….
GF3=area1C,"Street 5",R,…
GF4=area1E," B1004",R,….
GF5=area12,"Street 256",R,….
```

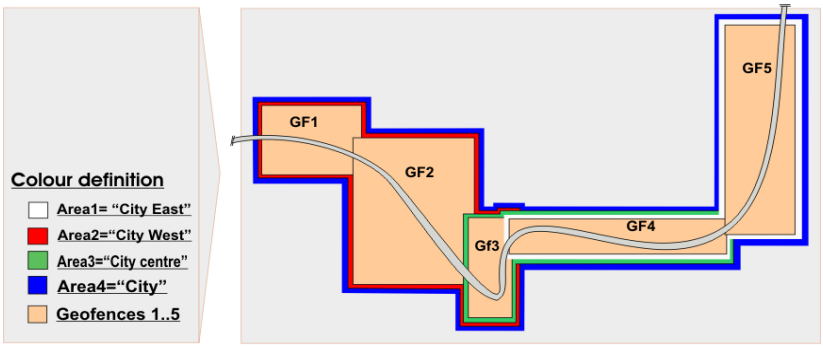
areaE means that the **GF1** and **GF2** belong to both areas **GF.AREA2** and **GF.AREA4.** The hexadecimal value 0x**E** represents the sum of the hexadecimal values **GF.AREA2** (^=0x4) and **GF.AREA4** (^=0x10).

area1C means that the **GF3** belongs to the three areas **GF.AREA2**, **GF.AREA3** and **GF.AREA4.** The hexadecimal value 0x**1C** represents the sum of the hexadecimal values **GF.AREA2** (^=0x4), **GF.AREA3** (^=0x8) and **GF.AREA4** (^=0x10).

area1E means that the **GF4** belongs to the four areas **GF.AREA1**, **GF.AREA2**, **GF.AREA3** and **GF.AREA4.** The hexadecimal value 0x**1E** represents the sum of the hexadecimal values **GF.AREA1** (^=0x2), **GF.AREA2** (^=0x4), **GF.AREA3** (^=0x8) and **GF.AREA4** (^=0x10).

area12 means that the **GF5** belongs to the areas **GF.AREA1** and **GF.AREA4.** The hexadecimal value 0x**12** represents the sum of the hexadecimal values **GF.AREA1** (^=0x2) and **GF.AREA4** (^=0x10).

Figure 5-6 Covering a route using geofences in different areas



This is only a classification of areas and Geofences. The areas boundaries 1 to 4 are outlined in different colors, and their color is given on the left side of the diagram. While the device is moving

in or out an area or Geofence the corresponding event of that area and Geofence always are occurred.

However, depending on your application to be implemented only some of area/geofence events can be set as condition to execute actions.

Notes

The **GF0** should not be used as a standard Geofence zone, if the parking are is set/activated (see chapter *4.7.3.1.* for more details). Based on the user specified <park_radius> value, the **GF0** is automatically set up and attached to the GF.AREA0 parameter, if the command is executed.

5.18. AL - Set alarm configuration

Features of the alarm configuration

- Clearly arranged functional structured alarm conditions.
- Events can be combined to define very complex conditions.
- Several actions can be specified and will be executed when all conditions are true.
- Alias names, which are defined for PFAL commands like Inputs/Output names, Timer, Trigger or Counter names, can be used for conditions as well. This improves clearly arranged easy to understand system configurations.
- Can be extended very flexible and easily.

Limitations

- Currently it is not possible to transmit various system states by using alarms (i.e. sending TCP state periodically via SMS).
- It is not possible to evaluate the execution of alarm actions (whether they are successfully executed or resulting errors). No conditions can be defined to react on failed alarm actions before. (The only way is using i.e. state conditions, which should be changed if the action was successfully executed).
- If two or more alarms with different indexes are triggered at the same time upon a certain event, the alarm with lower index will be executed first.
- Commands/actions within the <action> field will NOT be executed in the same order as you specify them. Timers, Triggers, GSM commands have more priority. Therefore, it is recommended to combine these actions in such a way that no false alarms are being executed.
- PFAL responses cannot be transmitted or displayed for executed alarm commands.
- None of read commands cannot be efficiently used as alarm actions.
  - The purpose of read commands (i.e. all commands which retrieve a state or other information) is to provide information on demand.
  - This information is returned only within the PFAL response, which is not available for alarm commands.
    - To send information automatically use "MSG.Send" commands, see chapter *4.10.1.*

5.18.1. AL<index>= <conditions>:<actions>

Parameter syntax	AL<index>=<conditions>:<actions>

This parameter is intended to configure and store alarms in the AVL unit. The user-specified settings are stored into the FLASH memory and will be available after each device power up.

These alarms allow the AVL device to react and to perform different actions under certain conditions. To allow a very flexible but still structured configuration, each alarm is divided into the conditions and actions. Actions will be executed when all set conditions are evaluated to True.

Example:

An AVL device transmits a TCP packet (including GPS position) to the remote server, when

the level of the digital input 0 changes from Low to High and the event IO.e0=redge occurs.

```
Alarm declaration: AL0=IO.e0=redge:TCP.Client.Send,48,"IN 0 rising edge:"
```

An AVL device sends an SMS to the predefined phone number, when

the GPS position changes from valid to invalid

```
Alarm declaration: AL1=GPS.eFix=invalid:GSM.SMS.Send,"+49123445",0,"location unknown"
```

A - executed command (action), when
E - event raises.

An event is one that only notifies the system that the particular event happened. An action is a command that will be executed when a particular event occurs. An alarm (AL) may contain up to 5 actions/commands and up to 5 conditions. All conditions have an AND- or OR-Conjunction. The block of commands specified within the <actions> field that follow the <conditions> field will be executed once the block of conditions specified within the <conditions> field are evaluated true.

The common syntax of the **AL**-parameter is shown above. The configuration settings for an alarm (AL) can be transferred to and stored in the AVL unit using the ***"$PFAL,***CNF.Set***,AL<index>"*** command. You can combine several configuration settings into a single command line, but it is not recommended. **However, the maximum number of characters specified in a single command line is limited to 1500**. More than 1500 characters will be ignored.

The command syntax when combining several alarm configuration settings in a single command line is:

```
$PFAL,
Cnf.Set,,AL<index>=<conditions>:<actions>;,AL<index>=<conditions>:<actions>;…
```

Various examples can also be found in chapter *11.6.*

Note: *Setting up a large number of complex alarms is not recommended, as it may affect the performance of the device. Therefore, we strongly advise to verify correct behaviour of the system under real term conditions.*

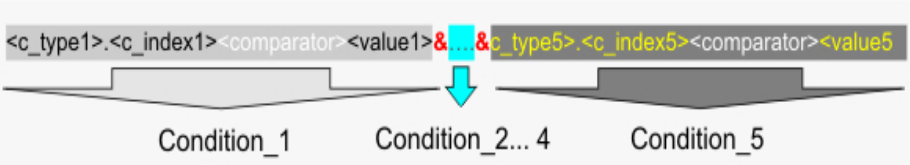
Set Parameter description

Parameter	Value	Meaning
<index>	Defines the alarm index to be configured. The index is a number without leading "0", e.g. **AL0, AL1, AL8, AL10**. The index, for example, **AL05** is invalid. Please note, that alarms with lower index will be first executed, if lower and higher indices are triggered at the same time upon a certain event	
	0 ... 99	Indices for standard alarms
	100 ... 249[1]	Indices for extended alarms (Premium-feature)

Parameter	Value	Meaning
<conditions>	It determines the list of condition(s) to be monitored. More specifically, it determines when the AVL unit should generate an action (alarm). When you are specifying conditions for alarms (AL), to apply the filter to the condition field, use the logical operators such as either "**&**" or "**?**". The logical operators enable an alarm to be more precisely to execute actions/commands when the conditions are met. Up to **5** conditions separated either by ampersand "**&**" or "**?**" without spaces can be specified. The ampersand character "**&**" represents the "**AND**" conjunction which it is used to test whether all set events and states within the condition field are evaluated to True, While the question mark character "**?**" represents the "**OR**" conjunction, which it may be used to test whether at least one of the set events and states within the condition field is evaluated to True.	

Note: *The conjunctions "&" and "?" could not be mixed/nested within a condition field. Within a condition field the user is able to use either "&" or "?"*

The syntax to define several conditions is:



What do conditions mean?

Conditions are system events and/or states, which rise either when a command/action is manually or automatically executed or the device state changes during the system runtime. Conditions are requirements for one or more actions to be executed. As above you can define up to 5 conditions. In that case all user-defined actions are executed, only when the event and states specified within the condition field are evaluated to True.

Additionally, the condition field <conditions> consists of one **event** and/or up to 4 **states**.

What are the differences between Events and States?

Events:

Are a link between occurrences in the system - e.g. the device indicates an incoming voice call or an input changes or a lost GPS signal. More specifically, an event is a pointer that points to a process in a specific case. The conditions containing an event are evaluated just when the event occurs (making it possible to release actions only one time) - so it makes no sense to combine two different events in one condition field. In chapter *6*, all supported **Events** are grouped by major category. The **Events** are represented by adding an **"e"** character directly after the last dot-delimiter [**.**], for example **Sys.Device.eStart** or **Sys.Timer.e1** etc.

States:

Are conditions of the unit checked all the time **(but with low priority[b]),** which makes them ideal as additional condition(s) *to the event. Special care has to be taken when defining alarms which consist of just State conditions. These alarms will be checked with low priority (usually they are checked several times per second, but just, if there are no other pending system operations (like GSM operator search, PFAL commands etc...). If all conditions are true, the defined alarm action*

(commands) are executed one time per second. In worst case this means the defined alarms are executed once per second periodically. It is recommended to prevent such periodical command executions. In chapter *6* all supported **States** are grouped by major category. The **States** are represented by adding an **"s"** character directly after the last dot-delimiter [**.**], for example **Sys.Timer.s1=erased** or **IO.s1=high** etc.

B) Low priority means that such alarms will not be checked periodically when the device performs other tasks (like GSM operator search, PFAL commands, other actions etc...)

If the <conditions> field includes only **STATES** and they are set to True, the AVL unit will permanently execute the specified action, from the field, until the specified **STATES** returns False. The combination of states with an event prevents permanently executing of actions.

Depending on the configuration to be implemented inside the <conditions> field, the events and states, which are available in chapter *6*, can be set.

The following table shows the events and states grouped by major category. Some events and states might have special communication interface requirements and others might have special design considerations. The section number referenced in the table points to the location in chapter *6* of the event or state description and specific information for you to consider when adding these events and states to your application.

Table 5-2 Events and states grouped by major category

Events/States by Major category	Chapter	Description
Sys.eSerialData	*6.1.1.*	Provides events on the serial ports and shows the event signatures.
Sys.eUSBData	*6.1.2.*	Provides events on the USB port and shows the event signatures.
Sys.UserEvent	*6.1.3.*	Gives you an overview about the user events and shows the event signatures.
Sys.CAN	*6.1.4.*	Provides information about the CAN events and shows the event signatures.
Sys.eOBDII	*6.1.6.*	Gives you an overview about the ODBII events and shows the event signatures.
Sys.eFMS	*6.1.7.*	Gives you an overview about the FMS events and shows the event signatures.
Sys.eJ1939	*6.1.8.*	Gives you an overview about the J1939 events and shows the event signatures.
Sys.Device	*6.1.10.*	Provides device events and shows their signatures.
Sys.Timer	*6.1.11.*	Provides events and states of Timer and shows their signatures.
Sys.Trigger	*6.1.12.*	Provides states of the Trigger and shows their signatures.
Sys.Counter	*6.1.13.*	Provides events and states of Counter and shows their signatures.
Sys.nvCounter	*6.1.14.*	Provides events and states of nvCounter and shows their signatures.
Sys.Power	*6.1.15.*	Provides events and states of power and shows their signatures.
Sys.Bat	*6.1.16.*	Provides events and states of Battery and shows their signatures.
Sys.Info	*6.1.17.*	Provides info events and shows their signatures.
Sys.eTimeSync	*6.1.18.*	Provides time synchronisation executed events and shows their signatures.

Events/States by Major category	Chapter	Description
Sys.1Wire	6.1.19.	Provides time 1-Wire events and shows their signatures.
Sys.BLE	6.1.20.	Provides events and states of BLE (Bluetooth Low Energy) and shows their signatures.
Sys.NFC	6.1.21.	Provides events and states of NFC (Near Field Communication) and shows their signatures.
Sys.eDTCO.DRIVER.STATE	6.1.23.	Provides events and states of DTCO-D8 and shows their signatures.
IO.e/s	6.3.1.	Provides events of IO and shows their signatures.
IO.Motion	6.3.2.	Provides events and states of Motion and shows their signatures.
IO.PulseCnt	6.3.3.	Provides events and states of Pulse Counter and shows their signatures.
GPS.eJamming	6.4.1.	Provides events and states of GPS navigation and shows their signatures.
GPS.Nav	6.4.2.	Provides events and states of navigation and shows their signatures.
GPS.Time	6.4.3.	Provides states of GPS Time and shows their signatures.
GPS.History	6.4.4.	Provides History states and shows their signatures.
GPS.Geofence	6.4.5.	Provides Geofence events and states and shows their signatures.
GPS.Area	6.4.6.	Provides Area events and states and shows their signatures.
GPS.WPGF	6.4.7.	Provides Waypoints events and states and shows their signatures.
GPS.eExtAnt(Un)Plugged	6.4.8.	Provides antenna events and shows their signatures (FOX3 only)
GPS.MultiGeofence	6.4.9.	
GSM.eJamming	6.6.1.	Provides GSM operator events and states and shows their signatures.
GSM (Operator)	6.6.2.	Provides events and states of GSM operator and shows their signatures.
GSM.eCellChange	6.6.3.	Provides events and states of cell changes and shows their signatures.
GSM.VoiceCall	6.6.4.	Provides Voice Call events and states and shows their signatures.
GSM.SMS	6.6.5.	Provides SMS events and shows their signatures.
GSM.DataCall	6.6.6.	Provides Data Call events and states and shows their signatures.
GSM.GPRS	6.6.7.	Provides GPRS events and states and shows their signatures.
TCP.Client	6.7.1.	Provides TCP events and states and shows their signatures.
TCP.SMTP	6.7.2.	Provides TCP events and states and shows their signatures.
TCP.UDP	6.7.3.	Provides events and states of UDP and shows their signatures.
WLAN	6.1.22.	Provides events and states of WLAN and shows their signatures.

Use the PDF document bookmarks to navigate the events and states you need.

<actions>

It specifies the commands/action(s) to be executed for a specific task, when the specified conditions (states and event) are True. Up to **5** commands/actions separated by ampersand "**&**" can be specified with some restrictions. The order of the commands/actions in the action field is significant. For details, read important notes listed at the end of this chapter.

The commands/alarms within the action field will NOT be executed in the same order as they are specified*. Timers and Triggers have more priority.* Therefore, it is recommended to combine these actions in such a way that no false alarms are being executed.

To specify an action, please, refer to chapter *Chapter 4: PFAL Commands*. Except the read commands used to read the specific states of the system, all other commands within this chapter can be used as alarm without leading the "$PFAL,".

The syntax to define several actions (commands) is:

Notes

It's not recommended to use read commands inside alarms because this would only slow down system performance. (The information read out can't be displayed this way – use **Msg.Send** commands for such purposes).

- Configuration commands (i.e. **Cnf.Set**)
- It should be used with caution and never being executed in a periodical matter. Each use of such a command drains the lifetime of the internal flash memory. There are several 100000 write/erase cycles for the flash memory, however these could be reached with alarms executed periodically.
- It may only be used as last command within an alarm. Do not write other actions after the **Cnf.Set** command, as these might be not interpreted as additional command action. Instead these commands would be part of the configuration setting.

  How the configuration could be set/ or the settings requested:

Set Configuration	$PFAL,Cnf.Set,AL0=Sys.Device.eStart:IO4.Set=hpulse,3000 $PFAL,Cnf.Set,AL1=IO.e0=redge:GSM.SMS.Send,"+491234567",8,"AL1 SMS" $PFAL,Cnf.Set,AL2=GPS.Nav.eFix=valid:IO4.Set=cyclic,2000,5000 $PFAL,Cnf.Set,AL249….
Get Configuration	$PFAL,Cnf.Get,AL0 $PFAL,Cnf.Get,AL1 $PFAL,Cnf.Get,AL2 $PFAL,Cnf.Get,AL249

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓
1	🗝	🗝	🗝	🗝

Notes

Consider the permanent action execution by specifying only state conditions:

- When an action will be released, the corresponding event or state (if it is specified) is also called by the internal firmware.
- If the AVL unit is configured to release an action via SMS, please, consider that the maximum length of SMS text <text> is predefined up to 160 characters using the 7-bit GSM coding scheme.

5.19. ConsoleFlow Settings

5.19.1. CF.CLIENT.CONNECT

Parameter syntax	CF.CLIENT.CONNECT=<0/1>,<URL>

This setting enables, disables the ConsoleFlow client and allows you to configure the server URL. The value 0 disables and the value 1 enables the client.

Note: *Device has to be rebooted for the setting to take effect.*

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,CF.CLIENT.CONNECT=1,<consoleflow_host_name>
Get Configuration	$PFAL,Cnf.Get,CF.CLIENT.CONNECT

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.19.2. CF.CLIENT.DEVICE_NAME

Parameter syntax	CF.CLIENT.DEVICE_NAME=<device_name>

This setting allows you to configure the device name.

Note: *No spaces are permitted in the <device_name>*

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,CF.CLIENT.DEVICE_NAME=<device_name>
Get Configuration	$PFAL,Cnf.Get,CF.CLIENT.DEVICE_NAME

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.19.3. CF.CLIENT.DEVICE_DESCRIPTION

Parameter syntax	CF.CLIENT.DEVICE_DESCRIPTION=<description>

This settings allows you provide a description for the device

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,CF.CLIENT.DEVICE_DESCRIPTION=<description>
Get Configuration	$PFAL,Cnf.Get,CF.CLIENT.DEVICE_DESCRIPTION

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.19.4. CF.CLIENT.STATUS_UPDATE_INTERVAL

Parameter syntax	CF.CLIENT.STATUS_UPDATE_INTERVAL=<minutes>

This setting allows you to configure the status update interval. This value determines the duration between the status updates sent by the device to the ConsoleFlow server.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,CF.CLIENT.STATUS_UPDATE_INTERVAL=<minutes>
Get Configuration	$PFAL,Cnf.Get,CF.CLIENT.STATUS_UPDATE_INTERVAL

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.19.5. CF.CLIENT.CONTENT_CHECK_INTERVAL

Parameter syntax	CF.CLIENT.CONTENT_CHECK_INTERVAL=<minutes>

This setting allows you to configure the content check interval. This value determines the duration between the checks made by the device for content available at the server, which the device has to process.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,CF.CLIENT.CONTENT_CHECK_INTERVAL=<minutes>
Get Configuration	$PFAL,Cnf.Get,CF.CLIENT.CONTENT_CHECK_INTERVAL

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.19.6. CF.CLIENT.GROUPS_CAP_EXCHANGE_DATA

Note: *The settings 5.19.6, 5.19.7, and 5.19.8 are required if the user wants to publish additional or custom telematic data other than the device's regular telematic data to the ConsoleFlow server.*

Parameter syntax	CF.CLIENT.GROUPS_CAP_EXCHANGE_DATA=<description>

This setting allows you to inform the device, about custom telematic groups that you intend to publish to the ConsoleFlow server.

Example: **$PFAL,CNF.SET,CF.CLIENT.ADDLGRPS_CAP_EXC_DATA={"name": "sample_group","record_name": "Sample Group","record_description": "Description of sample group","record_category": "Sample Group","record_subcategory": "Sample Group"}**

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.19.7. CF.CLIENT.GROUPS_CAP_SELECTION_DATA

Parameter syntax	CF.CLIENT.GROUPS_CAP_SELECTION_DATA=<description>

This setting allows the user to update the meta data of the groups published in the above command.

Example: **$PFAL,CNF.SET,CF.CLIENT.ADDLGRPS_CAP_SEL_DATA={"name": "sample_group","record_name": "Sample Group","record_description": "Description of sample group","detail": [{"name": "sample_param1","record_name": "Sample param1","record_description": "Description of Sample param1","record_type": "string","record_length": 50},{"name": "sample_param2","record_name": "Sample param2","record_description": "Description of Sample param2","record_type": "string","record_length": 50}]}**

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.19.8. CF.CLIENT.GROUPS_TELEMETRY_DATA

Parameter syntax	CF.CLIENT.GROUPS_TELEMETRY_DATA=<description>

This setting allows the user to update telematic data of each parameter with its value.

Example: **$PFAL,CNF.SET,CF.CLIENT.ADDLGRPS_TELE_DATA={"name": "sample_group","detail": [{"name": "sample_param1","value": "value123"},{"name": "sample_param2","value": "88"}]**

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

5.20. Optional Settings

The following configuration settings are created from the system itself and therefore, need not be changed manually. They are just named and explained as a reference.

5.20.1. STORAGE<id>

Parameter syntax	STORAGE<id>

Used to store user data such as current trigger, counter timer or position information to non-volatile memory. These 5 storage slots can be used to keep certain information even when AVL device is shut down. Each storage slot may contain only one user data set (*i.e. a position or a trigger/ counter/timer value*).

Example: **If a timer value is saved to slot 0, only a system timer may load its value and setting from this configuration slot. Any attempt to load e.g. Trigger0 from storage slot 0 will fail as the type of stored data doesn't match.**

However, it is possible to overwrite a storage slot with other user data (e.g. with a position).

<id> Specifies a number from **0** to **4**.

5.20.2. DEVICE.CAN.STARTUP

Parameter syntax	DEVICE.CAN.STARTUP

This setting is used to configure an optional CAN bus interface. It is managed by system and by PFAL commands and shouldn't be modified directly by **CNF.Set** command.

5.20.3. GPS.HISTORY.MODE

Parameter syntax	GPS.HISTORY.MODE=1 // writes each history entry as Full record GPS.HISTORY.MODE=0 // default, writes different history records

If this setting is set to 1 (i.e. $PFAL,Cnf.Set,GPS.HISTORY.MODE=1), then each history entry will be written as Full record *(no Standing, City or Motorway records available)*. If this setting is set to 0 (i.e. $PFAL,Cnf.Set,GPS.HISTORY.MODE=0), the device starts writing of different history records.

Note: *After every change of GPS.HISTORY.MODE, the unit should be rebooted to activate that mode.*

5.20.4. GPS.HISTORY.PUSHMODE

Parameter syntax	GPS.HISTORY.PUSHMODE=500

This setting may be configured with a limitation value representing the fill state (number of bytes) of buffered outgoing TCP data. A History Push will transmit data only of the specified limitation is not exceeded (i.e. if the current buffer fill level is below the specified value). Using this setting, it is possible to prioritize other data and read TCP data with low priority.

5.20.5. DEVICE.CAN.MSG

Parameter syntax	DEVICE.CAN.MSG

This setting is used to configure an optional CAN bus interface. It is managed by system and by PFAL commands and shouldn't be modified directly by **CNF.Set** command.

5.20.6. DEVICE.CAN.VAR

Parameter syntax	DEVICE.CAN.VAR

This setting is used to configure an optional CAN bus interface. It is managed by system and by PFAL commands and shouldn't be modified directly by **CNF.Set** command.

5.20.7. DEVICE.OBD.STARTUP

Parameter syntax	DEVICE.OBD.STARTUP

This setting is used to configure an optional CAN OBDII interface. It is managed by system and by PFAL commands and shouldn't be modified directly by **CNF.Set** command.

5.20.8. DEVICE.FMS.STARTUP

Parameter syntax	DEVICE.FMS.STARTUP

This setting is used to configure an optional CAN FMS interface. It is managed by system and by PFAL commands and shouldn't be modified directly by **CNF.Set** command.

5.20.9. GSM.BANDPREF

Parameter syntax	GSM.BRANDPREF

This setting is managed internally and should be modified only by PFAL commands. It selects at which band the GSM engine starts to search for operators and allows to find operators more quickly when being outside of Europe (which is the default setting).

5.20.10. GSM.PROFILE.AUDIO<prof_index>

Parameter syntax	GSM.PROFILE.AUDIO<id>=<echo_cancel>,<side_tone>,<spk_mute>,<spk_volume>,<mic_mute>,<hfmic_vol>,<hsmic_vol>,<ring_path>,<ring_tone>,<ring_vol>,<path>,<mode>

These settings are used to configure audio profiles. All these settings are managed by system and by PFAL commands and shouldn't be modified directly with ***Cnf.Set*** commands.

Parameter	Value	Meaning
<*id*>	It can be set to a value from **0**..**5.**	
<echo_cancel>	It can be set to a value as follow:	
	0	Disables echo cancelling.
	1	Enables echo cancelling (default)
<side_tone>	It can be set to a value as follow:	
	0	Disables side tones (default)
	1	Enables side tones
<spk_mute>	It can be set to a value as follow:	
	0	Unmutes (activates) the speaker (default)
	1	Mutes (deactivates) the speaker
<spk_volume>	It can be set to a value as follow:	
	0...14	Loudness of the speaker (maximum might depend on GSM version). Default is 7.
<mic_mute>	It can be set to a value as follow:	
	0	Un-mutes (activates) the microphone (default)
	1	Mutes (deactivates) the microphone
<hfmic_vol>	It can be set to a value as follow:	
	0...7	Loudness of the microphone (maximum might depend on GSM version). Default is 5.
<ring_path>	It can be set to a value as follow:	
	0	Automatic path selection (default)
	1	Handsfree audio path
	2	Handset audio path
	3	Internal (not used) audio path

Parameter	Value	Meaning
<ring_tone>	Type of ring tone *(default is 24)*. You have a choice of 32 different ring tones and melodies. All will be played from the audio output	
	1…32	Sequence 1... 32
<ring_vol>	It can be set to a value as follow:	
	0…4	Ringer gain(loudness). Default is **3**.
<path>	It can be set to a value as follow:	
	0	Automatic path selection.
	1	Handsfree audio path (default).
	2	Handset audio path.
	3	Internal (not used) audio path.
<mode>	It can be set to a value as follow:	
	0	Automatic path selection (default).
	1	Handsfree audio path.
	2	Handset audio path.
	3	Internal (not used) audio path.

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,GSM.PROFILE.AUDIO1=1,0,0,7,0,5,5,0,24,3,1,0
Get Configuration	$PFAL,Cnf.Get, GSM.PROFILE.AUDIO1

5.20.11. GSM.PROFILE.CURRENTAUDIO

Parameter syntax	GSM.PROFILE.CURRENTAUDIO

This configuration setting should not be changed manually. Please use PFAL commands instead. Specifies the currently used audio profile. It may only be set to valid profiles. If the specified profile isn't valid (i.e. configured before) this setting will not be updated and return an error.

5.20.12. MACRO<index>

Parameter syntax	MACRO<index>=<command1>&….&command10>

This parameter specifies the macro configuration. A macro can consist of more alarm actions than a usual alarm. It is possible to specify the command e.g. "**Sys.Macro0**" as an alarm action. Thereby, it activates this macro (**0** in our example). If such a macro is activated, all commands defined inside this macro will be executed. **REPLACE** parameter can also be used within commands in a Macro.

Note: *Keep in mind that the MACRO parameter is case sensitive. It must always be written in capital letters, otherwise no action within a MACRO will be executed.*

Macros are designed to make possible executing of the large numbers of commands within a single line. Macro can also be used to store several commands, which can be used frequently inside the alarm configuration. This configuration improves the style of a clearly arranged alarm

configuration; it prevents the configuration mistakes and can ease the readability of a configuration.

Parameter	Value	Meaning
<index>	It specifies the macro index – a number that ranges from 0 to 39 (40 macros in firmware versions 2.16 & 3.1; 10 macros in older firmware versions).	
<command1>&….&command10>	It specifies the command to be released for a specific task. Up to **10** commands separated by ampersand "**&**" can be specified and executed. However, the maximum number of characters in a single command line is limited to 1500. More than 1500 characters will be ignored. To specify a command, please, refer to *Chapter 4: PFAL Commands*. All commands within that chapter can be set as alarms without leading the "$PFAL,". Please do not use any of read commands. Examples in the table below illustrate the use of the MACRO parameter.	

How the configuration could be set/requested:

Set Configuration	$PFAL,Cnf.Set,MACRO0=IO4.Set=hpulse,3000 $PFAL,Cnf.Set,MACRO1=GSM.SMS.Send,"+491234567",8,"AL1 SMS" $PFAL,Cnf.Set,MACRO3=IO5.Set=cyclic,2000,5000
Get Configuration	$PFAL,Cnf.Get,MACRO0 $PFAL,Cnf.Get,MACRO1 …. $PFAL,Cnf.Get,MACRO39

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
PFAL	✓	✓	✓	✓

Notes

The maximum number of commands to be specified within a Macro is limited to **10** and the maximum number of characters in a single command line is limited to **1500**.

- It is not recommended to specify read commands inside a macro because this would only slow down the system performance. (The information to be read out cannot be displayed in this way. Use the **MSG.send** commands for such purposes).
- It is **NOT** allowed to activate other macro commands inside a macro configuration parameter as the activation of them might lead to endless loops of set commands.

Note: *$PFAL,Cnf.Set,MACRO0=IO4.Set=hpulse,3000&Sys.Macro0*

5.20.13. DEVICE.GPS.HEADING

Parameter syntax	DEVICE.GPS.HEADING

This setting is used to configure the GPS heading feature. The settings are managed by system and by PFAL commands and shouldn't be modified directly by ***PFAL.Cnf.Set*** command!

5.20.14. DEVICE.GPS.HEADING2

Parameter syntax	DEVICE.GPS.HEADING2

This setting is used to configure the GPS heading2 feature. The settings are managed by system and by PFAL commands and shouldn't be modified directly by ***PFAL.Cnf.Set*** command

5.20.15. DEVICE.RUPDATE.SELECTION

Parameter syntax	DEVICE.RUPDATE.SELECTION

This setting is managed internally by the system and should be modified only by

Cnf.Set commands. The present of this setting in the configuration indicates that a remote update has been started and not finished/aborted. In this case, history functionality is disabled and the remote update may be resumed. The remote update **finish** and **abort** command will erase this setting, allowing save history functionality again.

5.20.16. DEVICE.WLAN.STARTUP

Parameter syntax	DEVICE.WLAN.STARTUP

This setting is managed internally by the system and should be modified only with the corresponding PFAL commands. The present of this setting in the configuration indicates that a remote update has been started and not finished/aborted. In this case, history functionality is disabled and the remote update may be resumed. The remote update **finish** and **abort** command will erase this setting, allowing save history functionality again.

5.20.17. DEVICE.RUPDATE.SELECTION

Parameter syntax	DEVICE.RUPDATE.SELECTION

This setting is managed internally by the system and should be modified only by Cnf.Set commands. The present of this setting in the configuration indicates that a remote update has been started and not finished/aborted. In this case, history functionality is disabled and the remote update may be resumed. The remote update **finish** and **abort** command will erase this setting, allowing save history functionality again.

5.20.18. DEVICE.CANopen.STARTUP

Parameter syntax	$PFAL,CNF.Set,DEVICE.CANopen.STARTUP=on,<can_device>,<node id> $PFAL,CNF.Set,DEVICE.CANopen.STARTUP=off

This setting is used to enable or disable the CANopen interface.

Refer to ***App Note: CANOpen Gateway Functions for AVL Devices*** for a description of the CANopen gateway functions, configuration, events and states for AVL devices. See *1.3.*

Example: **Example: $PFAL,CNF.Set,DEVICE.CANopen.STARTUP=on,0,5**

Example: **$PFAL,CNF.Set,DEVICE.DTCO.D8=off**

Parameters:

<can_device>:

0-CAN 1-CANB

<node id>:

hexadecimal value for the CANopen node address

5.20.19. WHITELIST.ACTIVE

Parameter syntax	WHITELIST.ACTIVE=<first>,<last>

This setting activates part of the white list using the indices.

Example: PFAL,cnf.set,WHITELIST.ACTIVE=0,3 In this example, the first 4 whitelist entries are now active.

5.20.20. DEVICE.MODBUS

Parameter syntax	DEVICE.MODBUS

This setting is used to configure a ModBus interface. It is managed by system and PFAL commands and should not be modified directly by CNF.Set command.

5.20.21. MODBUS.POLL

Parameter syntax	MODBUS.POLL

This setting is used to configure the ModBus functionality. It is managed by system and PFAL commands and should not be modified directly by CNF.Set command.

6: Events & States

The applications are based on event handling. Events can be generated at run-time or by user actions, such as receiving a voice call, Input changes. Event-driven applications execute action(s) in response to an event or state. If one of these events occurs and it is available in one or more alarm configuration, then the specified action(s) is/are executed. The types of events raised by AVL device vary. For example, an Input raises an event — if it changes its state for low-to-high or vice-versa. Many events occur in conjunction with executed action.

For example:

```
$PFAL,Cnf.Set,AL0=IO.e8=fedge:Sys.Device.Sleep=IGN
```

Whenever the Ignition line performs a falling edge, the **Sys.Device.eShutdown** event raises before going to sleep.

All condition types **<c_type>** and their definition using the *Firmware version 2.6.x are listed below*. Each of them is used to separate the huge amount of conditions to different types. Depending on the alarm configuration to be implemented inside the <conditions> field, the following states and events can be set:

- **Sys** (System) accomplishes a number of system states and events such as:
- System management tasks, including: Reset, Shutdown/power management, etc.
- Initialization/interruption of system processes, including: Timers, Counters, etc.
- **IO** accomplishes a number of input events including I/O events, which can rise when one of them changes its state.
- **GPS** accomplishes a number of GPS events including navigation, history logging and Geofencing data, which can be occurred during AVL device operation.
- **EcoDrive** accomplishes a number of EcoDrive events including, which can be occurred during AVL device operation.
- **GSM** accomplishes a number of GSM events, including SMS, voice and data calls, GPRS attachment/detachment, etc., which can be occurred during AVL device operation.
- **TCP** accomplishes a number of TCP events including connecting disconnection and sending of TCP packets to the predefined address of remote server, etc., which can be occurred during AVL device operation.
- **WLAN** accomplishes a number of WLAN events including, which can be occurred during AVL device operation.

Comparators:

In some conditions the comparators listed in table below are used to compare, i.e. the current speed of the device with a user specified value or the current state of the device inputs, etc.

Table 6-1 Comparators List

Comparator	Operation	Example
=	equality	Sys.Timer.s0=running
==	within the range	IO.s2==10.0:12.5
!=	inequality/out of the range	Sys.Counter.s0!=0 or IO.s2!=0.0:10.0
<	less than	GSM.DataCalll.sRingCounter<2
>	greater than	GPS.Nav.sSpeed>10
<=	less than or equal to	Sys.Counter.s0<=10

Comparator	Operation	Example
>=	greater than or equal to	Sys.Counter.s0>=10
^=	AND - Bit mask check	Compares values read from CAN interface bitwise
^!	NOT - Bit mask check	Compares values read from CAN interface bitwise

For most simple types, comparison is straightforward. For example, **Sys.Timer.s0=running** is True just in case the **Sys.Timer.s0** is **running** and stays True until the Timer 0 changes its state. The following rule applies to comparators.

```
String values can't be compared by using the comparators <,>,
<=,>=, and !=.
In such cases use only the comparator =.
Do not change the order of comparators (<= and >=).
Bitwise comparison allows evaluation of specific bits within
decimal or hex CAN-Bus values.
```

6.1. SYS

6.1.1. Sys.eSerialData

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENT	✓	✓	✓	✓
1	✓	✓	✓	×

STATES		
	State Notification Code	Meaning
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	Meaning
	Sys.eSerialData<port>[=<"text">]	Occurs when the incoming message on the specified serial port starts with the same characters defined in the event text. The specified port must be set either into the event mode with **$PFAL,MSG.Mode.Serial<interface>=6F,E** or into the command mode with **$PFAL,MSG.Mode.Serial<interface>=6F,C** and the received message on the specified port should not start with $PFAL. This event is also occurred when the command **MSG.Event,Serial<port>,<"text">** is executed. To reset the port from event into the command mode, send the following command via SMS or TCP to the device: **$PFAL,MSG.Mode.Serial0=6F,C** // if you send it via SMS, exclude the "$" sign from the command. Example:Sys.eSerialdata0 or Sys.eSerialdata1 Sys.eSerialdata0="AB12" or Sys.eSerialdata1="AB12"

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	Sys.eSerialData<port>.Start[<pos>]=<"text">	Occurs when the incoming message on the specified serial port starts with or includes at the specified position the same characters defined in the event text. The specified port must be set either into the event mode with or into the command mode with and the received message on the specified port should not start with $PFAL. To reset the port from event into the command mode, send the following command via SMS or TCP to the device: $PFAL,MSG.Mode.Serial0=6F,C // if you send it via SMS, exclude the "$" sign from the command. Example:Sys.eSerialdata0.Start="AB12" or Sys.eSerialdata1.Start="DD0" Sys.eSerialdata0.Start4="2A1" or Sys.eSerialdata1.Start7="12ED5"
	Sys.eSerialData<port>.End<endchars>=<"text">	Occurs when the incoming message (max. 100 byte) on the specified serial port ends with the same characters defined in the event text. The specified port must be set either into the event mode with $PFAL,MSG.Mode.Serial<interface>=6F,E or into the command mode $PFAL,MSG.Mode.Serial<interface>=6F,C whereby the incoming message on the specified port should not start with $PFAL. To reset the port from event into the command mode, send the following command via SMS or TCP to the device: $PFAL,MSG.Mode.Serial0=6F,C // if you send it via SMS, exclude the "$" sign from the command. Example: Sys.eSerialdata0.End4="AB12" or Sys.eSerialdata1.End2="AB"
	Sys.eSerialData<port>[End<endchars>]<comp>WhiteList	Occurs when the incoming message on the specified serial port matches one of the entries available in the Whitelist. Entries in the Whitelist can be set with $PFAL,Sys.Whitelist.Set - see chapter *4.2.23.4.*
	<port>	Determines from which serial port the message will be received for the for comparison: **0** = Message will be received from Serial port 0 **1**[1] = Message will be received from Serial port 1
	<pos>	Refers to the position within the incoming text where the comparison will begin.
	<endchars>	Defines the number of characters defined in the <"text">. e.g. "4" for "AB12".
	<"text">	Specifies the text to be compared with the incoming message. The comparison text can be a number or any sequence of characters. The comparison is case-sensitive. The event occurs if the specified event text matches exactly with the incoming message on the specified serial port. It can also show the text (user command specified with the command MSG.Event,Serial >,<"text">.
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=,

6.1.2. Sys.eUSBData

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✗

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS - *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	Sys.eUSBData=[<"text ">]	Occurs when the incoming message on the USB port starts with the same characters defined in the event text or the command MSG.Event,USB,<"text"> is executed. Example: Sys.eUSBData.Start or Sys.eUSBData.Start="AB12"
	Sys.eUSBData.Start[<pos>]=<"text ">	Occurs when the incoming message on the USB port starts with or includes at the specified position the same characters defined in the event text. Example: Sys.eUSBData.Start="AB12" or Sys.eUSBData.Start4="2A1"
	Sys.eUSBData.End<endchars>=<"text ">	Occurs when the incoming message on the USB port ends with the same characters defined in the event text. Example: Sys.eUSBData.End4="AB12"
	Sys.eUSBData[.End<endchars>]<comp>WhiteList	Occurs when the incoming message on the USB port matches one of the entries available in the Whitelist. Entries in the Whitelist can be set with $PFAL,Sys.Whitelist.Set - see chapter *4.2.23.4.* The syntax with [.End<endchars>] is option. If specified, please enter the number of end characters that will be used for the comparison between the entries in the Whitelist and the incoming message on the USB port.
	Sys.eUSBData.TXT=<"text ">	Occurs when an incoming message on the USB port matches the text in the event text. Example: Sys.eUSBData.TXT="abcdef" or Sys.eUSBData.TXT="abcdef"
	<pos>	Specifies the text to be compared with the incoming message. The comparison text can be a number or any sequence of characters. The comparison is case-sensitive. The event occurs if the specified event text matches exactly with the incoming message on the USB port. It can also be the text (user command specified with the command MSG.Event,USB,<"text">.
	<endchars>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=
	<"text">	Specifies the text to be compared with the incoming message. The comparison text can be a number or any sequence of characters. The comparison is case-sensitive. The event occurs if the specified event text matches exactly with the incoming message on the USB port. It can also be the text (user command specified with the command MSG.Event,USB,<"text">.
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=

6.1.3. Sys.UserEvent

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	Meaning
	None	

EVENTS – *are evaluated just when the event occurs*			
	Event Notification Code		Meaning
	Sys.UserEvent.e<index>		Occurs when the corresponded PFAL command is executed. This event can be used to combine/link directly several alarms (AL) i.e. for optimizing larger configurations or simply if more than 5 conditions are needed. To raises such an event, use the corresponding PFAL command and enter the corresponding index number. The UserEvent is not recommended to be used as it allows to create "endless loops" which can slow down the system or even it may affect the stability of other functions. The UserEvents may be use at own risk, however think about all consequences of (maybe recursively) launching alarms when using it. Especially in combination with various states which can itself be influenced by actions, the system behaviour can be very unpredictable and complex. Therefore no support will be given for configurations containing the UserEvents. Example: Sys.UserEvent.e1
		<index>	It is a number, which determines the index of the UserEvent to be set. It can be set to a value from **0** to **9**

6.1.4. Sys.CAN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	🗝	🗝	🗝	✖
1	🗝	🗝	🗝	✖

STATES		
	State Notification Code	Meaning
	Sys.CAN.s<slot>[<comp><value>]	This state checks the currently value of a configured CAN variable of the 1st CAN interface. **Hint:** In order to check a variable when it changes, the CAN variable event should be used. Example: Sys.CAN.s0=4342
	<slot>	It is a number, which determines the slot of the CAN message (see command *4.2.15.6.* Sys.CAN.Var.Add,<slot>,…. for more details). Currently, it can be set to a value from 0 to 49.

STATES		
	State Notification Code	**Meaning**
	[<comp>]	**Optional**. Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <=,>=, ^= or ^!.
	[<value>]	**Optional**. Specifies the value, in decimal, to compare. It ranges from **0** to **216-1.**

EVENTS		
	Event Notification Code	**Meaning**
	Sys.CAN.e<slot>[<comp><value>]	Occurs whenever a configured CAN variable changes its value. Note that, in order to generate such events, the parameter <*notification*> of the specified CAN variable slot must be set to ***"event"***. Example: Sys.CAN.e0=4342 $PFAL,CNF.Set,AL2=SYS.Can.e0^=18:16:Msg.Send.Serial0,0,"can0=&(can0)" $PFAL,CNF.Set,AL2=SYS.Can.e0^!18:16:Msg.Send.Serial0,0,"can0=&(can0)"
	Can.error=<cond>[,<cond>][1]	Occurs when the received error from CAN bus matches the user specified condition(s)
	Can.eStat=<state>[1]	Occurs when the state of the CANbus on the 1st CAN interface (main port) on the device changes from idle to active or vice-versa depending on the specified state.
	<slot>	It is a number, which determines the slot of the CAN message (see command *4.2.15.6.* Sys.CAN.Var.Add,<slot>,…. for more details). Currently, it can be set to a value from 0 to 49.
	[<comp>]	**Optional**. Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <=,>=, ^= or ^!.

EVENTS		
	Event Notification Code	**Meaning**
	[<value>] or [<mask:value>]	**Optional**. The <value> specifies the value, in decimal, to compare. It ranges from **0** to **4294967295**. It is possible to use a bit mask check to easily check the state of individual bits regardless of the other bits. To do this, use the format <mask:value> and specify a bit mask by setting to 0 all the other bits. What makes this convenient is that it is not necessary to figure out what the value actually is, just that it is not 0. Example 1: $PFAL,CNF.Set,AL2=SYS.Can.e0^=18:16:Msg.Send.Serial0,0,"can0=&(can0)" Explanation about : 18 = 00010010 mask in binary 16 = 00010000 value in binary xxx1xx1x Event occurs when the bits 1 and 4 changes to a value of 1xx0 --- Example 2: $PFAL,CNF.Set,AL2=SYS.Can.e0^!18:16:Msg.Send.Serial0,0,"can0=&(can0)" Explanation: 18 = 00010010 mask in binary 16 = 00010000 value in binary xxx1xx1x Event occurs when the bits 1 and 4 changes from 1xx0 to another value
	<cond>	Specified the condition(s) to compare with received error constants received from CAN bus. Separated by commas “,” specify one or a set of conditions: **STUFF** - More than 5 equal bits in a sequence have occurred **FORM** - A fixed format part of a received frame has the wrong format **ACK** - Sent message was not acknowledged by another node **BIT1** - Bit 1 error (monitored bus value was dominant) **BIT0** - Bit 0 error (monitored bus value was recessive) **CRC** - The CRC check sum was incorrect in the message received **WARNING_RX** - TX error counter (TEC) reached warning level (>96) **WARNING_TX** - RX error counter (REC) reached warning level (>96) **PASSIVE** - CAN “error passive” occurred **BUS_OFF** - CAN “bus off" error occurred **OVERRUN_RX** - Overrun in RX queue or hardware occurred **OVERRUN_TX** - Overrun in TX queue occurred **ARBITRATION_LOST** - Arbitration lost **PHY_FAULT** - General failure of physical layer detected (if supported by hardware). **PHY_H** - Fault on CAN-H detected (Low Speed CAN) **PHY_L** - Fault on CAN-L detected (Low Speed CAN)
	<state>	Specifies the state of the CANbus on the 1st CAN interface (main port) on the device. It can be set to one of following supported states: **idle** -the CAN bus is in idle mode when the CAN stops sending data for the user specified time out with command $pfal,sys.can.Timeout,<timeout>. **Active** - the CAN bus is in active mode when the CAN starts sending data for the user specified time out with command $pfal,sys.can.Timeout,<timeout>.

6.1.5. Sys.CANB

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	?	?	?	✗

STATES		
	State Notification Code	Meaning
	None	

EVENTS – *are evaluated just when the event occurs*			
	Event Notification Code		Meaning
	Canb.eStat=<state>		Occurs when the state of the CANbus on the 2nd CAN interface (accessory port) on the IOBOX-CAN device changes from idle to active or vice-versa depending on the specified state.
		<state>	Specifies the state of the CANbus on the 2nd CAN interface (Accessory port) on the IOBOX-CAN device. It can be set to one of following supported states: **idle** -the CAN bus is in idle mode when the CAN stops sending data for the user specified time out with command $pfal,sys.canb.Timeout,<timeout>. **Active** - the CAN bus is in active mode when the CAN starts sending data for the user specified time out with command $pfal,sys.canb.Timeout,<timeout>.

6.1.6. Sys.eOBDII

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	🗝	🗝	🗝	✗

STATES		
	State Notification Code	Meaning
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	Meaning
	Sys.eOBDII=<event_slot>, <sub_event>	This event is generated whenever a configured OBDII event occurs. See configuration reference for more details. Example: Sys.eOBDII=0,0
	Sys.eOBDII.DTC	This event is generated after executing the PFAL command $PFAL,SYS.CAN.OBDII.DTCrq and the requested data by this command is already stored in the dynamic variable "OBDII.DTC". The dynamic variable "OBDII.DTC" contains valid data only when this event occurs. Example: $PFAL,CNF.Set,AL2=Sys.eOBDII.DTC:Msg.Send.Serial0,0,"DTC=&(OBDII.DTC)" The DTC data is available and stored in the dynamic variable "OBDII.DTC only when this event occurs, otherwise no data is available, or the reported data is invalid.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	<event_slot>	Specifies the index of the event slot generating the desired event. See configuration reference for more details.
	<sub_event>	Specifies the event number (index of <eventX>) configured within this event slot. See configuration reference for more details.

6.1.7. Sys.eFMS

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	🗝	🗝	🗝	✖

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	Sys.eFMS=<event_slot>, <sub_event>	This event is generated whenever a configured FMS event occurs. See configuration reference for more details. Example: Sys.eFMS=0,0
	<event_slot>	Specifies the index of the event slot generating the desired event. See configuration reference for more details.
	<sub_event>	Specifies the event number (index of <eventX>) configured within this event slot. See configuration reference for more details.

6.1.8. Sys.eJ1939

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	🗝	🗝	🗝	✖

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	Sys.eJ1939=<event_slot>,<sub_event>	This event is generated whenever a configured eJ1939 event occurs. See configuration reference for more details. Example: Sys.eJ1939=0,0
	<event_slot>	Specifies the index of the event slot generating the desired event. See configuration reference for more details.
	<sub_event>	Specifies the event number (index of <eventX>) configured within this event slot. See configuration reference for more details.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	SYS.eJ1939.TIRE_FAULT<comp><value>	This event is generated whenever the contents of the J1939.TIRE_FAULT variable changes. Example: SYS.eJ1939.TIRE_FAULT=0 or SYS.eJ1939.TIRE_FAULT>0
	SYS.eJ1939.TIRE_STAT<comp><value>	This event is generated whenever the contents of the SYS.eJ1939.TIRE_STAT changes. Example: SYS.eJ1939.TIRE_STAT=0 or SYS.eJ1939.TIRE_STAT>0
	SYS.eJ1939.TIRE_CPC_STAT_HEALTH<comp><value>	This event is generated whenever the contents of the SYS.eJ1939.TIRE_CPC_STAT_HEALTH variable changes. Example: SYS.eJ1939.TIRE_HEALTH>=0
	SYS.eJ1939.TIRE_CPC_STAT_WEX<comp><value>	This event is generated whenever the contents of the SYS.eJ1939.TIRE_CPC_STAT_WEX variable changes. Example: SYS.eJ1939.TIRE_CPC_STAT_WEX>=0
	SYS.eJ1939.TIRE_CPC_STAT_LEARN<comp><value>	This event is generated whenever the contents of the SYS.eJ1939.TIRE_CPC_STAT_LEARN variable changes. Example: SYS.eJ1939.TIRE_CPC_STAT_LEARN>=0
	SYS.eJ1939.TIRE_CPC_TTM_STATE<comp><value>	This event is generated whenever the contents of the SYS.eJ1939.TIRE_CPC_TTM_STATE variable changes. Example: SYS.eJ1939.TIRE_CPC_TTM_STATE>=0
	SYS.eJ1939.TIRE_CPC_TTM_ALARM<comp><value>	This event is generated whenever the contents of the SYS.eJ1939.TIRE_CPC_TTM_ALARM variable changes. Example: SYS.eJ1939.TIRE_CPC_TTM_ALARM>=0
	SYS.eJ1939.TIRE_CPC_TTM_BAT<comp><value>	This event is generated whenever the contents of the SYS.eJ1939.TIRE_CPC_TTM_BAT variable changes. Example: SYS.eJ1939.TIRE_CPC_TTM_BAT>=0
	SYS.eJ1939.TIRE_CPC_TTM_DEFECT<comp><value>	This event is generated whenever the contents of the SYS.eJ1939.TIRE_CPC_TTM_DEFECT variable changes. Example: SYS.eJ1939.TIRE_CPC_TTM_DEFECT>=0
	SYS.eJ1939.TIRE_CPC_TTM_LOSE<comp><value>	This event is generated whenever the contents of the SYS.eJ1939.TIRE_CPC_TTM_LOSE variable changes. Example: SYS.eJ1939.TIRE_CPC_TTM_LOSE>=0
	<comp>	Compares the contents of this variable with the used specified one and return a Boolean (True/False) representing the result of the comparison. It can be set to =, !=, <,>, <= or>=.
	<value>	Specifies the value to be compared.

6.1.9. Sys.eCanDTCO

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	🗝	🗝	🗝	✘

STATES		
	State Notification Code	**Meaning**
	SYS.sCan.DTCO.Confim	True if the incoming APDU message has data.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	SYS.eCan.DTCO.Confirm	This event is generated whenever a APDU message has been sent completely to the tachograph. This is just a confirmation event.
	SYS.eCan.DTCO.Incoming	This event is generated whenever a new incoming APDU message is received $PFAL,CNF.Set,AL1=Sys.eCan.DTCO.Incoming:Msg.Send.Serial0,0,"dtco: &(Can.Dtco.Incoming)"

6.1.10. Sys.Device

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓
1	✓	✓	✓	×
2	✓	✓	✓	×
3	×	×	×	×
4	?	?	?	✓
5	✓	✓	✓	×
6	?	?	?	×

STATES		
	State Notification Code	**Meaning**
	Sys.Device.sStart=[<type>][<reason>]	This state shows the kind of reason for wakeup from sleep mode.
	[<type>]	Uses the same setting as [<type>] for events.
	[<reason>]	Uses the same settings as [<reason>] for events.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	Sys.Device.eStart=[<type>],[<reason>]	Occurs when the device is powered-on, restarts or wakes up from sleep. (The action to be executed can be e.g. Set a LED lights - IO4.Set=hpulse,5000). ***Note:*** *This event can be used to launch actions after the device awakes by a specific wakeup reason. Without start <type> and start <reason> it can be used to initialize certain timers, counters etc.. during system startup (independent from wakeup reason).* Example: Sys.Device.eStart Sys.Device.eStart=PowerUp Sys.Device.eStart=Reset,User3 Sys.Device.eStart=Wakeup,Ring Sys.Device.eStart=Doze

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	[<type>]	An optional startup type can be specified: **PowerUp** Device is powered on externally **Reset** Device is restarted from reset **Wakeup** Device is woken up from sleep **Doze** Device is woken up from doze

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	[<reason>]	Defines an optional startup reason for the specified <type>:

<type>	**[<reason>]**	**Description**
PowerUp	-	No reason is required for PowerUp, leave it empty.
Reset	**Watchdog<id>**	Device rebooted by internal watchdog. The ID ranges from 1 to 6 (index is used for internal in depth analysis only.
	Watchdoglobox[6]	Device rebooted by IOBOX-MINI/CAN/ WLAN watchdog.
	GSM_CCE1	Device rebooted by critical GSM communication error.
	FactoryReset	Device rebooted due to the FactoryReset command.
	RemoteUpdate	Device rebooted due to Remote Update.
	Serial2GSM	Device rebooted due to leaving serial GSM mode.
	UserX	Device rebooted by User using the command **PFAL,SYS.Device.Reset,<user_reset_with timeout>**
	SystemAbort	Device rebooted due to the time out after executing PFAL,Sys.CfgUpdateMode.
Wakeup	**Ign**	Wakeup on Ignition level change.
	Ring [1]	Wakeup when SMS/CALL is received.
	Timer	Wakeup when the timer is expired.
	ExtPwrDetect	Wakeup on detection of external power supply.
	ExtPwrDrop	Wakeup because of external power dropped.
	Motion	Wakeup on change of attitude (Device has been moved)
	AiWu[2]	Wakeup because of specified voltage levels from AiWu were exceeded.
	CAN[3]	Wakeup on CAN-BUS activities.
	LowBat[4]	Wakeup on low battery voltage.
Doze	**RTC**[5]	Wakeup from doze on RTC (Real Time Clock).
	IGN	Wakeup on Ignition level change.
	Ring[1]	Wakeup when SMS/CALL is received.
	CAN	Wakeup by activities on CAN-BUS.
	Timer	Wakeup when the timer is expired.
	Motion	Wakeup on change of attitude (Device has been moved).
	AiWu[2]	Wakeup because of specified voltage levels from AiWu were exceeded.
	Serial1	Wakeup on serial port 1 activities.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	[<reason>], continued	**Idle** Wakeup on UART activities. **LowBat**[4] Wakeup on low battery voltage. **ExtPwrDetect** Wakeup on detection of external power supply. **ExtPwrDrop** Wakeup because of external power dropped
	Sys.Device.eOverVoltage	This event is shown whenever an over-voltage has been detected. Voltages are considered as over-voltage when they exceed 35V. Being continuously over 35V won't launch continuous over-voltage events - it is launched again after voltage drops below approx. 34V and then increases over 35V again. Counter and maximum value shown within the event strings are for information purpose only - these values cannot be triggered or used within a condition.
	Sys.Device.eShutdown=[<wakeup_reason>]	This event is generated for **Sys.Device.Sleep** commands right before sleep mode is entered. Executing **$PFAL,Sys.Device.Shutdown** command shuts down the system immediately - without occurring any shutdown event).
	Sys.Device.ePfalExecuted	This event is shown if: - PFAL execution events are activated (enabled with CMD.PFAL.EN=F,1). - PFAL commands are executed (from any user interface). Using of the dynamic variable &(LastPFALAnswer) allows to create an alarm which sends the last PFAL answer as user customized string to a server (or store it to history).
	[<wakeup_reason>]	An optional wakeup reason can be specified. See PFAL commands **Sys.Device.Sleep** for further details. **Ign** Wakeup on level change of Ignition **Ring** [1] Wakeup on SMS/CALL **Timer** Wakeup when the times is expired **ExtPwrDetect** Wakeup on detection of external power **ExtPwrDrop** Wakeup because of external power dropped **Motion** Wakeup on change of attitude (Device has been moved) **AiWu** [2] Wakeup because of specified voltage levels from AiWu were exceeded.

6.1.11. Sys.Timer

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓
1	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	Sys.Timer.s<index>=<status>	Checks the current operating mode of the timer index. True when the timer <index> is at the specified status <index> until it changes to another status. Example: Sys.Timer.s1=erased Sys.Timer.s1=initialized

STATES			
	State Notification Code	**Meaning**	
	<index>	Specifies the timer identifier. It can be set to a value from 0 to 39 (20-39 [1]).	
	<status>	The state to be compared The various states are not exclusive (e.g. a running timer is always active and initialized). It can be set to.	
		erased	Determines whether the specified timer is currently erased.
		initialized	Determines whether the specified timer is currently initialized.
		active	Determines whether the specified timer is currently started.
		inactive	Determines whether the specified timer is currently stopped.
		running	Determines whether the specified timer is currently running.
		paused	Determines whether the specified timer is currently paused.
		armed	Determines whether the specified timer is currently armed.

	EVENTS – *are evaluated just when the event occurs*	
	Event Notification Code	**Meaning**
	Sys.Timer.e<index>	Occurs when the started Timer<index> expires. **Hint:** Only the armed timers raise events. Example: Sys.Timer.e1 How to use such an event, refer to chapter *11.6.2.3.*
	<index>	Specifies the timer identifier. It can be set to a value from 0 to 39 (20-39 [1]).

6.1.12. Sys.Trigger

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓
1	✓	✓	✓	✓
2	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	Sys.Trigger.s<index>=high	True when the Trigger <index> is High until it changes to Low. Trigger states can be set/changed using the corresponding PFAL command. Example: Sys.Trigger.s1=high
	Sys.Trigger.s<index>=low	True when the Trigger <index> is Low until it changes to High. Trigger states can be set/changed using the corresponding PFAL command. Example: Sys.Trigger.s1=low

STATES		
	State Notification Code	**Meaning**
	<index>	Specifies the trigger index. It can be set to a value from **0** to **39 (20-39[2])**. How to use such events, refer to chapter *11.6.2.4.* (advanced example related to a used trigger).

EVENTS - are evaluated just when the event occurs		
	Event Notification Code	**Meaning**
	Sys.Trigger.e<index>[=<level>] [1]	Occurs when setting a Trigger <index> to High. The state of trigger can be changed with the corresponding PFAL command. Example: Sys.Trigger.e1=high Sys.Trigger.e1=low Sys.Trigger.e1
	Sys.Trigger.e<index>=low [1]	Occurs when setting a Trigger <index> to Low. The state of trigger can be changed with the corresponding PFAL command. Example: Sys.Trigger.s1=low
	<index>	Specifies the trigger index. It can be set to a value from 0 to **39 (20-39 [2]).** How to use such events, refer to chapter *11.6.2.4.* (advanced example related to a used trigger).
	<level>	Optional. It specifies the level of the trigger index. It can be set to: **High** Occurs when the trigger index changes to high **Low** Occurs when the trigger index changes to low

6.1.13. Sys.Counter

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓
1	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	Sys.Counter.s<index><comp><value>	True when the value of the Counter<index> matches the specified value. Counters do not automatically increment or decrement themselves. To reach a certain value for a counter use the increment or decrement command (see chapter *4.2.12.1.*) which should be x-time called until the counter reaches the supposed value. When system starts all counters are initialized to 0. Example: Sys.Counter.s0>30 How to use such a state see chapter *11.6.2.5.*
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <= or>=.
	<value>	Specifies the value to be compared. 32-bit integer value from 0 to 2147483647.

EVENTS - are evaluated just when the event occurs		
	Event Notification Code	**Meaning**
	Sys.Counter.e<index>	Occurs when a configured system **Counter** reaches its minimum value zero (0) for the first time. To reach a 0 value for a counter, the decrement command should be x- time called until the counter reaches that value. For example: Decrements a counter (counts toward the minimum 0) when a SMS is sent. When the specified counter reaches the value 0, then send a TCP packet to the connected remote server. $PFAL,Sys.Counter0.set=5;Sys.Counter0.save1 $PFAL,Cnf.Set,AL0=Sys.Device.eStart:Sys.Counter0=load1 $PFAL,Cnf.Set,AL1=GSM.SMS.eSent:Sys.Counter0.Decrement=1 $PFAL,Cnf.Set,AL2=Sys.Counter.e0:TCP.Client.Send,8,"positions:" When device delivers 5 SMSs, it transmits also a TCP packet to the server.
	<index>	Enables you to specify the counter identifier. It can be set to a value from **0** to **39** (**20-39** [1]).

6.1.14. Sys.nvCounter

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	Sys.nvCounter.s<index><comp><value>	True when the value of the nvCounter<index> matches the specified value. Counters do not automatically increment or decrement themselves. To reach a certain value for a counter use the increment or decrement command which should be x-time called until the counter reaches the supposed value. When system starts all counters are initialized to 0. Example: Sys.nvCounter.s0>30
	<index>	Specifies the counter identifier. It can be set to a value from 0 to 19
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <= or>=.
	<value>	Specifies the value to be compared. 32-bit integer value from 0 to 2147483647.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	Sys.nvCounter.e<index>	Occurs when a configured **nvCounter** reaches its minimum value zero (0) for the first time. To reach a 0 value for a counter, the decrement command should be x- time called until the counter reaches that value. **For example:** Decrements a nvcounter (counts toward the minimum 0) when a SMS is sent. When the specified counter reaches the value 0, then send a TCP packet to the connected remote server. $PFAL,Sys.nvCounter0.set=5;Sys.nvCounter0.save1 $PFAL,Cnf.Set,AL0=Sys.Device.eStart:Sys.nvCounter0=load1 $PFAL,Cnf.Set,AL1=GSM.SMS.eSent:Sys.nvCounter0.Decrement=1 $PFAL,Cnf.Set,AL2=Sys.nvCounter.e0:TCP.Client.Send,8," positionns:" When device delivers 5 SMSs, it transmits also a TCP packet to the server.
	<index>	Specifies the counter identifier. It can be set to a value from 0 to 19

6.1.15. Sys.Power

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	Event Notification Code	**Meaning**
	Sys.Power.sVoltage<comp><value>	Checks the current voltage of a connected external power supply.
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <= or>=.
	<value>	It is a signed decimal value (in volts). Its range is nothing to care about – basically any voltage can be specified. Also a fractional value can be specified ranging from **0** to **.999**. **Examples**:*-5.1= -5.001 V* 12= 12.000 V 1.123 V

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	Sys.Power.eDetected	This event is generated when an external power supply is connected (*external voltage is higher than battery voltage*).
	Sys.Power.eDropped	This event is generated when an external power supply is disconnected (*external voltage drops below battery voltage*).

6.1.16. Sys.Bat

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	?	?	?	✓

STATES		
	State Notification Code	**Meaning**
	Sys.Bat.sVoltage<comp><value>	Checks the currently available battery voltage. Example: Sys.Bat.sVoltage<3.8
	Sys.Bat.sCharge=[<state>]	Checks the current state of the internal battery.
	Sys.Bat.sMode=[<function>]	The condition is true when a battery is being charged.
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <= or>=.
	<value>	It is a signed decimal value (in volts). Its range is nothing to care about – basically any voltage can be specified. Also a fractional value can be specified ranging from **0** to **999.** **Examples**:-5.1= -5.001 V 12= 12.000 V 1.123 V
	[<state>]	**start** Battery is currently charged. **stop**,[<**stop_reason**>] Charging was stopped, a **<stop_reason>** can be specified.. <stop**_reason**> The following are listed the optional setting for stop charging: **full** Battery is fully charged **disabled** Battery charger has been disabled (e.g. by PFAL command or an alarm action) **no_power** External power has dropped, so charging isn't possible anymore **temperature** Charging temperature exceeds range
	[<function>]	**Auto** When the device starts with or without external power. **Disabled** When internal battery is not used for normal operations. **Always** When device uses the internal battery as power source.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	Sys.Bat.eLow	This event is generated only if battery mode is set to auto or always and NO external power is detected. If battery voltage drops below a defined minimum, the event will be generated. If the battery voltage drops further and reaches a critical level an emergency sleep is entered automatically (wakeup reason: external power). The event eShutdown is created in this case. Usually there is enough energy to send a few SMS and/or send some TCP packets if this emergency sleep occurs. **Note**: It can be used to send e.g. a notification to call center or inform the user of a low battery in order to prevent an emergency sleep. The device will exit its emergency sleep mode if external power is detected.
	Sys.Bat.eCharge=[<state>]	This event is generated when a battery changes its charge state. **Note** that the battery charger will attempt to keep a battery fully charged. This means the charge state might change sometimes from full to start and vice versa if the battery is almost fully charged.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	[<state>]	**start** Charger has been started **stop**,[<>]Charger has been stopped, a <stop_reason> can be specified. ***stop_reason***> The following are listed the optional setting for stop charging: **full** *Battery is fully charged.* **disabled** *Battery charger has been disabled (e.g. by PFAL command or an alarm action).* **no_power** *External power has dropped, so charging isn't possible anymore.* **temperature** *Charging temperature exceeds range.*

6.1.17. Sys.Info

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	Sys.Info.e<index>	This event is occurred by the device to report a list of specific system events. It can be used to perform alarm actions like sending a user defined error/ success message etc.
	<index>	Specifies a decimal value from 0...n. The following values are currently supported by the device: **0** Backup configuration successfully restored (caused by user command OR automatically by the device) **1** Backup configuration itself is corrupted and is automatically deleted by the device. ***Warning:*** ***Special care should be taken when using system info events. It is strongly recommended to try out possible side effects before configuring alarms which affect the reason of the event.You may use System info events at your own risk as they might produce unexpected system behavior if combined with wrong alarm actions.***

6.1.18. Sys.eTimeSync

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓
1	✓	✓	✓	✗

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	Sys.eTimeSync=<event>	This event is generated when the system time is synchronized either the command <Sys.SetTime> (see chapter *4.2.6.1.*) executed by user, the last valid GPS position and time is synchronized after startup (RTC) or after a request for a valid GPS fix time.
	<event>	The event can be occurred with one of the following settings: **USER** - User executed the command **Sys.SetTime**. **RTC** [1] - The last valid GPS position and time after startup (RTC) **GPS** - when the device requests for a valid GPS fix time.

6.1.19. Sys.1Wire

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓
1	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	SYS.1Wire.sAvailable=<"value">	True as long as a 1-Wire device is connected to the bus on the FOX3-2G/3G/4G or BOLERO40 device. Example: Sys.1Wire.sAvailable="21c2272e000000EF"
	SYS.1Wire.sAvailable=whitelist[1]	True as long as the Id of the connected 1-Wire device to the bus on the FOX3-2G/3G/4G or BOLERO40 device is available on the whitelist.
	<value>	Specify an unique ID of this device in hexadecimal notation (8 bytes). The parameter is optional and can be used to detect specific devices or iButton.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	SYS.1Wire.eRegister=<"value">	This event is generated when a 1-Wire device is connected to the 1-Wire interface on the FOX3. Example: Sys.1Wire.eRegister="21c2272e000000EF"
	SYS.1Wire.eRegister<comp>WhiteList	This event is generated whenever the ID of the connected 1-Wire device is available in the Whitelist set with PFAL command Sys.Whitelist.Set - see chapter *4.2.23.4.* Example: Sys.1Wire.eRegister=WhiteList

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	SYS.1Wire.eRelease=<"Value">	This event is generated when a 1-Wire device is disconnected from the 1-Wire interface on the FOX3. In case several 1-Wire devices are disconnected immediately, only the event of the last removed ID device is generated (→ not suited for this case, use dynamic variable „removed 1-wire-list“ instead) Example: Sys.1Wire.eRelease="21c2272e000000EF"
	SYS.1Wire.eRelease<comp>WhiteList	This event is generated whenever the ID of the disconnected 1-Wire device is available in the Whitelist set with PFAL command Sys.Whitelist.Set. Example: Sys.1Wire.eRelease=WhiteList see chapter *4.2.23.4.*
	<"Value">	Specifies an unique ID of this device in hexadecimal notation (8 bytes). The parameter is optional and can be used to detect specific 1-Wire devices or iButton connected to the FOX3.
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=

6.1.20. Sys.BLE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	×	×	×	×
1	×	×	×	×

Exception: This command is supported only for FOX3-3G BLE variant. All related BLE commands, events and configuration can only be used by a BLE variant device (FOX3-3G BLE, FOX3-4G BLE).

STATES		
	State Notification Code	**Meaning**
	SYS.BLE.sConnected	True as long as the BLE connection state of the FOX3-3G-BLE (slave) to a BLE master (Smartphone/Tablet/PC) has changed from disconnected to connected.
	SYS.BLE.sDisconnected	True as long as the BLE connection state of the FOX3-3G-BLE (slave) to a BLE master (Smartphone/Tablet/PC) has changed from connected to disconnected. It may happen after the master device has disconnected an established connection, the connection of FOX3-3G-BLE (slave) to a master is lost.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	SYS.BLE.eConnected	Occurs when the connection to a host is established.
	SYS.BLE.eDisconnected	Occurs when the connection to a host is lost.
	SYS.BLE.eRegister=<"id_name">	Occurs when a BLE beacon is inside the range of the FOX3-3G-BLE. This event can also be used when the incoming ID, name or MAG matches one of the entries available in the Whitelist. Entries in the Whitelist can be set with $PFAL,Sys.Whitelist.Set. Entries in this whitelist are BLE.WHITELIST configuration setting dependent. E.g.: if BLE.WHITELIST=MAC then $PFAL,Sys.Whitelist.Set. should contain only MAC addresses of the BLE sensors (e.g.: E2:12:34:4D:5F:4A)- see chapter *4.2.19.*

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	SYS.BLE.eRegister=whitelist[1]	Occurs when the incoming ID, name or MAG matches one of the entries available in the Whitelist. Entries in the Whitelist can be set with *$PFAL,Sys.Whitelist.Set.* Entries in this whitelist are BLE.WHITELIST configuration setting dependent. E.g.: if BLE.WHITELIST=MAC then $PFAL,Sys.Whitelist.Set. should contain only MAC addresses of the BLE sensors (e.g.: E2:12:34:4D:5F:4A)- see chapter *4.2.19.*
	SYS.BLE.eRelease=<"id_name">	Occurs when an already registered BLE beacon is out of range of the FOX3-3G-BLE.
	SYS.eBleData="text"	Occurs when the FOX3-3G-BLE receives specific data from the connected host. Example: SYS.eBleData="test"
	["id_name"]	Defines either the ID or name of the BLE beacon.
	["text"]	It specifies in quotation marks (" ") the text (i.e. "your text") to be compared. This comparison is case sensitive. The incoming text must be terminated with <CR><LF>.

6.1.21. Sys.NFC

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	?	?	?	?

STATES		
	State Notification Code	**Meaning**
	SYS.NFC.sCARD=<"id">	Checks whether or not the scanned card has the same ID with specified one.
	SYS.NFC.sCARD<comp>whitelist	Checks whether or not the ID of the scanned card is already available in the whitelist.
	["id"]	Defines ID of the NFC card .
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=,

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	SYS.NFC.eCARD=<"id">	Occurs when ID of the scanned card is the same with the specified one.
	SYS.NFC.eCARD<comp>whitelist	Occurs when ID of the scanned card is available in the whitelist (see)
	SYS.NFC.eLOST[<comp>whitelist]	Occurs when NFC has lost connection to an NFC card or to one of the NFC card' IDs stored in the whitelist (see *4.2.21.*)
	["id"]	Defines ID of the NFC card .
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=,
	SYS.NFC.eStart	Occurs when a connected NFC reader starts to operate.

6.1.22. Sys.WLAN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	?	?	?	×

STATES		
	State Notification Code	Meaning
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	Meaning
	SYS.WLAN.TCP.ePingSent	Occurs when a ping is sent to connected TCP server over the WLAN access point.

6.1.23. Sys.eDTCO.DRIVER.STATE

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	?	?	?	×

STATES		
	State Notification Code	Meaning
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	Meaning
	SYS.eDTCO.DRIVER.STATE	Occurs when the Drivers Id or state has changed

6.1.24. Sys.eUserText

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	?	?	?	×

STATES		
	State Notification Code	Meaning
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	Meaning
	SYS.eUserText[=<"text">]	This events occurs when the device receives any or optional a specific text with command $PFAL,MSG.Event,User,<"text">

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	<"text">	Optional. Specifies the text to be compared with the incoming message used in command $PFAL,MSG.Event,User,<"text">. The comparison text can be a number or any sequence of characters. The comparison is case-sensitive. The event occurs if the specified event text matches exactly with the incoming message specified with $PFAL,MSG.Event,User,<"text">.

6.1.25. Sys.Iobox

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	?	?	?	×

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	SYS.Iobox.eReset[=<cond>[...<cond>]]]	This events occurs when the IOBOX-CAN/MINI or IOBOX-WLAN connected to the device has performed a software reset.
	SYS.IoBox.eLost	This event is occurred when IOBOX-MINI/CAN/WLAN is disconnected or the connection to that box is lost.
	<cond>	Defines the reason of device reset. Separated by commas “,” specify one or a set of reasons: **PINRST** - 0x04 PIN reset **PORRST** - 0x08 POR/PDR reset **SFTRST** -0x10 Software reset **IWDGRST** - 0x20 Independent watchdog reset **WWDGRST** - 0x40 Window watchdog reset **LPWRRST** - 0x80 Low-power reset

6.1.26. Sys.eco.PDO

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	?	?	?	×

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	SYS.Eco.PDO	This event occurs when a PDO has been received.

6.2. BLUEID

6.2.1. BLUEID.e/s

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	×	×	×	×

Note: *Exception: This command is supported only in FOX3-3G BID Variant.*

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	BLUEID.eDATA	Occurs when the device detects incoming data sent from the mobile device app
	BLUEID.eCMD=<"Command">	Occurs when the device detects incoming data is specific command or in case of using a nfc card “nfc1” from successful decrypted and as valid approved BlueID ticket. Example: BLUEID.eCMD=<"nfc1">
	BLUEID.eTICKETS	Occurs when the device detects incoming BlueID ticket data as a binary large object while an over-the-air ticket synchronization process.
	< Command>	String type.

6.3. IO

6.3.1. IO.e/s

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	IO.s<index>=<DigLevel>	Checks if the chosen input is at the specified level. Example: IO.s1=high or IO.s1=low
	IO.s<index><comp><voltage>	Checks whether the current voltage of the analog input is at/within/out of the range user specified level(s). Example: IO.s3<5.5 or IO.s2==10.0:12.5 or O.s2!=10.0:12.5
	<index>	See chapter *4.4.1.* for a list of all available IO indexes in different AVL and accessory devices such as IOBOX-MINI and IOBOX-CAN.
	<DigLevel>	Defines when the state should be true: **high** As long as the level is high **low** As long as the level is low
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <=, >= or ==.

STATES		
	State Notification Code	**Meaning**
	<voltage>	Specifies the number of volts between 0 and 31 for analog inputs. This number may have an accuracy of 3 digits which are separated by dot "." *(i.e. 1.24 or 28.459)*. It allows to specify also a voltage range separated by semicolon ":" (e.g. 10.0:12.5).

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	IO.e<index>=<event>	This event is generated when an input changes its state. Example: IO.e1=redge or IO.e1=fedge or IO.e1=edges How to use such a state, see 1.
	<index>	See chapter 1, for a list of all available IO indexes for different AVL and accessory devices such as FOX3-2G/3G/4G, BOLERO40, IOBOX-MINI, IOBOX-CAN and IOBOX-WLAN.
	<event>	Defines when the event should be occurred: **redge** when level changes from low to high **fedge** when level changes from high to low **edges** when level changes (H> L or L->H) Note:This event can be used to detect when an input changes (i.e. a button is pressed / released etc..)

6.3.2. IO.Motion

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	IO.Motion.sMoving	True when the device is moving with changeable velocity.
	IO.Motion.sStanding	True when the device is moving at a constant velocity.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	IO.Motion.eMoving	Occurs when the device starts moving (changes velocity).
	IO.Motion.eStanding	Occurs when the device stops moving (constant velocity)
	IO.Motion.eForce	This event occurs when the configured force acceleration is exceeded (see configuration parameter in chapter *5.4.3.* for more details)
	IO.Motion.e3DForce=<direction>	This event is generated when the Device exceeds the configured force acceleration in one direction (see configuration in chapter *5.4.4.* for more details)

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	<direction>	The event is generated only on the specified axe. It can be set to: x+ The pre-defined force threshold exceeded in positive x direction x- The pre-defined force threshold exceeded in negative x direction y+ The pre-defined force threshold exceeded in positive y direction y- The pre-defined force threshold exceeded in negative y direction z+ The pre-defined force threshold exceeded in positive z direction z- The pre-defined force threshold exceeded in negative z direction When needing more events for different axis, use either different alarms or use the logic operator "?" which represents the "OR" conjunction in alarms. E.g.$PFAL,Cnf.Set,AL91=IO.Motion.e3DForce=x+?IO.Motion.e3DForce=x-?IO.Motion.e3DForce=Y+:TCP.Client.Send,8,"3DForce occurred"
	IO.Bearing.e<index>	Occurs when the specified values of BEARING<index> are exceeded. This event occurs when the MOTION.BEARING parameter are reached.
	<index>	Specify the index of a pre-configured bearing which ranges from 0 to 4.

6.3.3. IO.PulseCnt

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	IO.PulseCnt.s<id><comp><value>	It checks if the specified pulse counter fulfils the specified condition. Note: This state can be used to check the current pulse counter with an additional condition (i.e. launch an alarm if the pulse counter exceeds a specific value)
	<id>	Pulse counter slot ID – dedicated to the specific IO using pulse counter functionality.
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <= or>=.
	<value>	A counter value number between 0 and 65534 (2^{16}-1)

6.4. GPS

6.4.1. GPS.eJamming

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GPS.eJamming[=<flag>]	The FOX3 series has a GPS jamming detection event that can be generated when GPS jamming is detected. This feature detects, at radio resource level, an anomalous source of interference and signalling. It occurs when the GPS module detects abnormalities/interferences during operation, e.g. interference, noise, jamming, after re-acquisition, wake-up. Once it occurs, the device can be configured to send information to the server or to a phone number.
	[<flag>]	Optional. Following indicator flags can be set to generate this event: **OK** If no interference detected. The jamming level is less than the specified minimum level on the parameter in chapter *5.1.26.* **Warning** The GPS position ok, but interference detected. The jamming level is greater than the specified maximum level on the parameter in chapter *5.1.26.* and the state of GPS fix is 2D. **Critical** No GPS position fix, interference present. The jamming level is greater than the specified maximum level on the parameter in chapter *5.1.26.* and the state of GPS fix is 3D.

6.4.2. GPS.Nav

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	GPS.Nav.sFix=valid	True when the device obtains a valid GPS-fix (GPS data goes from invalid to valid) until it loses that fix. This state is configuration-dependent.
	GPS.Nav.sFix=invalid	True when the device loses a valid GPS-fix *(GPS data goes from valid to invalid)* until the GPS-fix available again. State is checked very second.
	GPS.Nav.sFix=correct	True when the current GPS position is valid and passed **GPS.AUTOCORRECT** conditions. It can be true only if **GPS.AUTOCORRECT** feature is used. State is checked very second.

STATES		
	State Notification Code	**Meaning**
	GPS.Nav.sDeltaSpeed<comp><value>	This state can be used to check the speed de/increase per second of the device. Deltaspeed can check speed differences up to an accuracy of 1cm/s. True when the delta speed matches the user-specified value. The user-set comparator determines when this state should be true. Only integer values may be entered. It is also possible to enter negative value (which means decrease). Example: GPS.Nav.sDeltaSpeed>100 // increase of more than 100cm/s GPS.Nav.sDeltaSpeed<-100 //decrease of less than 100cm/s
	GPS.Nav.sDist<comp><value>	This state can be used to check navdist counter within alarms. True when the driven distance (controlled by GPS.Nav.Distance) matches the user-specified value. The user-set comparator determines when this state should be true. Only integer values may be entered. It is possible but strongly not recommended, to enter negative distances.
	GPS.Nav.sDist2<comp><value>	This state can be used to check navdist2 counter within alarms. True when the driven distance (controlled by GPS.Nav.DeltaDistance2) matches the user-specified value. The user-set comparator determines when this state should be true. Only integer values may be entered. It is possible but strongly not recommended, to enter negative distances.
	GPS.Nav.sSpeed<comp><value>	True when the current speed of the vehicle matches the user-specified value. The user-set comparator determines when this state should be true. <value> Sets the value of speed, in metres per second, between 0 and 2147483647 that matches your application. Example: GPS.Nav.sSpeed>40
	GPS.Nav.Position.s<buffer_index><comp><value>	True when the current distance of the device from a stored position <buffer_index> matches the user-defined value. The user-set comparator determines when this state should be true. **Hint:** This state can be used to monitor the distance of the device from a stored position. In combination with a counter it is possible to calculate the distance of the device on which it has been moved since e.g. AVL device is turned on. (= the driven kilometres without breaks). **Example:** GPS.Nav.Position.s1>120 (See chapter *4.5.1.1.* for more details)
	<buffer_index>	It is a number, which determines how far from a stored position the AVL device is. The system gets the contents of the defined buffer index and calculates the distance the AVL device has moved from that point. It can be set to a value from 0 to 4. A GPS position can be temporarily stored using the following command: *GPS.Nav.Position<buffer_iindex>*=<type> A GPS position can be permanently stored using the following command, if the content of <buffer_index> is not empty. *GPS.Nav.Position<buffer_index>*=save<slot_id> Whenever system starts up the contents of the <slot_id> has to be loaded into the <buffer_index> for further use. GPS.Nav.Position<buffer_index>=load<slot_id>
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <= or>=.
	<value>	Set the value of the distance, in meter, between *0* and *2147483647* that matches your application.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GPS.Nav.eFix=valid	Occurs when four or more GPS satellites are used. By default, the Tmark-GPIO (GPIO9) is configured to indicate the GPS fix validity. How to use such an event, refer to chapters 1
	GPS.Nav.eFix=invalid	Occurs when three or less GPS satellites are used. (GPS position goes from valid to invalid). How to use such an event, refer to chapter 1
	GPS.Nav.eChangeHeading	This event is generated whenever the device changes its heading for more than the specified heading tolerance. This event can be used to launch actions whenever the device deviates from the predefined degree - which allows to write history positions efficiently. See PFAL command **GPS.Nav.SetHeadingTolerance** for more details.
	GPS.Nav.eChangeHeading2	This event is generated whenever the device changes its heading for more than the specified Heading2 tolerance. This event can be used to launch actions whenever the device deviates from the predefined degree - which allows to write history positions efficiently. See PFAL command **GPS.Nav.SetHeading2Tolerance** for more details.
	GPS.Nav.eSpeed<comp><value>	Occurs when the When the predefined speed threshold is exceeded. Example: $PFAL,Cnf.Set,AL1=GPS.Nav.eSpeed>25:TCP.Client.Send,8,"Speed exceeded 90km/h &(Speed.kmh)"
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <= or>=.
	<value>	Set the value of the speed, in meter/second, between *0* and *2147483647* that matches your application.
	GPS.Nav.eCellLocate	Get celllocate data (latitude/longitude) from µBlox server Note: The Celllocate works only when no GNSS is available.

6.4.3. GPS.Time

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	GPS.Time.sYear<comp><value>	True as long as the current year obtained from the GPS date matches the specified year <value> (format is "**yyyy**", e.g. 2006). <value> Set the value of year in a range of 1 and 10000 that matches your application. Example: GPS.Time.sYear=2006
	GPS.Time.sMonth<comp><value>	True as long as the current month obtained from the GPS date matches the specified month <value> (the format is "**mm**", e.g. 12). <value> Integer value between 1 and 12, where January is the first month of the year and December is the twelfth. Example: GPS.Time.sMonth=12

STATES		
	State Notification Code	**Meaning**
	GPS.Time.sMDay<comp><value>	True as long as the current day of month obtained from the GPS date matches the specified day <value> (the format is "**dd**" without leading "0" , e.g. 8). <value> Integer value between 1 and 31. Valid Day values are 1 through 28, 29, 30, or 31, depending on the current Month value. For example, the possible Day values for month 2 (February) are 1 through 28 or 1 through 29, depending on whether or not the Year value specifies a leap year. If the specified value is not within range, this state never results True. Example: GPS.Time.sMDay=8
	GPS.Time.sWeek<comp><value>	True as long as the current number of weeks obtained from the GPS date matches the specified number of weeks <value> (the format is "**ww**" without a leading "0", e.g. 40). <value> Integer value between 1 and 52. Example: GPS.Time.sWeek=40
	GPS.Time.sWDay<comp><value>	True as long as the current week day obtained from the GPS date matches the specified week day <value> (the format is "**d**", e.g. 2). <value> Integer value between 1 and 7, where Monday is the first day of the week and Sunday is the seventh. Example: GPS.Time.sWDay=2
	GPS.Time.sHour<comp><value>	True as long as the current hour obtained from the GPS date matches the specified hour <value> (the format is "**hh**" without a leading "0", e.g. 40). <value> Integer value between 0 and 23. If the specified value is not within range, this state never results True. The value 0 corresponds to midnight, 12 corresponds to noon, and so on. Example: GPS.Time.sHour=12
	GPS.Time.sTimespan=<value>	True as long as the current system time is within the user specified time span <value>. **Note**: This state can be used to generate activity reports or stop reports for a period of time (e.g. generate activity reports between 8:30:00 and 14:45:00). <value> Represents a period of time as separate start time and end time values separated by dash "-" without spaces (TimeStamp as a string in the format **hh:mm:ss-hh:mm:ss**). Valid Hour (**hh**) values are 0 through 23, without a leading zero. Valid Minute (**mm**) and Second (**ss**) values are 0 through 59, without a leading zero. Example: GPS.Time.sTimespan=8:30:40-14:45:51
	GPS.Time.sDatespan=<value>	True as long as the current GPS date is within the user specified date span <value>. **Note**: This state can be used to generate activity reports or stop reports for a range of date (e.g. generate activity reports from Tuesday 31th January 2006 until Thursday 16th February 2006). <value>Represents a range of date as separate start date and end date values delimited by dash "-" (DateStamp as a string in the format dd.mm.yyyy-dd.mm.yyyy). The start and end dates consist of the number of day, month and year. depending on the specified Month value (mm), valid Day (dd) values are 1 through 28, 29, 30, or 31, without a leading zero. For example, the possible Day values for month 2 (February) are 1 through 28 or 1 through 29, depending on whether or not the Year value specifies a leap year. Valid Month (mm) values are 1 through 12, without a leading zero and Year (yyyy) values are 2000 through 9999. Example: GPS.Time.sDatespan=10.01.2006-20.01.2006

STATES		
	State Notification Code	**Meaning**
	GPS.Time.sMinute<comp><value>	True as long as the specified minute value matches the minute range in the system time. <value> The minute. The valid values for this setting are 0 through 59. Example: GPS.Time.sMinute>50 This example shows that this state is true each hour from 51 to 59 minute.
	GPS.Time.sSecond<comp><value>	True as long as the specified second value matches the second range in the system time. <value> The second. The valid values for this setting are 0 through 59. Example: GPS.Time.sSecond>20 This example shows that this state is true each minute from 21 to 59 second.
	GPS.Time.sPeriod=<time_period>	True as long as the system time is within the specified time periodes. Example: $PFAL,CNF.Set,AL1=Sys.eSerialData0="0049123456789"&GPS.Time.sPeriod=01.01.2015,12:00-31.01.2015,16:00:TCP.Client.Send,8,"&(SerialData0)"
	<time_period>	Specifies the time period in the following formats: **Format Description** --- **dd.mm.yyyy,hh:mm[:sec]-dd.mm.yyyy,hh:mm[:ss]** It specifies the date and time periodes in which this state will be set to true. **[:ss]** - Optional **dd.mm.yyyy,hh:mm[:ss]-hh:mm[:ss]**It specifies the date and the time frame in which this state will be set to true. **[:ss]** - Optional Where: **dd** = day; **mm** = month; **yyyy** = year; **hh**= hour (0-23); **mm** = minutes(0-59); **[ss]** = seconds (0-59). You can setup a time period and combine it with other events to execute some actions within this time frame. The time is taken from the GPS or the RTC. Suppose you want to transfer a read RFID to the server in a pre-defined time frame 01.01.2015,12:00-31.01.2015,16:00 StartDate:01.01.2015 StartTime:12:00 EndDate: 31.01.2015 EndTime: 16:00 $PFAL,CNF.Set,AL1=Sys.eSerialData0="0049123456789"&GPS.Time.sPeriod=01.01.2015,12:00-31.01.2015,16:00:TCP.Client.Send,8,"&(SerialData0)" It means, the AVL sends to the server the ID of the TAG only in the pre-defined time frame. Please note that the GPS time is being checked once/second.
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <= or>=.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	Meaning
	GPS.Time.eYear=<value>	Occurs when the specified year value corresponds to the year in the system time. <value>: Specifies the year as a numeric value representing the year.
	GPS.Time.eMonth=<value>	Occurs when the specified month value corresponds to the month in the system time. It occurs one time in year in the specified month. <value>: Specifies the month as a numeric value from 1 to 12 representing the month.
	GPS.Time.eMDay=<value>	Occurs when the specified value of day of month corresponds to the day of month in the system time. It occurs once per month on the specified month day. <value>: Specifies the day of month as a numeric value from 1 to max. 31 representing the month day.
	GPS.Time.eWeek=<value>	Occurs when the specified value of week corresponds to the week of calculated in the system date. It occurs one time in year in the specified week. <value>: Specifies the week of year as a numeric value from 1 to 52 representing the week.
	GPS.Time.eWkDay=<value>	Occurs when the specified value of day of week corresponds to the day of week calculated from the system date. It occurs one time in week in the specified week. <value>: Specifies the day of month as a numeric value from 1 to 7 representing the day of the week (1= Monday; 7=Sunday).
	GPS.Time.eHour=<value>	Occurs when the specified hour value corresponds to the hour in the system time. It occurs one time in 24 hours. <value>: Specifies the hour as a numeric value from 0 to 23 representing the hour of day.
	GPS.Time.eMinute=<value>	Occurs when the specified minute value corresponds to the minute in the system time. It occurs one time per hour. <value>: Specifies the minute as a numeric value from 0 to 59 representing the minute.
	GPS.Time.eSecond=<value>	Occurs when the specified second value corresponds to the second in the system time. It occurs one time per minute. <value>: Specifies the second as a numeric value from 0 to 59 representing the second.

6.4.4. GPS.History

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	GPS.History.sDist<comp><value>	True when the current distance of the device from the position of the last stored history entry matches the specified value <value>. The set comparator determines how the comparison of both values (device's location and user's value) will take place. Example: GPS.History.sDist>1000 How to use such a state, refer to chapter *4.5.2.*
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <= or>=.
	<value>	Sets the value of distance, in meter, between **0** and **2147483647** that matches your application.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GPS.History.eTaut	Occurs when the memory used for history data gets low (approx. 93% of this memory has been used up). As mentioned in chapter *4.5.2.* , the history operates on a first-in-first-out (FIFO) basis, limiting the amount of memory space used by history. Use this event to prevent overwriting of available data in the history. You may configure an alarm that notifies the user to download the history data on the server.
	GPS.History.ePushFinished	Occurs when the history push command completes. It can be used to clear the history after reading it out via ***History.Push*.** Note that history events which are written during the readout are erased too. **Note:** This event will be generated in any case – even if push mode fails (i.e. no **setread** has been performed before)

6.4.5. GPS.Geofence

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	GPS.Geofence.s<id>=[<state_type>]	True when the device leaves the geofence<id>. Example: GPS.Geofence.s1=inside GPS.Geofence.s1=outside
	<id>	Used to identify the configured areas and to split states from different Geofences Following are listed all supported geofence IDs. **0** - checks the state of parking area. **1**..**99** - checks the state of a specific user-configured area. **100-2999** - checks the state of a specific user-configured area (Premium-Feature) Please, specify an geofence <id> that is already configured. How to configure a Geofence <id> refer to chapter *4.5.3.*

STATES		
	State Notification Code	**Meaning**
	[<state_type>]	Defines the type of an geofence state. **inside** - True when the device enters into the pre-configured Geofence<id>. Once it results outside the geofence <id> this state goes false, and the state "G*PS.Geofence.s*<id>=*outside*" goes true. **outside** - True when the device leaves the geofence <id>.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GPS.Geofence.e<id>=[<event_type>]	Occurs when the device enters into and/or leaves the geofence <id>. Example: GPS.Geofence.e1=inside GPS.Geofence.e1=outside GPS.Geofence.eX=inside GPS.Geofence.eX How to use such an event refer to chapters *4.5.3.*
	<id>	Used to identify the configured areas and to split events from different Geofences. Following are listed all supported geofence IDs. **0** - used for parking area. **1** .. **99** - used for user-configured geofences. **100-2999** - checks the state of a specific user-configured area (Premium-Feature) **x** - used for any user-configured geofences. Please, specify a geofence <id> that is already configured, otherwise no event occurs. How to configure a Geofence <id> refer to chapter *4.5.3.*
	[<event_type>]	It is **optional**. Defines the type of an area event. **inside** - Occurs only when the device enters into a geofence <id> **outside** - Occurs only when the device leaves a geofence <id> If it is omitted, event occurs when the device enters into and leaves a geofence <id>

6.4.6. GPS.Area

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	GPS.Area.s<id>=[<state_type>]	True when the e device leaves the area <id> (created from one or several Geofences). True when the device is inside the defined area <id> (created from one or several Geofences). Example: GPS.Area.s1=inside GPS.Area.s1=outside

STATES		
	State Notification Code	**Meaning**
	<id>	Used to identify the configured areas and to split states from different Areas. Following are listed all supported area IDs. **0** - Checks the state of parking area **1** .. **31** - Checks the state of a specific user-configured area Please, specify an area <id> that is already configured. How to configure an area <id> refer to chapter 1.
	[<state_type>]	**Optional**. Defines the type of an area state, when the state will be set to true. **inside** - True when the device enters into the area <id>. Once it results outside area <id>, this state goes false, while the state "*GPS.Area.s<id>=outside*" goes true. **outside** - True when the device leaves the area <id>.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GPS.Area.e<id>=[<event_type>]	Occurs when the device enters into and/or leaves an Area <id>. Example: GPS.Area.e1=inside GPS.Area.e1=outside GPS.Area.eX=inside GPS.Area.eX How to use such an event refer to chapter *11.8*.
	<id>	Used to identify the configured areas and to split events from different Areas. Following are listed all supported area IDs. **0** - used for parking area **1** .. **31** - used for user-configured areas. **x** - used for any user-configured areas Please, specify an area <id> that is already configured, otherwise no event occurs. How to configure an area <id> refer to chapter 1.
	[<event_type>]	It is **optional**. Defines the type of an area event. **inside** - Occurs only when the device enters into an Area <id> **outside** - Occurs only when the device leaves an Area <id> If it is omitted, event occurs when the device enters into and leaves an Area <id>

6.4.7. GPS.WPGF

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	GPS.WPGF.s[<state_type>	True when the device leaves the corridor of the waypoints. Example: GPS.WPGF.sInside GPS.Area.sOutside
	[<state_type>]	Defines the type of the waypoints corridor state. **inside** - True when the device enters into the corridor of the waypoints. Once it results outside this corridor this state goes false, and the state "GPS.WPGF.sOutside" goes true. **outside** - True when the device leaves the corridor of the waypoints.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GPS.WPGF.e[<event_type>]	Occurs when the device enters into or leaves the corridor of the waypoints. Example: GPS.WPGF.eInside GPS.WPGF.eOutside
	[<event_type>]	It is **optional**. Defines the type of the corridor of the waypoints event. **inside** - Occurs only when the device enters into the corridor of the waypoints. **outside** - Occurs only when the device leaves the corridor of the waypoints.

6.4.8. GPS.eExtAnt(Un)Plugged

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	×

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GPS.eExtAntPlugged	The firmware running in the AVL devices can detect when an external GPS antenna is connected to the device. This event occurs when connecting (plugging in) an external GPS antenna to the device. Example: $PFAL,CNF.Set,AL30=GPS.eExtAntPlugged:GSM.SetExternalAntenna This alarm switches the device to the external antennas when the external antenna is plugged.
	GPS.eExtAntUnplugged	The firmware running in the AVL devices can detect when an external GPS antenna is disconnected from the device. This event occurs when disconnecting (unplugging) an external GPS antenna from the device. Example: $PFAL,CNF.Set,AL31=GPS.eExtAntUnplugged:GSM.SetInternalAntenna This alarm switches the device to the internal antennas when the external antenna is unplugged.
Both events above can be used in separate alarms to automatically switch between the internal and the optional external antennas.		

6.4.9. GPS.MultiGeofence

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	🗝	🗝	🗝	🗝

STATES		
	State Notification Code	**Meaning**
	GPS.MultiGeofence.s<id>=[<state_type>]	True when the device leaves the multigeofence<id>. Example: GPS.MultiGeofence.s1=inside GPS.MultiGeofence.s1=outside

STATES		
	State Notification Code	**Meaning**
	<id>	It specifies the multigeofence ID. It can be set either: **0** ... **2999** - used for user-configured multi-geofences.
	[<state_type>]	Defines the type of an geofence state. **inside** - True when the device enters into the MultiGeofence<id>. Once it results outside the MultiGeofence <id> this state goes false, and the state "*GPS.MultiGeofence.s<id>=outside*" goes true. **outside** - True when the device leaves the MultiGeofence <id>.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	PS.MultiGeofence.e<id>=[<event_type>]	This event can be used to catch events from all multi geofences. Optionally a state (inside or outside) can be defined, which allows to restrict the caught events by the direction (multi geofence entered or left). This alarm can be efficiently used for transmitting the ID (and/or status) of the last multi geofence which was entered or left to a server or the device history. Example: GPS.MultiGeofence.e1=inside GPS.Geofence.e1=outside
	<id>	It specifies the multigeofence ID. It can be set either: **0** ... **2999** - used for user-configured multi-geofences. **x** - used for all pre-configured multi-geofences.
	[<event_type>]	Optional. Define when the event should occur. Event is generated whenever the device moves into/out of the specified multi-geofence (having sufficient GPS coverage). **inside**Occurs when entering the defined Multi-Geofence ID. **outside**Occurs when leaving the defined Multi-Geofence ID. If it is omitted, event occurs when the device enters into and leaves a multigeofence <id>

6.5. EcoDrive

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
	🗝	🗝	🗝	🗝

STATES		
	State Notification Code	**Meaning**
	Ecodrive.sStart	True as long as an EcoDrive trip is active.
	Ecodrive.sStop	True as long as an EcoDrive trip is inactive.
	Ecodrive.sOverSpeed1	True as long as the current speed is between the predefined city and country speed limits.
	Ecodrive.sOverSpeed2	True as long as the current speed is between the predefined country and highway speed limits.
	Ecodrive.sOverSpeed3	True as long as the current speed is over the predefined highway speed limit.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	Ecodrive.eStart	Occurs when an EcoDrive trip is started.
	Ecodrive.eStop	Occurs when an EcoDrive trip is stopped.
	Ecodrive.eHarshTurn	Occurs when the device detects a harsh turn during an active EcoDrive trip.
	Ecodrive.eHarshBrake	Occurs when the device detect a strong brake during an active EcoDrive trip.
	Ecodrive.eHarshAccelerate	Occurs when the device detect a strong acceleration during an active EcoDrive trip.

6.6. GSM

6.6.1. GSM.eJamming

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GSM.eJamming[=<flag>]	The FOX3 series has GSM jamming detection that can be generated when GSM jamming is detected. The feature detects, at radio resource level, an anomalous source of interference and signalling. Occurs when the GSM module detects interference and the device is no longer camped on the serving cell and cannot select any other suitable cell. Once occurred, the device will save data to the device flash memory during the period of jamming and transfer this to the server when the connection is established.
	[<flag>]	**Optional**. Following indicator flags can be set to generate this event: **OK** = 2G jamming no longer detected **Warning** = 2G jamming detected **Critical** = 3G jamming detected

6.6.2. GSM (Operator)

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	GSM.sOpValid [<comp><ID|"operator ">]	True when GSM network operator is available until the GSM network operator results invalid. Once it results invalid, the "*GPS.sOpInvalid..*" state responds true. Example: GSM.sOpValid="T-Mobile D" or GSM.sOpValid="26201"

STATES		
	State Notification Code	**Meaning**
	GSM.sOpInvalid=[<"operator">]	True when the GSM network operator fails until the GSM network operator results valid. Once it results valid the "*GPS.sOpValid.*." state responds true.
	GSM.sRoaming	True when GSM is currently roaming between networks (not registered to home network).
	GSM.sNoRoaming	True when GSM is currently registered into the home network.
	[<ID|"operator">]	The parameter of this event is **optional** and used to launch actions just if this special operator is used. If used, it specifies in quotation marks (" ") the GSM operator name (i.e. "*T-Mobile D*") or without quotation marks the operator ID (i.e. *26201*) to be compared. The comparison is case sensitive and must match exactly that operator name. The mobile network code (operator ID) of the GSM operator is a 5 digit decimal number: first 2 digits are country code, followed by 3 digits containing the operator ID. **Note:** It is possible to enter just the beginning of an operator name or ID – i.e. 26 would match all German operator IDs (26XXX).
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, ==

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GSM.eOpFound[<comp><ID|"operator">]	Occurs when the device finds a (specific) GSM network operator and registered to that GSM operator. Example: GSM.eOpFound="T-Mobile D" or GSM.eOpFound="26201"
	GSM.CBM.e<message_slot>	Occurs whenever new location information is received within the corresponding GSM broadcast message slot. Therefore, this event is ideally suited to create a message containing the corresponding dynamic protocol (i.e. *&(CBM0)*) <message_slot> Number ranging from **0** to **4** which specifies the message slot which contains a new broadcast message now.
	GSM.eOpLost[<comp><ID]"operator">	Occurs when the device loses a GSM network operator name or ID.
	[<ID|"operator">]	The parameter of this event is **optional** and used to launch actions just if is used. If used, it specifies in quotation marks (" ") the GSM operator name (i.e. "*T-Mobile D*") or without quotation marks the operator ID (i.e. *26201*) to be compared. The comparison is case sensitive and must match exactly that operator name. The mobile network code (operator ID) of the GSM operator is a 5 digit decimal number: first 2 digits are country code, followed by 3 digits containing the operator ID. **Note:** It is possible to enter just the beginning of an operator name or ID – i.e. 26 would match all German operator IDs (26XXX).
	GSM.eSimLost	This event is occurred if the SIM card is removed while the device is running. - It can be used i.e. to record history entries if a user illegally removes the SIM card during the device is operating. - the device re-continues operation if the correct SIM card is re-plugged in the SIM card holder (or a different SIM card having the same SIM PIN that is available in the configuration)

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GSM.eMCC<comp><country_code>	Mobile networks are uniquely identified by their mobile country code (MCC) and their mobile network code (MNC). If your application needs to identify the county in which the AVL device is currently operating, then use this event by entering the 3-digit MCC. This event is occurred whenever the current mobile country code changes. This event can be used to i.e. generate messages to record if a user illegally leaves that country. **For Germany:** GSM.eMcc=262 <country_code> Integral decimal number of the mobile country code (e.g. for Germany: MCC=262).
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, ==

6.6.3. GSM.eCellChange

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GSM.eCellChange	Occurs when the device is already connected to the GSM network and it changes the GSM network cell (hand-over). Example: GSM.eCellChange

6.6.4. GSM.VoiceCall

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	×	×	×

Exception: This command is supported only in FOX3-3G AU Variant.

STATES		
	State Notification Code	**Meaning**
	GSM.VoiceCall.sReady	True when the device is ready to receive or dial a voice call (currently there is no incoming or established voice call).
	GSM.VoiceCall.sIncoming=[<"[<"p_number">]	True when the device receives an incoming voice call *[or a call from a specific phone number]*. Example: GSM.VoiceCall.sIncoming="+4913131313"
	GSM.VoiceCall.sRingCounter<comp><value>	True when the ring count (i.e., number of rings) of an incoming call exceeds the maximum ring count defined by the user. Example: GSM.VoiceCall.sRingCounter>2

STATES		
	State Notification Code	**Meaning**
	GSM.VoiceCall.sOutgoing =[<"p_number">]	True when the device makes an outgoing voice call *[or a call from a specific phone number]*, until the call is accepted or cancelled. Example: GSM.VoiceCall.sOutgoing="+4913131313"
	GSM.VoiceCall.sInside=[<" p_number">]	True when a GSM voice call [*or a call from a specific phone number*] is accepted, until the call is cancelled. Example: GSM.VoiceCall.sInside="+4913131313"
	[<"p_number">]	**Optional**. Specifies, in quotation marks (""), the phone number (e.g. +493677XXXX, including country and area code) to be compared. If it is omitted, the event occurs for every incoming/outgoing voice call.
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <= or>=.
	<value>	Specifies an integer value between 0 and 255, which determines the number of rings of an incoming call to wait for release an alarm.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GSM.VoiceCall.eIncoming =[<"[<"p_number">]	Occurs when the device receives a GSM voice call. This event can then decide whether to accept that call or not. Example: GSM.VoiceCall.eIncoming="+4913131313"
	GSM.VoiceCall.eRingStop ped=[<"p_number">]	Occurs when the device is already set into the ringing mode and the caller hangs up the phone. Example: GSM.VoiceCall.eRingStopped="+4913131313"
	GSM.VoiceCall.eOutgoing =[<"p_number">]	Occurs when the device sets up an outgoing voice call, before the recipient's phone rings. Example: GSM.VoiceCall.eOutgoing="+4913131313"
	GSM.VoiceCall.eEstablish ed=[<"p_number">]	Occurs when an incoming voice call has been established successfully. Example: GSM.VoiceCall.eEstablished="+4913131313"
	GSM.VoiceCall.eHangup=[<"p_number">]	Occurs when an established voice call has been hanged up. Example: GSM.VoiceCall.eHangup="+4913131313"
	[<"p_number">]	It is **optional**. It specifies in quotation marks (" ") the phone number (e.g. +493677XXXX, including country and area code) to be compared. If it is omitted, the event occurs for incoming/outgoing voice call.

6.6.5. GSM.SMS

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GSM.SMS.eIncoming	Occurs when the device receives a SMS message
	GSM.SMS.eIncoming.Number=<"p_number">	Occurs when the device receives a SMS message from a specific phone number. Example: GSM.SMS.eIncoming.Number="+4913131313"& GSM.SMS.eIncoming.Text="test"
	GSM.SMS.eIncoming.Text=<"text">	Occurs when the device receives a SMS message with a specific contents. Example: GSM.SMS.eIncoming.Number="+4913131313"& GSM.SMS.eIncoming.Text="test"
	GSM.SMS.eSent	Occurs when the device sends an SMS message.
	GSM.SMS.eSent.Number=<"p_number">	Occurs when the device sends a SMS message to a specific phone number. Example: GSM.SMS.eSent.Number="+4913131313"& GSM.SMS.eSent.Text="test"
	GSM.SMS.eSent.Text=<"text">	Occurs when the device sends an SMS message with a specific contents. Example: GSM.SMS.eSent.Number="+4913131313"& GSM.SMS.eSent.Text="test"
	<"p_number">	It specifies in quotation marks (" ") the phone number (e.g. +493677XXXX, including country and area code) to be compared.
	<"text">	It specifies in quotation marks (" ") the text (i.e. "your text") to be compared. The comparison is case sensitive. **Hint**:Due to both events (".Text="your text"" and ".Number="+493677XXXX"") may occur at the same time, you may combine them within a condition field.

6.6.6. GSM.DataCall

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	Event Notification Code	**Meaning**
	GSM.DataCall.sReady	True when the device is ready to receive a data call, but before the data call is established.
	GSM.DataCall.sIncoming=["p_number"]	True when the device receives an incoming data call *[or a call from a specific phone number]*. Example: GSM.DataCall.sIncoming ="+4913131313"
	GSM.DataCall.sRingCounter<comp><value>	True when the number of ring that have elapsed since an incoming data call was generated match the user specified value. Example: GSM.DataCall.sRingCounter>2
	GSM.DataCall.sInside=["<"p_number"]	True when a GSM data call [*or call from specific phone number*] is accepted, until the call isp cancelled. Example: GSM.DataCall.sInside="+4913131313"
	<"p_number">]	It is **optional**. It specifies in quotation marks (" ") the phone number (e.g. +493677XXXX, including country and area code) to be compared. If it is omitted, the event occurs for incoming/outgoing data call.

STATES		
	Event Notification Code	**Meaning**
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <= or>=.
	<value>	It is an integer value between 0 and 255, which determines the number of rings of an incoming call to wait for release an alarm.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GSM.DataCall.eIncoming=["p_number "]	Occurs when the device receives a CSD call or a CSD call from a specific phone number. Example: GSM.DataCall.eIncoming ="+4913131313"
	GSM.DataCall.eRingStopped=["p_number "]	Occurs when the device is already set into the ringing mode and the caller hangs up the phone. Example: GSM.DataCall.eRingStopped="+4913131313"
	GSM.DataCall.eEstablished=["p_number"]	Occurs when a CSD call has been established successfully. Example: GSM.DataCall.eEstablished="+4913131313"
	GSM.DataCall.eHangup=["p_number"]	Occurs when hanging up an established CSD call Example: GSM.DataCall.eHangup="+4913131313"
	GSM.DataCall.eReceived=["p_number"]	Occurs when receiving an packet. Example: GSM.DataCall.eReceived="User text"
	["p_number"]	**Optional**. Specifies, in quotation marks (""), the phone number (including country and area code e.g. +493677XXX) to be compared. If omitted, the event occurs for all incoming and outgoing data calls.
	["text"]	Text of which the packet starts. This text has to match exactly the packet text (case sensitive) to launch alarm actions. If the packet text contains more characters as specified inside <text> only the specified characters will be compared.

6.6.7. GSM.GPRS

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	GSM.GPRS.sConnecting	True when the device tries to perform a GPRS attach until the GPRS is successfully attached.
	GSM.GPRS.sOnline	True when the device is successfully attached to the GPRS until it is detached.
	GSM.GPRS.sDisconnecting	True when the device tries to perform a GPRS detach until the GPRS successfully detached.
	GSM.GPRS.sOffline	True when the device is successfully detached from the GPRS, or a GPRS connection is currently inactive until the device is attached to the GPRS again.
	GSM.GPRS.sTraffic<comp><value>	True when the complete current GPRS traffic (in Bytes) that match the user specified value.

STATES		
	State Notification Code	**Meaning**
	<comp>	Compares two values and return a Boolean (True/False) value that represents the result of the comparison. It can be set to =, !=, <,>, <= or>=.
	<value>	It is an integer value between **0** and **2147483647** that specifies the number of bytes for GPRS traffic completeness.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	GSM.GPRS.eConnecting	Occurs when the device initiates a connection to the GPRS network.
	GSM.GPRS.eConnected	Occurs when the device is successfully attached to the GPRS network. Depending on the user application, the **GPRS.eConnected** event can execute the **TCP.Client.Connect** command to initiate a TCP connection to the server. How to use such an event, see *4.7.7.1.*
	GSM.GPRS.eDisconnecting	Occurs when the device initiates a disconnection from the GPRS services.
	GSM.GPRS.eDisconnected	The GPRS connection between AVL device and the GPRS network is broken.

6.7. TCP

6.7.1. TCP.Client

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	TCP.Client.sConnecting	True when the device initiates a TCP connection to the server until it is successfully connected. Once connected the event *TCP.Client.sConnected* occurs.
	TCP.Client.sConnected	True when the device is successfully connected to the server until it attempts to disconnect.
	TCP.Client.sDisconnecting	True when the device is trying to disconnect from the server until it is disconnected.
	TCP.Client.sDisconnected	True when the AVL device is successfully disconnected from the server until it attempts to connect.
	TCP.Client.sIdle	True when the TCP connection is inactive until it is connected. (Use **TCP.Client.Connect** command to establish the connection). Usually TCP parameters are configured to automatically connect to the TCP server when GPRS state is already online. This state is used to read the connection state when a manual connection to the server is desired.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	TCP.Client.eConnecting	Occurs when the device initiates a TCP connection to the server, but still not connected. Once connected the event *TCP.Client.eConnected* occurs.
	TCP.Client.eConnected	Occurs when the device has successfully established a TCP connection to the remote server (device may send initial data to the used server for acknowledgment).
	TCP.Client.ePacketSent	Occurs when the device has sent a TCP packet to the connected remote server.
	TCP.Client.ePingSent	Occurs when the server receives a ping from the devices.
	TCP.Client.eReceived[="text"]	Occurs when the device receives a TCP packet (with a specific text) from the server or the command MSG.Event,TCP,<"text"> is executed. ["text"] **Optional**. It specifies in quotation marks (" ") the text (i.e. "your text") to be compared. If it is omitted, the event occurs for each incoming message sent from remote server. This comparison is case sensitive. The text to be sent from the remote server must be terminated with <CR><LF>. Example: TCP.Client.eReceived="test"
	TCP.Client.eDisconnecting	Occurs when the device attempts to disconnect from to the server. Once disconnected the event *TCP.Client.eDisconnected* occurs.
	TCP.Client.eDisconnected	The TCP connection between the device and the server has been broken.
	TCP.Client.eBufferEmpty	This event is generated whenever the outgoing TCP buffer becomes completely empty (if the last packet has been acknowledged by server side). **Notes**:Usually TCP settings are configured to automatically connect to the TCP server when GPRS is online. This state is used to retrieve the connection state if manual TCP connection is desired.
	TCP.Client.eFlashBufferEmpty	This event is generated if the internal non-volatile flash tcp buffer becomes empty (-> the last message has been acknowledged by server). Note that TCP send-mode 2 needs to be configured in order to use Flash TCP Buffer feature. **Notes:**Usually TCP settings are configured to automatically connect to the TCP server when GPRS is online. This state is used to retrieve the connection state if manual TCP connection is desired.

6.7.2. TCP.SMTP

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	TCP.SMTP.eSent	Occurs when the device sends out an e-Mail.
	TCP.SMTP.eFailed	Occurs when the device is not able to complete a send Email operation to a SMTP server. The connection to the SMTP server failed, or authentication failed, or the operation timed out etc.

6.7.3. TCP.UDP

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	None	

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	TCP.UDP.eReceived[="text"]	Occurs when the device receives a UDP packet (with an optional text) from a connected UDP server. Example: TCP.UDP.eReceived or TCP.UDP.eReceived="test"
	["text"]	**Optional**. It specifies in quotation marks (" ") the text (i.e. "your text") to be compared. If it is omitted, the event occurs for each incoming message sent from remote server. This comparison is case sensitive. The text to be sent from the remote server must be terminated with <CR><LF>.

Table 6-2 UDP States and Events

6.7.4. TCP.CLIENT2

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	TCP.Client2.sConnecting	True when the device initiates a TCP connection to the server for the second TCP connection until it is successfully connected. Once connected the event *TCP.Client.sConnected* occurs.
	TCP.Client2.sConnected	True when the device is successfully connected to the server for the second TCP connection until it attempts to disconnect.
	TCP.Client2.sDisconnecting	True when the device is trying to disconnect from the server for the second TCP connection until it is disconnected.
	TCP.Client2.sDisconnected	True when the AVL device is successfully disconnected from the server for the second TCP connection until it attempts to connect.

STATES		
	State Notification Code	**Meaning**
	TCP.Client2.sIdle	True when the TCP connection is inactive until it is connected for the second TCP connection. (Use **TCP.Client2.Connect** command to establish the connection). Usually TCP parameters are configured to automatically connect to the TCP server when GPRS state is already online. This state is used to read the connection state when a manual connection to the server is desired.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	TCP.Client2.eConnecting	Occurs when the device initiates a TCP connection to the server for the second TCP connection, but still not connected. Once connected the event *TCP.Client.eConnected* occurs.
	TCP.Client2.eConnected	Occurs when the device has successfully established a TCP connection to the remote server for the second TCP connection (device may send initial data to the used server for acknowledgement).
	TCP.Client2.eDisconnecting	Occurs when the device attempts to disconnect from to the server for the second TCP connection. Once disconnected the event *TCP.Client.eDisconnected* occurs.
	TCP.Client2.eDisconnected	The TCP connection between the device and the server has been broken for the second TCP connection.

6.7.5. MQTT Client

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	✓	✓	✓	✓

STATES		
	State Notification Code	**Meaning**
	MQTT.Client.sConnecting	True when the device initiates a MQTT connection to the server until it is successfully connected. Once connected the event MQTT.Client.eConnected occurs.
	MQTT.Client.sConnected	True when the device is successfully connected to the server until it attempts to disconnect.
	MQTT.Client.sDisconnecting	True when the device is trying to disconnect from the server until it is disconnected.
	MQTT.Client.sDisconnected	True when the AVL device is successfully disconnected from the server until it attempts to connect.
	MQTT.Client.sIdle	True when the MQTT connection is inactive until it is connected. (Use TCP.MQTT.Connect command to establish the connection). Usually TCP parameters are configured to automatically connect to the server when GPRS state is already online. This state is used to read the connection state when a manual connection to the server is desired.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	MQTT.Client.eConnecting	Occurs when the device initiates a MQTT connection to the server, but still not connected. Once connected the event MQTT.Client.eConnected occurs.
	MQTT.Client.eConnected	Occurs when the device has successfully established a MQTT connection to the server (device may send initial data to the used server for acknowledgement).
	MQTT.Client.ePacketSent	Occurs when the device has sent a TCP packet to the connected server.
	MQTT.Client.ePingSent	Occurs when the server receives a ping from the devices.
	MQTT.Client.eDisconnecting	Occurs when the device attempts to disconnect from to the server. Once disconnected the event MQTT.Client.eDisconnected occurs.
	MQTT.Client.eDisconnected	The MQTT connection between the device and the server has been broken.

6.8. WLAN

DEVICES	FOX3-2G	FOX3-3G	FOX3-4G	BOLERO40
EVENTS	?	?	?	×

STATES		
	State Notification Code	**Meaning**
	WLAN.TCP.sConnected	True when a connection to an access point is established until it disconnects.
	WLAN.TCP.sConnected	True when the device is successfully connected to the server until it attempts to disconnect.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	SYS.WLAN.eConnecting	Occurs when the device starts a connection to an access point. This is done after the end of scan and a scanned SSID was found in WLAN configuration.
	WLAN.eConnected[=<ssid>]	Occurs when the connection to an access point is established
	WLAN.eDisconnected	Occurs when the connection to an access point is broken.
	WLAN.TCP.eConnected	Occurs when the socket connection to the server is established.
	WLAN.TCP.eDisconnected	Occurs when the connection to the server is closed/broken.
	WLAN.eReceived[="text"]	Occurs when the device receives a TCP packet (with a specific text) from the server. Example: WLAN.eReceived="test"
	["ssid"]	The SSID of the connected network is given as a parameter, but it is optional in an alarm configuration. If no SSID is given, the event will trigger for all networks.

EVENTS – *are evaluated just when the event occurs*		
	Event Notification Code	**Meaning**
	["text"]	**Optional**. It specifies in quotation marks (" ") the text (i.e. "your text") to be compared. If it is omitted, the event occurs for each incoming message sent from remote server. This comparison is case sensitive. The text to be sent from the remote server must be terminated with <CR><LF>.

7: Dynamic Entries/Variables

The following table shows the dynamic variables that can be attached to the user specified text when they are requested. The format of the reported value of a dynamic variable can be either as a text, decimal or hex.

For example:

&(Timer1) will report: *erased, initialized, active, inactive, running, paused, armed or disarmed.*

&(VAccu) *will* report a value e.g.: 3.935

The format of the dynamic variable is:

&(dynamic variable) //dynamic variable can be set to one of the entries listed below.

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
SYS	
SecurityLock	Used to report whether or not the AVL device has already been locked. If system AVL has already been locked, the return value is 1, otherwise it returns 0
DeviceName	Used to report the name of the device specified with configuration command.
VAccu	Used to report the internal battery voltage. Note that, the battery voltage will be available approx. 6-10 seconds after the system startup. The unit of the returned value is Volt.
Bat	Used to report the current voltage of the internal battery (or "none" if no internal battery is available). The unit of the returned value is Volt.
Power	Used to report the current external power voltage. The unit of the returned value is Volt.
Timer<index>	Used to request the state of the selected Timer. <index> Determines the index of the timer to be requested. Up to 20 Timers are available. It can be set to a value from 0 to 39.
Trigger<index>	Used to request the state of the selected Trigger. <index> Determines the index of the trigger to be requested. Up to 20 Timers are available It can be set to a value from 0 to 39
Counter<index>	Used to request the state of the selected Counter. <index>Determines the index of the counter to be requested. Up to 20 Timers are available. It can be set to a value from 0 to 39
nvCounter<index>	Used to report the state of the selected non-volatile counter (see PFAL commands for more details). e.g. &(nvCounter0) <index> - Specifies the index of the Counter (0 – 19).
SerialData<port>	Used to report the last string received *(max.1024 bytes)* from the serial port **0** or **1**, which is terminated by a Carriage Return and Line Feed. Serial interface needs to be configured as command mode or event mode. i.e. **$PFAL,MSG.Mode.Serial0[1]=6F,C** - for more details refer to the description of this command. <port> 0 Serial port 0 1 Serial port 1

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
SerialData<port>.h	Used to report in hexadecimal the last string received from the serial port **0** or **1**, which is terminated by a Carriage Return and Line Feed (i.e. if "ABCD" has been received, it reports "41424344". Serial interface needs to be configured as command mode or event mode. i.e. ***$PFAL,MSG.Mode.Serial0[1]=6F,C*** *- for more details refer to the description of this command.* <port> **0** Serial port 0 **1** Serial port 1
SerialData<port>.End<endchars>	Reports the last specified characters (defined by <endchars>) of the incoming data. If <endchars> is bigger than the amount of incoming data within the last event, the full text will be displayed. Currently this dynamic variable is only available for plain text and not for hexadecimal data. <port> **0** Serial port 0 **1** Serial port 1 <endchars> Specifies the number (in range of 0 to 39) of the last characters to be shown.
StartupReason	Used to report in which way system is last started. Following strings can be reported: **Start**Normal start up **Ign**Starts up from **IGN-sleep** **Ring**Starts up from **Ring-sleep** **Time**Starts up from **Timer-sleep**
bin=<value>	Adds specific values into the user text. <value>Specifies hexadecimal (i.e. 0x1F,0xFF) or decimal (31,255) values to be added. Multiple values can be separated by characters which are not part of the value (i.e. comma, space or semicolon etc..).
Attitude	Reports current G-forces for each axis of the attitude sensor.
Force	Reports the G-force value of the last *eForce* event (reports **0** if no force event is available).
MaxMem	Used to report the maximal used up memory after system start. This entry can be used to check remotely if memory is almost used up by i.e. too many alarms.
LastPFALanswer	Used to report the PFAL answer of the last executed PFAL command (on any user interface). At maximum 1024 bytes of PFAL answer can be displayed; an ERROR is shown in case this size is exceeded (i.e. for PFAL,CNF.Show). This entry can be used in combination with the event **Sys.Device.ePfalExecuted** to generate customized server messages containing all PFAL answers. When using this dynamic variable it is strongly recommended not to use.
Temp	Used to report the internal temperature of the FOX3 device. The unit of the returned temperature value is Celsius.

<table>
<tr><th>DYNAMIC VARIABLES</th><th>MEANING (Refer to corresponding PFAL Command)</th></tr>
<tr><td>FAL</td><td>Used to report some GPS data in the FAL format. The format of the reported data looks like following:
saaaaaaa sooooooo ss ccc yymmdd hhmmss iiii oooo
e.g. 0506734800109809100046150304145051010000 10
To decode this data, use the conversation below:
Format Example Description
saaaaaaaa 05067348 = 8 digits, starting with the sign for the latitude (0 = positive(+),9=negative(-)), followed by the latitude of thelocation in decimal format multiplied by 100000
latitude of the
soooooooo 001098091 = 9 digits, starting with the sign for the latitude (0 = positive (+), 9 = negative (-)), followed by the longitude of the location in decimal format multiplied by 100000
location in
ss 00 = 2 digits, represents the speed in decimal
ccc 046 = 3 digits, represents the course in degrees °
yymmdd 150304 = 6 digits, represents the year, month, day
hhmmss 145051 = 6 digits, represents the hour, minute, sec
iiii 0100 = 4 digits, represents in binary the logical state (1 = high and 0 = low) of all digital inputs of the device (1-4)
oooo 0010 = 4 digits, represents in binary the logical state (1 = high and 0 = low) of all digital outputs of the device (1-4)</td></tr>
<tr><td>FALPOS</td><td>Used to report just longitude and latitude data in the FALPOS format. The format of the reported data looks like following:
saaaaaaa sooooooo e.g. 05067348001098091
To decode this data, use the conversation below:
Format Example Description
saaaaaaaa 05067348 = 8 digits, starting with the sign for the latitude (0 = positive(+),9=negative(-)), followed by the latitude of thelocation in decimal format multiplied by 100000
soooooooo 001098091 = 9 digits, starting with the sign for the latitude (0 = positive (+), 9 = negative (-)), followed by the longitude of the location in decimal format multiplied by 100000</td></tr>
<tr><td>Replace<index></td><td>Used to report the text specified into the corresponding REPLACE<index>. <index> = 0..9</td></tr>
<tr><td>VIN</td><td>Used to report the vehicle identification number from config setting CNF.Set,Device.VIN=<”VIN”>. It is a 17-character string.</td></tr>
<tr><td>UserEventText</td><td>Used to report the text (customized command) specified with PFAL command MSG.Event[,<interface>],<"text"></td></tr>
<tr><td>WLAN.MAC</td><td>Prints the MAC address of the WIFI module.</td></tr>
</table>

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
UnixTime2	It is used to report the number of seconds since 1.1.1970 UTC (UNIX epoch time)
USBData	Used to report the last string received (max.1024 bytes) from the USB port, which is terminated by a Carriage Return and Line Feed. USB interface needs to be configured as command mode or event mode. i.e. $PFAL,MSG.Mode.USB=6F,C - for more details refer to the description of this command.
USBData.H	Used to report in hexadecimal the last string received from the serial port 0 or 1, which is terminated by a Carriage Return and Line Feed (i.e. if "ABCD" has been received, it reports "41424344". Serial interface needs to be configured as command mode or event mode. i.e. $PFAL,MSG.Mode.USB=6F,C - for more details refer to the description of this command.
USBData.End<endchars>	Reports the last specified characters (defined by <endchars>) of the incoming data. If <endchars> is bigger than the amount of incoming data within the last event, the full text will be displayed. Currently this dynamic variable is only available for plain text and not for hexadecimal data. <endchars> Specifies the number (in range of 0 to 39) of the last characters to be shown.
SerialID	The serial ID of the device.
ProductionID	The production ID of the device.
ModBus:register	The content of a polled ModBus register.
ModBus:register[*factor][<+->offset]	For numeric values you can use an additional factor and offset to display the register content.
MQTT	
ThingID	The used ThingID for MQTT. These value comes from the MQTT configuration i.e $PFAL,CNF.Set,MQTT.CLIENT.ID=<ThingID> - for more details refer to the description of this command.
LatLon	Used to report the current GPS latitude and longitude value in degrees. Format: is e.g. 50.6731,10.9805.
LastLatLon	Used to report the last valid GPS latitude and longitude value in degrees. Format: is e.g. 50.6731,10.9805.
1-Wire	
1WIRE.LIST	Used to report the IDs of the connected 1-Wire devices (e.g. 10c2272e000000E1)

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
1Wire.Temp[=<sensor_id>]	Used to report the temperature of 1-Wire sensors connected to the FOX3. The unit of the returned value is Celsius. If no device ID is specified, the first found temperature sensor is read out e.g: &(1WIRE.Temp) reports 21°C <sensor_id> - Optional. Specifies the id of the external 1-Wire sensor. The ID can be used when more than one sensor is connected to the FOX3. e.g. `&(1WIRE.Temp=6700080265f19210)` ***Note:*** ◆ *if several temperature sensors are to be used, the specific unique device ID of each sensor is required to address this sensor!* ◆ *This implies, that all devices need a unique configuration, which contains the correct ID's of the attached sensors.* ◆ *This implies a specific installation procedure for car installation, which attaches AND configures the sensors sequentially,* ◆ *This assures to have predefined system behaviour and avoids mixing sensor ID's and the installed location of the sensor.*
1Wire.Temp=all	Used to report the temperatures of all 1-Wire sensors connected to the AVL device.
BLE	
BLE.Name[:index]	Used to report the name of last registered iBeacon sensor. It is set to "unnamed" if it is not advertised from the iBeacon sensor. Use the optional index to select a beacon from the list that is created during scanning.
BLE.relName[:index]	Used to report the name of last released iBeacon. It is set to "unnamed" if it is not advertised from the iBeacon sensor. Use the optional index to select a beacon from the list that is created during scanning.
BLE.RSSI	Used to report the RSSI value in dbm from last registered iBeacon sensor
BLE.Released	Used to report the name of last released iBeacon sensor
BLE.List	Used to report the found devices from last scan listed in the list. Depending on the length of the advertised friendly name of the BLE Beacons, the device can store up to 28 BLE beacons, if name has a length of 24 bytes.
BLE.List2	Used to report the list of iBeacon MACs found after completing a scan process.
BLE.MAC[:index]	Used to report the MAC address of the registered iBeacon sensor. Use the optional index to select a beacon from the list that has been created during scanning.
BLE.relMAC[:index]	Used to report the MAC address of the last released iBeacon. Use the optional index to select a beacon from the list that is created during scanning.
BLE.UUID[:index]	Used to report the universally unique identification (UUID) of a registered iBeacon sensor. Use the optional index to select a beacon from the list that is created during scanning.
BLE.relUUID[:index]	Used to report the universally unique identification (UUID) of the last released iBeacon. Use the optional index to select a beacon from the list that has been created during scanning.

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
BLE.Major[:index]	Used to report the unique major value of the registered and selected iBeacon sensor. Major value is a number assigned to an iBeacon, in order to identify it with greater accuracy than using UUID alone. Major is unsigned integer value between 0 and 65535. The iBeacon standard requires both a Major and Minor value to be assigned. Use the optional index to select a beacon from the list that is created during scanning.
BLE.relMajor[:index]	Used to report the unique major value of the last released iBeacon. Use the optional index to select a beacon from the list that has been created during scanning.
BLE.Minor[:index]	Used to report the unique minor value of the registered and selected iBeacon sensor. Minor value is a number assigned to an iBeacon, in order to identify it with greater accuracy than using UUID alone. Minor is unsigned integer value between 0 and 65535. The iBeacon standard requires both a Major and Minor value to be assigned. Use the optional index to select a beacon from the list that is created during scanning.
BLE.relMinor[:index]	Used to report the unique minor value of the last released iBeacon. Use the optional index to select a beacon from the list that has been created during scanning.
BLE.ListDev[:<content>]	Used to report the list of active beacons or use the optional property to report the list with Name, MAC,UUID, Major or RSSI of active beacons. The optional <content> entry can be set to: Name, MAC,UUID, Major or RSSI.
BLE.ListAdd[:<content >]	Used to report the list of added beacons or use the optional property to report the list with Name, MAC,UUID, Major or RSSI of added beacons. The optional <content> entry can be set to: Name, MAC,UUID, Major or RSSI.
BLE.ListRel[:<content >]	Used to report the list of released beacons or use the optional property to report the list with Name, MAC,UUID, Major or RSSI of released beacons The optional <content> entry can be set to: Name, MAC,UUID, Major.
BLUE ID	
TicketID	Used to report the ID of the last successful executed BlueID command.
MobileDeviceID	Used to report the ID of the registered mobile device (NFC card or mobile with installed BlueID library based app) which sent the last successful executed BlueID command.
NFC	
NFCUID	Reports the UID (manufacturer code) of the NFC card
DTCO.D8	

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
DTCO.D8.COMPLETE	It reports a formatted text. e.g. INFO-DATA:ECU ID: DTCO SW:00 Date/Time: 29.04.2015 09:05:13 +120 min Vehicle: moving, 9.898 km/h, V Vehicle ID: 13ABCDE Vehicle Reg: N BG 112 Driver1 ID: 1100000007158000, ???, card ok Driver2 available, no card Motor: 8192 rpm Distance: 731.485 km, trip: 3172.260 km D1= low D2= low Ign= on Drawer= closed OpMode= operational *42
DTCO.D8.BLOB	It reports a string with comma separated value fields: ECU ID, String ECU SW ID, 2 digits hex year UTC, 4 digits decimal month UTC, 2 digits decimal day UTC, 2 digits decimal hour UTC, 2 digits decimal minute UTC, 2 digits decimal second UTC, 2 digits decimal time offset to UTC, 3 digits decimal work state, 2 digits hex, according J1939, SPN 1612, 1613 d river1, 2 digits hex, according J1939, SPN 1615-1618 driver2, 2 digits hex, according J1939, SPN 1615-1618 status, 2 digits hex, according J1939,SPN 1620 speed, float, 3 significant figures, km/h distance, float, 3 significant figures, km trip distance, float, 3 significant figures, km K-factor, decimal, pulses/km engine speed, decimal, rpm additional, 4 digits hex, additional info D1 D2 Ignition Drawer op-mode vehicle ID, string vehicle reg., string driver1 ID, string driver2 ID, string Output example: "DTCO",00,2015,04,29,09,14,36,120,4b,10,c0,c0,9.898,733.035,3173.810,12000,8192,0150,"ABCD01102013ABCDE","N BG 112 ","1100000007158000",""*68
DTCO.D8.ECU_ID	It reports max. 4 ascii characters. Output example DTCO*58
DTCO.D8.ECU_SW_ID	It reports 2 digits hexadecimal value Output example: 00*00
DTCO.D8.DATE	It reports the date dd.mm.yyyy UTC Output example: 29.04.2015*3B
DTCO.D8.TIME	It reports the time hh:mm:ss in UTC time Output example: 09:31:01*3A

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
DTCO.D8.TIME_ZONE	It reports the Time zone shift ([+/-]mmm) in minutes Output example: +120*33
DTCO.D8.VEHICLE_STATE	It reports a string: standing|moving|error Output example: moving*79
DTCO.D8.VEHICLE_SPEED	It reports a decimal number, 3 significant figures [km/h] Output example: 9.898*17
DTCO.D8.VEHICLE_ID	It reports a string, max 17 characters Output example: ABCD01102013ABCDE*04
DTCO.D8.VEHICLE_REG	It reports a string, max 14 characters Output example: N BG 112 *17
DTCO.D8.DRIVER1.ID	It reports a string, max 18 characters Output example: 1100000007158000*3A
DTCO.D8.DRIVER2.ID	It reports a string, max 18 characters Output example: 1100000007158000*3A
DTCO.D8.DRIVER1.WORKSTATE	It reports string: break|available|working|driving|error Output example: driving*0D
DTCO.D8.DRIVER2.WORKSTATE	It reports a string: break|available|working|driving|error Output example: available*18
DTCO.D8.DRIVER1.TIMESTATE	It reports a string: break within 15 min!!!|break!!!|error|ok Output example: ok*6B
DTCO.D8.DRIVER2.TIMESTATE	It reports a string: break within 15 min!!!|break!!!|error|ok Output example: ok*6B
DTCO.D8.DRIVER1.CARDSTATE	It reports a string: no card|card ok|card error|N/A Output example: card ok*53
DTCO.D8.DRIVER2.CARDSTATE	It reports a string: no card|card ok|card error|error Output example: no card*5B
DTCO.D8.DRIVER1.OVERSPEED	It reports string: overspeed|overspeed error|N/A|ok
DTCO.D8.DRIVER2.OVERSPEED	It reports a string: overspeed|overspeed error|ok
DTCO.D8.DISTANCE	It reports a decimal number, 3 significant figures [km] Output example: 742.055*18
DTCO.D8.TRIPDIST	It reports a decimal number, 3 significant figures [km] Output example: 742.055*18
DTCO.D8.D1	It reports a string: low|high|ERROR|N/A Output example: low*18
DTCO.D8.D2	It reports a string: low|high|ERROR|N/A Output example: low*18
DTCO.D8.IGN	It reports a string: off|on|ERROR|N/A Output example: on*6E
DTCO.D8.DRAWER	It reports a string: open|closed|ERROR|N/A Output example: closed*71
DTCO.D8.OPMODE	It reports a string: not activated|operational|control|calibration|company|unknown|error|N/A Output example: error*1D
CAN **For more details see also** *1.3. Related documents*, ***App Note: CAN Applications with AVL Devices.***	
CAN<msg_slot>	Used to report the current value, in decimal value, of a CAN variable. if no valid message is stored in that slot then an empty string will be reported. <msg_slot> determines the index of the CAN variable to be requested. Up to 45 CAN variables are available. It ranges from **0** to **49**.

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
CanMsg<msg_slot>	Used to report the current message data contents of this CAN message (having valid data in hexadecimal values). <msg_slot> It ranges from **0** to **49**. if no valid CAN message has been received, the dynamic variable is ignored (nothing is reported).
CanMsgData[<msg_slot>]	Used to report all CAN data in hex or just the data within the slot. [<msg_slot>] - Optional. Specifies the slot, from **0** to **49,** of the specified CAN message. If used it reports only the data within the specified slot.
CanMsgDump	Used to report the data of all configured messages (having valid data in hexadecimal values)). If no valid CAN message has been received, the dynamic variable is ignored (nothing will be displayed). This entry must be enclosed in quotation marks **"&(CANMsgDump)"** Output format: [<msg_slot>]:<msg_data>{;[<msg_slot>]:<msg_data>...} **i.e.** 0: A0 B1 C2 D3 E4 F5 06 07;[1]: A0 B1 C2 D3 E4 F5 06 07
CanMsgDump<msg_slot>	Used to report the data in the specified (having valid data in hexadecimal values). If no valid CAN message has been received, the dynamic variable is ignored (nothing will be displayed). This entry must be enclosed in quotation marks **"&(CANMsgDump0)"** <msg_slot> Specifies the slot, from **0** to **49,** of the specified CAN message. Output format: [<msg_slot 0>] **i.e.** 0: A0 B1 C2 D3 E4 F5 06 07
OBDII<obd_id>	Used to report the data of all configured messages ID from 00 – 5F (having valid data in hexadecimal values) of ODBII messages. If no valid message has been received, the dynamic variable is ignored (nothing will be displayed). <obd_id> Specifies a 2 digit Hexadecimal value of OBDII message ID (00 – 5F).
OBDII.DTC	Used to report all OBD device trouble codes. Codes are separated by a space character. Pxxxx - Powertrain followed by a code Cxxxx - Chassis followed by a code Bxxxx - Body followed by a code Uxxxx - Network followed by a code ***Note:*** *To get valid data from this dynamic variable, first execute the command $PFAL,SYS.CAN.OBDII.DTCrq and wait until the event Sys.eOBDII.DTC occurs. This event identifies that the requested data is available now in the dynamic variable. To automate this process, you have to configure an alarm that reports this data once the event is occurred as follow:* *$PFAL,CNF.Set,AL2=Sys.eOBDII.DTC:Msg.Send.Serial0,0,"DTC=&(OBDII .DTC)"*
CO.pdo	Displays the CANopen PDO event string. For more details see also *1.3. Related documents*, **App Note: CAN Applications with AVL Devices.**

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
CanERR	Shows what type of CANbus error has been generated. If more than one error is active, the errors are separated by a comma ",". The error codes are: "STUFF"More than 5 equal bits in a sequence have occurred "FORM"A fixed format part of a received frame has the wrong format "ACK"Sent message was not acknowledged by another node "BIT1"Bit 1 error (monitored bus value was dominant) "BIT0"Bit 0 error (monitored bus value was recessiv) "CRC"The CRC check sum was incorrect in the message received "WARNING_RX"tx error counter (TEC) reached warning level (>96) "WARNING_TX"rx error counter (REC) reached warning level (>96) "PASSIVE"CAN "error passive" occurred "BUS_OFF"CAN "bus off" error occurred "OVERRUN_RX"overrun in RX queue or hardware occurred "OVERRUN_TX"overrun in TX queue occurred "ARBITRATION_LOST"arbitration lost "PHY_FAULT"General failure of physical layer detected (if supported by hardware) "PHY_H"Fault on CAN-H detected (Low Speed CAN) "PHY_LFault on CAN-L detected (Low Speed CAN)
FMS **For more details see also** *1.3. Related documents*, ***App Note: How to collect CAN FMS/J1939/OBD-II Data with Fox3 Series.***	
FMS.ACCEL	Used to report the current accelerator pedal position in % (percent)
FMS.ACCEL1_PEDAL_LOW	Accelerator pedal 1 low idle switch
FMS.ACCEL2_PEDAL	Accelerator pedal position 2
FMS.ACCEL2_PEDAL_LOW	Accelerator pedal 2 low idle switch
FMS.ACCEL_KICKDOWN	Accelerator pedal kickdown switch
FMS.ACCEL_LIMIT	Vehicle acceleration rate limit status
FMS.ACCEL_REM_PDL	Remote accelerator pedal position
FMS.AMB_TEMP	Temperature of air surrounding vehicle
FMS.BRAKE_SWITCH	Used to report the current status of the brake switch: **0** - Brake switch is off **1** - Brake switch is on **err** - Device error **n/a** - Value not available
FMS.CLUTCH_SWITCH	Used to report the current status of the clutch switch **0** – Clutch switch is off **1** – Clutch switch is on **err** – Device error **n/a** – Value not available
FMS.CRUISE_CONTROL	Used to report the current status of the cruise control **0** – Cruise control is off **1** – Cruise control is on **Err** – Device error **n/a** – Value not available
FMS.DELAY_CALENDER	Delay calendar time based - 1 week

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
FMS.DELAY_TIME	Delay time operational time based - 1h (hour)
FMS.DRIVER_ID	Drivers identification
FMS.ENG_CTRL_ADDR	Source address of controlling device for engine control - 8 bit addr
FMS.ENG_FUEL_TEMP1	Engine fuel temperature 1 - 1°C (-40° Offset)
FMS.ENG_HR	Total engine hours (0.05 h / Bit gain; 200.000 = 10.000 h)
FMS.ENG_LOAD	Engine percent load at current speed
FMS.ENG_MAX_TORQUE	Actual maximum available Engine percent torque
FMS.ENG_OIL_TEMP1	Engine oil temperature - 1/32 °C (-273° offset)
FMS.ENG_REV	Total engine revolutions
FMS.ENG_START_MODE	Engine starter mode - 16 modes
FMS.ENG_TORQUE	Actual engine percent torque - 1% offset -125%
FMS.ENG_TORQUE_HR	Actual engine percent torque high resolution part - 0,125% bit3==0->n/a
FMS.ENG_TORQUE_MODE	Engine torque mode - 16 states
FMS.ENGINE_SPEED	Used to report the engine speed in rpm *(rotations per minute).*
FMS.ENGINE_TEMP	Used to report the current engine coolant temperature in °C *(degree).*
FMS.FUEL%	Used to report the fuel level in % (percent).
FMS.FUEL2%	Fuel level 2 - 0.4%
FMS.FUEL_FLTR_PRES	Engine fuel filter differential pressure - 2 kPa
FMS.FUEL_RATE	Fuel_rate - 0.05 l/h
FMS.FUEL_TOTAL	Used to report the total used fuel in l (litre).
FMS_HR_TOTAL_FUEL_USED	Used to report the high resolution total fuel used in ml (millilitre).
FMS.INTERCOOL_TEMP	Intercooler temperature - 1°C (-40° Offset)
FMS.INTERCOOL_TOPEN	Intercooler thermostat opening - 0-100% 0,4%
FMS.MAINTANCE	Used to report the remaining distance to the next regular maintenance in km (kilometres).
FMS.OIL_FLTR_PRES	Engine oil filter differential pressure - 0,5 kPa

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
FMS.PTO	Used to report the current status of the power take off governor. **00** – Off / disabled **01** – Hold **02** – Remote hold **03** – Standby **04** – Remote standby **05** – Set **06** – Decelerate/Coast **07** – Resume **08** – Accelerate **09** – Accelerator override **10** – Preprogrammed set speed 1 **11** – Preprogrammed set speed 2 **12** – Preprogrammed set speed 3 **13** – Preprogrammed set speed 4 **14** – Preprogrammed set speed 5 **15** – Preprogrammed set speed 6 **16** – Preprogrammed set speed 7 **17** – Preprogrammed set speed 8 **18** – PTO set speed memory 1 **19** – PTO set speed memory 2 **31** – Value not available
FMS.PTO_ENG	At least one PTO is engaged
FMS.SERV_COMP1	Service component identification - component ID (Table SPN911_A)
FMS.SERV_COMP2	Service component identification - component ID (Table SPN911_A)
FMS.SERV_COMP3	Service component identification - component ID (Table SPN911_A)
FMS.SPD_LIMIT	Road speed limit status
FMS.SPEED_WB	Used to report the wheel based vehicle speed in cm/s
FMS.SPEED_WB_KMPH	Used to report the wheel based vehicle speed in km/h
FMS.TC_DIR	Used to report the motion direction: **0** – Forward **1** – Reverse **Err** – Device error **n/a** – Value not available
FMS.TC_DRV1_CARD	Used to report the current driver 1 card status: **0** – Not present **1** – Present **Err** – Device error **n/a** – Value not available
FMS.TC_DRV1_STATE	Used to report the current Driver 1 working state: **0** - Rest **1** - Available **2** - Work **3** - Drive **6** - ERROR **7** - n/a

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
FMS.TC_DRV1_TIME	Used to report the current driver 1 time related state: **00** - Normal **01** - 15 min bef. 4½ h **02** - 4½ h reached **03** - 15 min bef. 9 h **04** - 9 h reached **05** - 15 min bef, 16 h **06** - 16 h reached **09** - Other **14** - ERROR **15** - n/a
FMS.TC_DRV2_CARD	Used to report the current driver 2 card status: **0** – Not present **1** – Present **err** – device error **n/a** – value not available
FMS.TC_DRV2_STATE	Used to report the current Driver 2 working state: **0** - Rest **1** - Available **2** - Work **3** - Drive **6** - ERROR **7** - n/a
FMS.TC_DRV2_TIME	Used to report the current driver 2 time related state: **00** - Normal **01** - 15 min bef. 4½ h **02** - 4½ h reached **03** - 15 min bef. 9 h **04** - 9 h reached **05** - 15 min bef, 16 h **06** - 16 h reached **09** - Other **14** - ERROR **15** - n/a
FMS.TC_EVENTS	**0** – No system events **1** – System events **err** – Device error **n/a** – Value not available
FMS.TC_HANDLING	**0** – No handling information present **1** – Handling information present **Err** – Device error **n/a** – Value not available
FMS.TC_MOTION	Used to report the current vehicle motion status: **0** – Not moving **1** – Moving **Err** – device error **n/a** – value not available

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
FMS.TC_OVERSPEED	Used to report whether the TC speed limit is exceeding: **0** – Not exceeding **1** – Exceeding **err** – Device error **n/a** – Value not available
FMS.TC_PERF	Used to report the current tachograph performance status: **0** – Normal performance **1** – Performance analysis **Err** – Device error **n/a** – Value not available
FMS.TC_SHAFT_SPEED	Used to report the calculated output shaft speed in rpm *(revolutions per minute).*
FMS.TC_SPEED	Used to report the calculated output shaft speed in rpm *(revolutions per minute)*
FMS.TC_SPEED_KMPH	Used to report the calculated output shaft speed in rpm *(revolutions per minute).*
FMS.TC_STATE	Used to report all 4 bytes of the current driver working state (*from PGN FE6C*).
FMS.THROTTLE	Throttle position - 0.4%
FMS.THROTTLE2	Throttle position2 - 0.4%
FMS.TORQUE_DRV_DEMAND	Driver's demand engine percent torque - 1% offset -125%
FMS.TORQUE_ENG_DEMAND	Engine demand percent torque - 1% offset -125%
FMS.TOTAL_FUEL_USED	High resolution engine total fuel used
FMS.TURBO_OIL_TEMP	Turbo oil temperature - 1/32 °C (-273° offset)
FMS.VEHICLE_DIST	Used to report the high resolution total vehicle distance in m (meters).
FMS.VEHICLE_ID	Vehicle identification number
FMS.VER_DIAG_SUPP	Used to report the diagnostics status: **0** – Diagnostics not supported **1** – Diagnostics supported **err** – Device error **n/a** – Value not available
FMS.VER_REQU_SUPP	Used to report the requests status: **0** – Requests not supported **1** – Requests supported **err** – Device error **n/a** – Value not available
FMS.VERSION	Used to report the FMS software version as a string.
FMS.WASHER_LVL	Washer fluid level - 0.4%

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
FMS.WEIGHT	Used to report the current status of the vehicle weight based on axis(axle). The output format is <**axis**>:<**weight**>{,<**axis**>:<**weight**>}...: <**axis**>: It is the index (in hexadecimal) of axis (might depend on car type and number of available axis) <**weight**>: It is the weight (decimal) in kg (rounded down to full kg). ***Note:*** *if working with simulated values, messages have to be resent within 20 seconds before reading this protocol, else output will be reset. This allows to i.e. add /remove a trailer (hanger) to/from a lorry and readout updated values.*
J1939 **For more details see also** *1.3. Related documents*, ***App Note: How to collect CAN FMS/J1939/OBD-II Data with FOX3Series.***	
J1939.FUEL_ECO_AVRG	Used to report the average fuel consumption in l / 1000 km (litre/1000 kilometres). Not in FMS-standard.
J1939.FUEL_ECO_INST	Used to report the instant fuel consumption in l / 1000 km (litre/1000 kilometres). Not in FMS-standard.
J1939.FUEL_TRIP	Used to report the used fuel in l (litre). Trip based not in FMS-standard
J1939.PARK_BRAKE_SWITCH	Used to report the current status of the parking brake switch (*additional - it is not in FMS-standard*) **0** – Parking brake switch is off **1** – Parking brake switch is on **err** – Device error **n/a** – Value not available
J1939.VEHICLE_DIST_TRIP	Used to report the high resolution total vehicle distance in m (meters). Not in FMS-standard.
J1939.TIRE2_NOMPRESS	Used to report the nominative tire pressure kPa
J1939.TIRE2_PRESSURE	Used to report the tire pressure kPa
J1939.TIRE_TEMP	Used to report the tire temperature °C 0.03125 K, -273 °C
J1939.TIRE_FAULT	Used to report the tire sensor fault 0 : OK, 1 : Leak
J1939.TIRE_PLOSS	Used to report the tire pressure loss Pa/s (0.1 Pa/s per bit)
J1939.TIRE_PRESSURE	Used to report the tire pressure kPa
J1939.TIRE_PTD	Used to report the tire Pressure Threshold Detection: 1 : Over pressure, 2 : No warning pressure, 3 : Under pressure, 4 : Extreme under pressure)
J1939.TIRE_SEN	Used to report the tire sensor enabled: 0 : off, 1 : enabled
J1939.TIRE_STAT	Used to report the tire status 0 : OK, 1 : Leak
J1939.TIRE_TEMP	Used to report the tire temperature °C 0.03125 K, -273 °C

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
J1939.TIRE_FAULT	Used to report the tire sensor fault: 0 : OK, 1 : Leak e.g: PFAL,CNF.Set,AL10=Sys.eJ1939.TIRE_STAT:Msg.Send.Serial0,0,"J1939.TIRE_STAT=&(J1939.TIRE_STAT)"
J1939.TIRE_STAT	Used to report the tire sensor: 0 : OK, 1 : Leak e.g:$PFAL,CNF.Set,AL10=Sys.eJ1939.TIRE_STAT:Msg.Send.Serial0,0,"J1939.TIRE_STAT=&(J1939.TIRE _STAT)" $PFAL,CNF.Set,AL11=Sys.eJ1939.TIRE_STAT=0:Msg.Send.Serial0,0,"J1939.TIRE_STAT=OK" $PFAL,CNF.Set,AL12=Sys.eJ1939.TIRE_STAT>0:Msg.Send.Serial0,0,"J1939.TIRE_STAT=leak"
ContiPressureCheck™ system **For more details see also** *1.3. Related documents*, ***App Note: ContiPressureCheck™ System Integration into FOX3 Series.***	
TireCondition	
J1939.TIRE_TEMP ≙ PNG 0xFEF433	It reports, into the comma separated value format, the temperature of each wheel as follows. e.g. **00**:21,**01**:22,**10**:20,**11**:20,**12**:20,**13**:21,**20**:21,**21**:22 00 = wheel position, 21 = temperature in °C.
J1939.TIRE2_NOMPRESS ≙ PGN 0xFEF433	It reports, into the comma separated value format, the nominative pressure in kPa of each tire as follows. e.g.: **00**:790,**01**:795,**10**:790,**11**:795,**12**:790,**13**:795,**13**:790,**13**:795 00 = wheel position, 790 = nominative tire pressure in kPa.
J1939.TIRE2_PRESSURE ≙ PGN 0xFEF433	It reports, into the comma separated value format, the pressure of each tire as follows. e.g.: **00**:4,**01**:4,**10**:249,**11**:4,**12**:4,**13**:4,**20**:249,**21**:4 10:790 – 00 = wheel position, 249 = tire pressure in kPa.
J1939.TIRE_PRESSURE ≙ PGN 0xFEF433	It reports, into the comma separated value format, the pressure of each tire as follows. e.g.: **00**:4,**01**:4,**10**:248,**11**:4,**12**:4,**13**:4,**20**:248,**21**:4 10 = wheel position, 248 = tire pressure in kPa.
J1939.TIRE_FAULT ≙ PGN 0xFEF433	It reports, into the comma separated value format, the fault state of each tire as follows. e.g.: **00**:0,**01**:0,**10**:0,**11**:1, **12**:0,**13**:0,**20**:0,**121**:0 11 = wheel position, 0 = no fault; 1 = fault.
J1939.TIRE_PTD ≙ PGN 0xFEF433	It reports, into the comma separated value format, the pressure threshold detection of each tire as follows. e.g.: **00**:4,**01**:2,**10**:2,**11**:2,**12**:4,**13**:2,**20**:2,**21**:2 00 = wheel position; 1 = Over pressure, 2 = No warning pressure, 3 = Under pressure, 4 = Extreme under pressure.
J1939.TIRE_PLOSS ≙ PGN 0xFEF433	It reports, into the comma separated value format, the pressure loss of each tire as follows. e.g.: **00**:0,**01**:0,**10**:0,**11**:0,**12**:0,**13**:0,**20**:0,**21**:0 00 = wheel position; 1 = tire pressure loss in Pa/s (0.1 Pa/s per bit).

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
J1939.TIRE_SEN ≙ PGN 0xFEF433	It reports, into the comma separated value format, if the sensors are enabled as follows. e.g.: **00:**1,**01:**1,**10:**1,**11:**1,**12:**1,**13:**1,**20:**1,**21:**1 00:1 – 00 = wheel position; 0 = Sensor disabled, 1 = Sensor enabled.
J1939.TIRE_STAT ≙ PGN 0xFEF433	It reports, into the comma separated value format, the status of each tire as follows. e.g.: **00:**0,**01:**0,**10:**0,**11:**0,**00:**0,**01:**0,**10:**0,**11:**0 00 = wheel position; 0 = tire OK, 1 = tire leak.
J1939.TIRE_ETP ≙ PGN 0xFEF433	It reports, into the comma separated value format, the while position and if the extended tire pressure is supported as follows. e.g.: **00**:1,**01**:1,**10**:1,**11**:1,**20**:1,**21**:1,**30**:1,**31**:1 00 = wheel position; 0 = Not using Extended Tire Pressure, 1 = Using Extended Tire Pressure, 10: Error, 11: Not available/Not supported
CPC System Configuration	
J1939.TIRE_CPC_CNF_NAXLE ≙ PGN 0xFF0033	It reports the number of axles for the tractor or trailer as follows. e.g.: **00:**4 00 = tractor, 01 = trailer; 4 = Number of axles.
J1939.TIRE_CPC_CNF_NCTTM ≙ PGN 0xFF0033	It reports the number of TTMs for the tractor and trailer as follows. e.g.: **00:**8 00 = tractor, 01 = trailer; 08 = Number of TTMs.
CPC System Status	
J1939.TIRE_CPC_STAT_HEALTH ≙ PGN 0xFF0133	It reports the status of CPC system for the tractor and trailer as follows. e.g.: **00:**1 00 = tractor, 01 = trailer; 0 = OK; “No TTM mounted” NOT detected, 0 = “No TTM mounted” detected.
J1939.TIRE_CPC_NOTTM ≙ PGN 0xFF0133	It reports if no TTMs mounted for the tractor and trailer as follows. e.g.: **00:**0 00 = tractor, 01 = trailer; 0 = OK “No TTM mounted” NOT detected, 0 = “No TTM mounted” detected..
J1939.TIRE_CPC_STAT_WEX ≙ PGN 0xFF0133	It reports, into the comma separated value format, the single wheel exchange as follows. e.g.: **00:**n/a 00 = tractor, 01 = trailer; 0 = “Single wheel exchanged” not detected, 0 = “Single wheel exchanged” detected.
J1939.TIRE_CPC_STAT_LEARN ≙ PGN 0xFF0133	It reports, into the comma separated value format, the automatic trailer learning as follows. e.g.: **00:**0 00 = tractor; 0: Automatic trailer learning ongoing; 1: Automatic trailer learning finished, known trailer found, 2: Automatic trailer learning finished, new trailer found, 3: Automatic trailer learning finished, no trailer found, 4: Feature not active.
CPC System Status	
J1939.TIRE_CPC_STAT_HEALTH ≙ PGN 0xFF0133	It reports the status of CPC system for the tractor and trailer as follows. e.g.: **00:**1 00 = tractor, 01 = trailer; 0 = OK; “No TTM mounted” NOT detected, 0 = “No TTM mounted” detected.

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
J1939.TIRE_CPC_NOTTM ≙ PGN 0xFF0133	It reports if no TTMs mounted for the tractor and trailer as follows. e.g.: **00:**0 00 = tractor, 01 = trailer; 0 = OK “No TTM mounted” NOT detected, 0 = “No TTM mounted” detected..
J1939.TIRE_CPC_STAT_WEX ≙ PGN 0xFF0133	It reports, into the comma separated value format, the single wheel exchange as follows. e.g.: **00:**n/a 00 = tractor, 01 = trailer; 0 = “Single wheel exchanged” not detected, 0 = “Single wheel exchanged” detected.
J1939.TIRE_CPC_STAT_LEARN ≙ PGN 0xFF0133	It reports, into the comma separated value format, the automatic trailer learning as follows. e.g.: **00:**0 00 = tractor; 0: Automatic trailer learning ongoing; 1: Automatic trailer learning finished, known trailer found, 2: Automatic trailer learning finished, new trailer found, 3: Automatic trailer learning finished, no trailer found, 4: Feature not active.
CPC TTM Data	
J1939.TIRE_CPC_TTM_PRESSURE ≙ PGN 0xFF0233	It reports, into the comma separated value format, the TTM pressure 4.706 kPa/bit as follows. e.g.: **00:**aa**,01:**0**,02:**0**,03:**0**,04:0,05:0,06:**0**,07:0,08:**0**,09:**0 00…1F = index range in hex used to identify a sensor id; 0 = Sensor defective or data not available; 01…FF = (aa -1) * 4.706 kPa/bit = 800 kPa, FF=Overflow.
J1939.TIRE_CPC_TTM_TEMP ≙ PGN 0xFF0233	It reports, into the comma separated value format, the TTM temperature 1 °C/bit -50 K offset as follows. e.g.: **00:**22**,01:**22**,02:**22**,03:**22**,04:**22**,05:**22**,06:**22**,07:**22**,08:**22**,09:**22 00…1F = index range in hex used to identify a sensor id; 0 = Sensor defective or data not available; 01…FF = (4F -50K) = 45 °C, FF=Overflow.
J1939.TIRE_CPC_TTM_STATE ≙ PGN 0xFF0233	It reports, into the comma separated value format, the TTM state as follows. e.g.: **00:**8**,01:**8**,02:**8**,03:**8**,04:**8**,05:**8**,06:**8**,07:**8**,08:**8**,09:**8 00…1F = index range in hex used to identify a sensor id; 00-FE = Don’t care, FF = No TTM data since Power On
J1939.TIRE_CPC_TTM_ALARM ≙ PGN 0xFF0233	It reports, into the comma separated value format, the alarm and warning as follows. e.g.: **00:**2**,01:**2**,02:**2**,03:**2**,04:**2**,05:**2**,06:**2**,07:**2**,08:**2**,09:**2 00…1F = index range in hex used to identify a sensor id; 0: OK, 1: Under-inflation warning, 2: Under-inflation alarm, 3: Tire leak alarm, 4: TTM mute, 5: Temperature warning, 8: TTM over temperature warning
J1939.TIRE_CPC_TTM_BAT ≙ PGN 0xFF0233	It reports the battery flag as follows. e.g.: **n/a** n/a = Battery flag.
J1939.TIRE_CPC_TTM_DEFECT ≙ PGN 0xFF0233	It reports the TTM defective as follows. e.g.: **n/a** 00 = wheel position; n/a = TTM defective.

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
J1939.TIRE_CPC_TTM_LOSE ≙ PGN 0xFF0233	It reports, into the comma separated value format, the loose TTM detection as follows. e.g.: **00:**0,**01:**0,**02:**0,**03:**0,**04:**0,**05:**0,**06:**0,**07:**0,**08:**0,**09:**0 00…1F = index range in hex used to identify a sensor id; 0: OK, 1: TTM loose, 2: TTM turned
CPC Graphical Position Configuration	
J1939.TIRE_CPC_POS:H<n> ≙ PGN 0xFF0433	It reports, into the comma separated value format, the value according to the matrix (graphical position and tire location) and the index used as a reference/conjunction for matching the tire location, graphical position and ID of the sensors in hexadecimal as follow s. e.g.: **00:**03,**01:**0b,**02:**43,**03:**4b,**04:**53,**05:**5b 00…1F = index range in hex used to identify a sensor id; 0b = Hexadecimal value to identify the graphical position of the sensors. Refer to the pic. 3 and Table 5 below.
J1939.TIRE_CPC_POS ≙ PGN 0xFF0433	It reports, into the comma separated value format, the value according to the matrix (graphical position and tire location) and the index used as a reference/conjunction for matching the tire location, graphical position and ID of the sensors in decimal as follows. e.g.: **00:**3,**01:**11,**02:**67,**03:**75,**04:**83,**05:**91 00…1F = index range in hex used to identify a sensor id; 0b = Decimal value to identify the graphical position of the sensors. Refer to the pic. 3 and Table 5 below.
J1939.TIRE_CPC_LOC:H<n> ≙ PGN 0xFF0433	It reports, into the comma separated value format, the value according to the matrix (graphical position and tire location) and the index used as a reference/conjunction for matching the tire location, graphical position and ID of the sensors in hexadecimal as follows. e.g.: **00:**00,**01:**01,**02:**10,**03:**11,**04:**20,**05:**21 00…1F = index range in hex used to identify a sensor id; 11 = Hexadecimal value to identify the tire location. Refer to the pic. 3 and Table 5 below.
J1939.TIRE_CPC_LOC ≙ PGN 0xFF0433	It reports, into the comma separated value format, the value according to the matrix (graphical position and tire location) and the index used as a reference/conjunction for matching the tire location, graphical position and ID of the sensors in decimal as follows. e.g.: **00:**0,**01:**1,**02:**16,**03:**17,**04:**32,**05:**33 00…1F = index range in hex used to identify a sensor id; 17 = decimal value to identify the graphical position of the sensors. Refer to the pic. 3 and Table 5 below.
J1939.TIRE_CPC_TTM_ID ≙ PGN 0xFF0433	It reports, into the comma separated value format, the while position and the TTM ID of the sensor in decimal as follows. e.g.: **00:**1835297152,**01:**1835297152,**02:**1821695488,**03:**1818171136,**04:**1825507328,**05:**1825000448,**06:**1821695360,**07:**1821695232 00…1F = index range in hex used to identify a sensor id; 1835297152=TTM ID in decimal value. Refer to the pic. 3 and Table 5 below.
J1939.TIRE_CPC_TTM_ID:H<n> ≙ PGN 0xFF0433	It reports, into the comma separated value format, the while position and the TTM ID of the sensor in <n> decimal digits as follows. e.g.: J1939.TIRE_CPC_TTM_ID:H8 **00:**6d646950,**01:**6d646948,**02:**6c94de3c,**03:**6c5f16e6,**04:**6ccf0811,**05:**6cc74bf2,**06:**6c94dd4e,**07:**6c94dd08 00…1F = index range in hex used to identify a sensor id; 6d646950=TTM ID in hexadecimal value. Refer to the pic. 3 and Table 5 below.

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
J1939.TIRE_CPC_X ≙ PGN 0xFF0433	It reports, into the carriage-return and line-feed, the collection of all CPC messages as follows. e.g.: J1939.TIRE_CPC_CNF_NAXLE:00:4 J1939.TIRE_CPC_CNF_NCTTM:00:8 J1939.TIRE_CPC_STAT_HEALTH:00:1 J1939.TIRE_CPC_STAT_WEX:n/a J1939.TIRE_CPC_STAT_LEARN:00:0 J1939.TIRE_CPC_TTM_PRESSURE:00:aa,01:0,02:0,03:0,04:0,05:0,06:0,07:0,08:0,09:0 J1939.TIRE_CPC_TTM_TEMP:00:22,01:22,02:22,03:22,04:22,05:22,06:22,07:22,08:22,09:22 J1939.TIRE_CPC_TTM_STATE:00:8,01:8,02:8,03:8,04:8,05:8,06:8,07:8,08:8,09:8 J1939.TIRE_CPC_TTM_ALARM:00:2,01:2,02:2,03:2,04:2,05:2,06:2,07:2,08:2,09:2 J1939.TIRE_CPC_TTM_BAT:n/a J1939.TIRE_CPC_TTM_DEFECT:n/a J1939.TIRE_CPC_TTM_LOSE:00:0,01:0,02:0,03:0,04:0,05:0,06:0,07:0,08:0,09:0 J1939.TIRE_CPC_POS:00:3,04:11,08:19,0c:27,10:67,14:75,18:83,1c:91 J1939.TIRE_CPC_LOC:00:0,04:1,08:16,0c:17,10:32,14:33,18:48,1c:49 J1939.TIRE_CPC_TTM_ID:00:1835297152,01:1835297152,02:1821695488,03:1818171136,04:1825507328,05:1825000448,06:1821695360,07:1821695232
DTCO	
Can.Dtco.Incoming	Used to report the incoming data from the tachograph in the format "<SA> <TA> <xx>...<xx>" <SA> hexadecimal source address <TA> hexadecimal target address <xx> hexadecimal data bytes
IO	
IO<index>	Used to report the state of the IO pins supported by the device. <index> Refer to chapter *4.4.*
GPS	
Time	Used to report the current system time. The system time is expressed in Coordinated Universal Time (UTC).
Time.hh	Used to report the hour, in the format "hh", from the current system time. The system time is expressed in Coordinated Universal Time (UTC).
Time.mm	Used to report the minutes, in the format "mm", from the current system time. The system time is expressed in Coordinated Universal Time (UTC).
Time.ss	Used to report the seconds, in the format "ss", from the current system time. The system time is expressed in Coordinated Universal Time (UTC).
Date	Used to report the current system date.
Date.dd	Used to report the days, in the format "dd", from the current system date.
Date.mm	Used to report the months, in the format "mm", from the current system date.
Date.yyyy	Used to report the years, in the format "yyyy", from the current system date.

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
SysTime	Used to request the current internal time of the device. The time in the time format **"hh:mm:ss"**. It is the time since the device has been running. The time value will be set to 0 each approx. 50 days.
SysDate	Used to request the current internal date of the device. The date in the date format **"dd:mm:yy"**. It is the date since the device has been running. The time value will be set to 0 each approx. 50 days.
Lat.n	Used to report the current GPS latitude value in nautical coordinates. Format: is e.g. 50N40'23" for north, 50°40‘23".
sLat	Used to report the current latitude information of the device with sign output format i.e. +50.6733601
Lon.n	Used to report the current GPS longitude value in nautical coordinates. Format: is e.g. 010E58'50 for east, 10°58‘50".
sLon	Used to report the current longitude information of the device with sign output format i.e. : +010.980697
Lat	Used to report the current GPS latitude value in decimal degrees.
Lon	Used to report the current GPS longitude value in decimal degrees.
Alt	Used to report the current GPS altitude value in meters.
Speed	Used to report the current value of the GPS speed in metres/second (its format is **x** ; **x:** 1-3 digits, integral m/s**;** e.g. 3).
Speed.cmps	Used to report the current value of the GPS speed in centimetres/second - floating point representation (its format is **x.yy; x:** 1-3 digits, integral cm/s**; yy:** 2 digits fractional cm/s**;** e.g. 3.45).
Speed.kmh	Used to report the current value of the GPS speed in kilometer/hour - floating point representation (its format is **x.yy; x:** 1-3 digits, integral km/h**; yy:** 2 digits fractional km/h**;** e.g. 3.45).
Speed.mph	Used to report the current value of the GPS speed in miles/hour - floating point representation (its format is **x.yy; x:** 1-3 digits, integral mph**; yy:** 2 digits fractional mp/h**;** e.g. 3.45).
DeltaSpeed.cmps	Used to report the delta value of the GPS speed in centimetres/second - floating point representation (its format is **x.yy** e.g. 3.45).
Course	Used to report the current course over ground, indicating the current direction of the AVL device.
DOP	Used to report the current DOP value (GPS position accuracy error) of the device.
SatsUsed	Used to report the number of satellites that currently are in use.
Fix	Used to report whether or not the AVL device has already got a valid GPS-Fix. If system AVL device has already got a valid GPS-Fix, the return value is 1, otherwise it returns 0.
NavDist	Used to report the actual driven distance in meters since the device has left a known start point. The *navdist* counter supports very high values **2.147.483.648 km** and **higher**. See chapter *4.5.1.5.*
NavDist.km	Used to report the actual driven distance in kilometres since the device has left a known start point. The *navdist* counter supports very high values **2.147.483.648 km** and **higher**. See chapter *4.5.1.5.*
NavDist.miles	Used to report the actual driven distance in miles since the device has left a known start point. The *navdist* counter supports very high values **2.147.483.648 km** and **higher**. See chapter *4.5.1.5.*

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
NavDist2	Used to report the actual driven distance in meters since the device has left a known start point. The *navdist2* counter supports very high values **2.147.483.648 km** and **higher**. See chapter *4.5.1.5.*
NavDist2.km	Used to report the actual driven distance in kilometres since the device has left a known start point. The *navdist2* counter supports very high values **2.147.483.648 km** and **higher**. See chapter *4.5.1.5.*
NavDist2.miles	Used to report the actual driven distance in miles since the device has left a known start point. The *navdist2* counter supports very high values **2.147.483.648 km** and **higher**. See chapter *4.5.1.5.*
DeltaNavDist	Used to report the actual delta distance counter in meters. See chapter *4.5.1.10.*
PosDist<slot_id>	Used to report the actual distance in meters between a stored point and current location of the device. It calculates the air-line distance between two locations by using latitudes and longitudes. <slot_id> specifies a storage index in the range of 0 to 4. By default, the stored point is the center of the Earth. Therefore, to get the correct distance between two points you have to store the current coordinates of the device *(using **$PFAL,GPS.Nav.Position0=current**)* before starting the trip and then request the distance.
GFName<id>	Used to report the name of a configured Geofence. <id> specifies the ID-number of a Geofence. Up to 100 Geofences are available. It can be set to a value from 0 to 99.
GF<id>	Used to report the current state of the selected Geofence. <id> specifies the ID-number of a Geofence. Up to 100 Geofences are available. It can be set to a value from 0 to 99.
MultiGF<id>	Used to report the current state of the selected multi geofence. 1 = inside 0 = outside <id> - Specifies the index of the Multi Geofence (0 - 2999). e.g. $PFAL,MSG.Send.Serial,0,"&(MultiGF234)"
AreaName<id>	Used to report the name of a configured Geofence area. <id> specifies the ID-number of an area. Up to 32 Geofencing areas are available. It can be set to a value from 0 to 31.
Area<id>	Used to report the current state of the selected Geofence area. <id> specifies the ID-number of an area. Up to 32 Geofencing areas are available. It can be set to a value from 0 to 31.
LastGF	Used to report the Geofence identification number <id> on which the last event has been occurred. "**none**" is reported if no GF event has been occurred before.
LastMultiGF	Used to report the index of the last multi geofence being entered or left. e.g. $PFAL,MSG.Send.Serial,0,"&(LastMultiGF)"
LastGFName	Used to report the Geofence name on which the last geofence event has been occurred. "**none**" is reported if no GF event has been occurred before.
LastGFState	Used to report the Geofence state *(**inside -1 or outside -0**)* on which the last event has been occurred. "**none**" is reported if no GF event has been occurred before.

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
LastMultiGFState	Used to report the state of the last entered/left multi geofence. 1 - multi-geofence entered 0 - multi-geofence left e.g: $PFAL,MSG.Send.Serial,0,"&(LastMultiGFState)"
LastArea	Used to report the Area identification number <id> on which the last event has been occurred. "**none**" is reported if no AREA event has been occurred before.
LastAreaName	Used to report the Area name on which the last event has been occurred. "**none**" is reported if no AREA event has been occurred before.
LastAreaState	Used to report the Area state ***(inside -1 or outside -0)*** on which the last event has been occurred. "**none**" is reported, if no AREA event has been occurred before.
LastTime	Used to report the last time of the last valid position.
LastDate	Used to report the last date of the last valid position.
LastLat	Used to report the last latitude of the last valid position.
LastLon	Used to report the last longitude of the last valid position.
LastAlt	Used to report the last altitude of the last valid position.
FAL	Used to report the GPS data in small format (Lantronix format).
FALPOS	Used to report the GPS coordinates in small format (Lantronix format).
ECODRIVE - GPS	
EcoTripID	Reports a string with the current trip ID.
EcoTripCar	Reports a string with the current car name.
EcoTripDist	Reports a string with the current trip distance
EcoTripTime	Reports a string with the current trip time.
EcoTripCurFuel	Reports a string with the current mileage in Litre/100km
EcoTripFuelTotal	Reports a string with the total fuel consumption in Litre
EcoTripCurRoad	Reports a string with the current road type
EcoTripAvgSpeed	Reports a string with the average speed in km/h
EcoTripCurSpeed	Reports a string with the current speed in km/h

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
EcoTripResult	Sends a string with the current trip data to TCP. The trip data are formatted as follows. (see example below) e.g."**15;10157;235000;110;77.1;Highway;20.65;8.5;360;1350;2310;34000;32;3.520;3313; 71000;67;6.834;4944;130000;89;10.268;3**" 15 -- Id trip (starts again at zero after a device reset); 10157 -- Trip time (s); 235000 -- Distance (m); 110-- Current speed (m/s); 77.1-- Avg. speed. (m/s); highway -- Current topology; 20.65 -- Fuel consumption (l); 8.5 -- Current mileage (l/100km); 360 -- Standby time (s); 1350-- Cruise control time (s); 2310 -- City time (s); 34000 -- City distance (m); 32 -- City counter overspeed (s); 3.520 -- City fuel consumption (l); 3313 -- Country time (s); 71000 -- Country distance (m); 67 -- Country counter overspeed (s); 6.834 -- Country fuel consumption (l); 4944 -- Highway time (s); 130000 -- Highway distance (m); 89 -- Highway counter overspeed (s); 10.268 -- Highway fuel consumption (l); 3 -- Counter invalid GPS Info
EcoTripCurData	Reports a string with the last trip data. The trip data are formatted similar to the &(EcoTripCurData) variable.
GSM	
RAT	It is used to report the GSM radio access technology. 0: GSM (2G) 1: GSM COMPACT 2: UTRAN 3: GSM with EDGE availability 4: UTRAN with HSDPA availability 5: UTRAN with HSUPA availability 6: UTRAN with HSDPA and HSUPA availability 7: LTE
IMEI	Used to report the current IMEI number of the device.
SIMID	Used to report the mobile subscriber ID from the used SIM.
SimCCID	Used to report the circuit card ID from the used SIM (production code).
OwnNumber	Used to report the GSM phone number of the inserted SIM card. ***Note:*** *This dynamic variable is only reported if the phone number of the used SIM card is already stored into the SIM card, otherwise the device reports error.*
OperatorID	Used to report the currently used GSM operator ID in decimal (contains MCC & MNC starting with 3 digits MCC followed by 2 digits MNC)
Operator	Used to report the current GSM operator name used by the device.

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
CallerNumber	Used to report the phone number of the currently incoming call. The incoming call may be established, but after it is finished, no caller number will be transmitted (an error will be delivered instead)
OwnNumber	Used to report the phone number of the SIM card used on the AVL device.
Fieldstrength	Used to report the field strength of the currently used GSM cell
CID	Used to report the currently used GSM cell ID in hex.
LAC	Used to report the currently used GSM local area code in hex.
CBM<index>	Used to report the specified message slot content (data of the last received broadcast message). <index> It can be set to **0..4**.
Call	Used to report the state of incoming calls. Possible values may be retrieved: idle incoming voice call (ring:3) incoming data call (ring:1) inside incoming voice call establish data call inside data call outgoing ring inside outgoing voice call
SMSNumber	Used to report the phone number of the last incoming SMS. No value will be shown if there was no SMS received after the system has been started.
SMSText	Used to request the text of last incoming SMS. No value will be shown if there was no SMS received after the system has been started.
GPRSOnline	Used to report information on the supported GPRS service states. It returns the current state of GPRS attachment. If AVL device has already been GPRS attached, the return value is 1, otherwise it returns 0.
GPRSTraffic	Used to report the complete GPRS traffic information. For example: (if GPRS is deactivated). $traffic after 00.00.0,00:00:00 :0 kB (0 Bytes) I:0 kB (0 Bytes) O:0 kB (0 Bytes)
GPRS	Used to report the current GPRS state of the device.
TCP	
TCPClientOnline	Used to report whether or not the AVL device has already established a TCP connection to the remote server. If system AVL device is TCP connected, the return value is 1, otherwise it returns 0.
TCPClient	Used to report the TCP state of the device.
TCPText	Used to report the last received TCP packet.
TCPClient2Online	Used to report whether or not the AVL device has already established a TCP connection to the remote server. If system AVL device is TCP connected, the return value is 1, otherwise it returns 0.
TCPClient2	Used to report the status of the second TCP connection in textual form (Connecting/Connected/Disconnecting/Disconnected)
TCP2Text	Used to output the last received TCP packet on the second TCP connection.
UDP	
UDPText	Used to report the received data from the connected UDP server (max.1024 bytes).

DYNAMIC VARIABLES	MEANING (Refer to corresponding PFAL Command)
IOBOX	
IOBOX_ANA<index>,<port>	Used to report the voltage of the specified analogue input port of the IOBOX device: <index>:Specifies the index of the IOBOX device <port>:Specifies the port number (**0** or **1**) of the analogue input on the IOBOX

8: Application Guide

The AVL device operating with firmware version later can be simply configured using the Lantronix Workbench Software. For more details, please refer to ***App Note: Remote Firmware Update with Workbench Software.*** See *1.3. Related documents*.

For more detailed information how to setup a serial connection, please refer to the ***FOX3 SERIES Promotion Kit User Guide.*** See *1.3. Related documents*.

8.1. Rules to be considered

- Read carefully this document and write down all events, states and actions, which are supposed to be used in your application.
- Decide, what kind of alarms will be executed and channel(s) to be transmitted along (SMS, TCP).
- Decide, when these alarms will execute
- Determine, which system state must already be present and which particular event the system will be waiting for to execute one or more alarms.
- Try always to filter alarms by using 1 event and up to 4 states. A small difference between alarms enables you to reproduce more than 1 alarm for one event.
- Make sure, when an event occurs, it does not affect other alarms except the premeditated ones.
- Use TIMER-events to activate events and execute actions at regular interval. Use TRIGGER-events to execute various actions at a particular time. Use COUNTER-events to limit the number of alarms. However, be careful when you use them. Write down all alarms, which will be executed either by TIMER, COUNTER or TRIGGER and compare them with other already configured alarms, to check if conflicts between them take place. They should never create a situation in which an empty action causes an error, the errors may be caused repeatedly affecting on system performances.
- Avoid executing of false alarms when TIMER, COUNTER and TRIGGER are in use.

8.2. Start a GPRS/TCP connection

To get connected to the GPRS and a TCP server, you have to change all GPRS and TCP settings to your application conditions. These settings can be changed using the command "$PFAL,Cnf.Set,<parameter>*Checksumm". When these settings are changed, then you have to enter the PIN number of the SIM card the device is using. *Your SIM card must already be inserted into the SIM card holder on the device.*

Note: *The configuration given in the table below must be done locally via serial port.*

Table 8-1 Adapt configuration settings to application conditions

SETUP	Enter the GPRS Settings, only if the Remote server is already available and your application interface requires a TCP connection: $PFAL,Cnf.Set,GPRS.APN=internet.t-d1.de (for example) $PFAL,Cnf.Set,GPRS.QOS=3,4,3,0,0 $PFAL,Cnf.Set,GPRS.QOSMIN=0,0,0,0,0 $PFAL,Cnf.Set,PPP.USERNAME=t-d1 (if your provider requires) $PFAL,Cnf.Set,PPP.PASSWORD=gprs (if your provider requires)
SETUP	Enter the TCP Settings, only if Remote server is already available and your application interface requires a TCP connection: IP address and port number should match correctly $PFAL,Cnf.Set,TCP.CLIENT.CONNECT=1,2222.222.222.222,2222 (for example)

SETUP	Enter the SIM PIN: $PFAL,Cnf.Set,DEVICE.PIN=1111 (for example)
SETUP	Configure the GPRS autostart: $PFAL,Cnf.Set,GPRS.AUTOSTART=1
SETUP	Backup your configuration settings: $PFAL,Cnf.Backup
COMMENT	It is strongly recommended that after you finish the configuration of an AVL device you have to execute this to back up your configuration settings. This is recommended that whenever a corrupted configuration is detected, the system will automatically load and start with the backed up configuration settings.

Once you enter the PIN number of the SIM card that the device is using and GPRS autostart setting is set to GPRS.AUTOSTART=1, then the AVL device will automatically try to register first to the GSM network and to connect to the GPRS and TCP services using your defined settings. To check the GPRS and TCP connection states you must monitor both events **GSM.GPRS.eConnected** and **TCP.Client.eConnected** which are outputted on the terminal program which is monitoring the serial port of the device. Note that, both events can be shown only if the debug port of the device is already enabled by using the configuration *($PFAL,Cnf.Set,DBG.EN=1)*.

If problems are met, regarding the TCP settings, contact your network administrator and request the correct IP address, Port number and login data of the server the device is going to connect. Please note that, the AVL device is responsible for initiating the connection to the remote server using the IP address and the Port number that you specified, while the used remote server is responsible for accepting the connection requested from the AVL device based on the logging data the device sends out.

9: Sending SMS to Lantronix Devices

Write Command	The text mode has to be set using at+cmgf=1 AT+.CMGS=<FOX3_phone_number><CR> <text> <ctrl-z>/<esc>
Response	+ CMGS: <m_ref> OK/ERROR

Command Description

The write command transmits a short message from a GSM modem to an AVL device using a terminal program. After invoking the write command wait for the prompt ">" and then start to write the message or PFAL command to be sent to the AVL device. To send the entered message or PFAL command, simply press <ctrl-z>. After the prompt a timer will be started to observe the input. To abort sending, use <esc>. Abortion is acknowledged with "OK", though the message will not be sent. The message reference <+ CMGS: <m_ref>> is returned to the GSM modem on successful message delivery.

This description is applied for the GSM modems distributed by Lantronix. How to send SMS messages using other modems or mobile phones, please, refer to their user's guides.

Parameter description

<FOX3_phone_number>

Specifies the phone number of the remote AVL device.

<CR>

Specifies the <RETURN> key or carriage return ASCI code (13), which has to be entered to enable the text entry <text>.

<text>

Specifies the message or PFAL command (without "$" dollar sign) to be sent to the AVL device (e.g. PFAL,Cnf.Set,DEVICE.NAME=myFOX3). If you are using a Mobile Phone just enter the SMS message or PFAL command (without "$" dollar sign) and then send it out to the remote AVL device.

<ctrl-z>

Specifies the keyboard shortcut <**CTRL**+**Z**> for sending the message to the specified phone number.

Notes

- SMS messages sent to the AVL device must not be longer than 160 characters.
- The maximum length of the SMS to be sent from the AVL device to the receiver is predefined up to 160 characters using the 7-bit GSM coding scheme. If the length of the SMS is longer than 160 characters, the AVL device does not split and deliver that SMS message in two or more messages. The specified protocols to be sent out that exceed the maximum length of the SMS (>160 characters) will be not attached into (delivered with) that SMS message.
- The text entered behind the prompt ">" will be recognized by the AVL device as an input message.

Example:

```
AT+CMGS=012345678
>PFAL,Cnf.Set,DEVICE.NAME=myFOX3<ctrl+z>
 or
>PFAL,Cnf.Set,AL0=IO.e1=short:GSM.SMS.Send,"+491234567",8,"test"<ctrl+z>
```

10: NMEA and Lantronix Messages

The AVL device transmits NMEA sentences every second, depending on the configuration. The identifiers for the NMEA messages transmitted by the AVL device are listed below. Excepting **GPIOP, GPGSM, AREA** and **BIN** all other messages are based on the NMEA standard messages.

GPGGA	GPS Fix Data
GPRMC	Recommended Minimum Specific GPS Data
GPGSV	GPS Satellites in View
GPGSA	GPS DOP and Active Satellites
GPVTG	Course Over Ground and Ground Speed
GLGSA	GNSS DOP and Active Satellites
GLGSV	GNSS satellites in view.
GPGLL	Geographic Position in Latitude/Longitude
GPIOP	Device Input/Output Ports
GPGSM	GSM operator and reception status
AREA	AREA states
3DP	Motion sensor data
BIN	Binary format

A full description and definition of the listed messages above is provided in the next sections of this chapter.

10.1. Description of NMEA output messages

The following table is intended as a quick reference to explain the formats used in the tables below:

Table 10-1 Description of NMEA output messages

Format	Description
hhmmss.ss	Time: hh hours, mm minutes, ss.ss seconds.
ddmmyy	Date: day dd, month mm, year yy.
ddmm.mmmm	Latitude: dd degrees, mm.mmmm minutes.
dddmm.mmmm	Longitude: ddd degrees, mm.mmmm minutes.
dd.dddddd	Latitude/longitude: dd.dddddd degrees.
dd'mm'ss"	Latitude/longitude: dd degrees, mm minutes, ss seconds
x	Integer.
xx	Integer having exactly two digits (using leading zeros).
x.x	Number including fraction.
hh	Two-digit hexadecimal number (using uppercase A–F).
bbbbbbbb	Eight-digit binary number.
a	ASCII text.
"a"	ASCII text in quotation marks.
<CR><LF>	Carriage return and line feed.

10.2. $GPGGA message

The $GPGGA message includes time, position, GPS quality and number of satellites in use.

Example:

$GPGGA,133726.569,5040.4365,N,01058.5646,E,1,03,8.9,92.9,M,,,,0000*3F

$GPGGA,093214.000,4121.9985,N,00210.3737,E,6,00,0.0,630.0,M,0.0,M,0.0,0000*45

Table 10-2 Description of NMEA output messages

Field	Format	Example	Description
1	$GPGGA	$GPGGA	Start of sentence
2	hhmmss.ss	133726.569	UTC time
3	ddmm.mmmm	5040.4365	Latitude
4	a	N	Latitude direction (N/S)
5	dddmm.mmmm	01058.5646	Longitude
6	a	E	Longitude direction (W/E)
7	x	1	GPS fix quality: 0 = No fix, 1 = Autonomous GNSS fix, 2 =Differential GNSS fix, 4 = RTK fixed, 5= RTK float, 6 = Estimated/Dead reckoning fix(3)
8	xx	03	Number of satellites in use
9	x.x	8.9	Horizontal dilution of precision (relative accuracy of horizontal position)
10	x.x	92.9	Altitude above mean sea level (geoid)
11	M	M	Altitude units (meters)
12	x.x		Height of geoid above earth ellipsoid
13	M		Geoid height units (meters)
14	x		Time since last DGPS update (seconds)
15	xxxx	0000	DGPS reference station ID
16	*hh	*3F	Checksum
17	<CR><LF>		End of message termination

10.3. $GPRMC message

The $GPRMC message includes time, date, position, course and speed data.

Example: **$GPRMC,133725.569,A,5040.4365,N,01058.5650,E,0.05,302.98,251004,,*00**

Table 10-3 GPRMC message data format

Field	Format	Example	Description
1	$GPRMC	$GPRMC	Start of sentence
2	hhmmss.ss	133725.569	UTC time
3	a	A	Position validity (A: valid, V: invalid)
4	ddmm.mmmm	5040.4365	Latitude
5	a	N	Latitude direction (N/S)
6	dddmm.mmmm	01058.5650	Longitude
7	a	E	Longitude direction (W/E)
8	x.x	0.05	Speed (knots)
9	x.x	302.98	Heading (degrees)
10	ddmmyy	251004	Date
11	x.x		Magnetic variation (degrees)
12	a		Magnetic variation direction (W/E)
13	*hh	*00	Checksum
14	<CR><LF>		End of message termination

10.4. $GPGSV message

The $GPGSV includes the number of satellites in view, satellite ID numbers and their evaluation, azimuth and signal-to-noise ratio.

Example: **$GPGSV,3,1,10,05,79,067,39,30,63,277,35,14,37,269,,09,36,145,*78**

Example: **$GPGSV,3,2,10,24,28,098,36,06,24,212,,04,24,058,29,17,16,129,*7F**

Example: **$GPGSV,3,3,10,01,13,328,34,25,05,311,*74**

Table 10-4 GPGSV message data format

Field	Format	Example	Description
1	$GPGSV	$GPGSV	Start of sentence
2	x	3	Number of messages (1 to 3)
3	x	3	Message number (1 to 3)
4	xx	10	Number of satellites in view (1 to 12)
5	xx	01	Satellite PRN number - Range 1 to 51 (SBAS ranges from 32...51)
6	xx	14	Satellite elevation (degrees) (00 to 90), may be null
7	xxx	328	Satellite azimuth (degrees) (000 to 359), may be null
8	xx	34	Satellite signal to noise ratio in dB (00 to 99), may be null
9		25	Similar to 5–8 for next satellite, may all be null
10		05	Similar to 5–8 for next satellite, may all be null
11		311	Similar to 5–8 for next satellite, may all be null
12	*hh	*74	Checksum
13	<CR><LF>		End of message termination

10.5. $GPGSA message

The $GPGSA message includes the list of satellites being used.

Example: **$GPGSA,A,2,05,09,04,,,,,,,,,13.4,8.9,10.0*3D**

Table 10-5 GPGSV message data format

Field	Format	Example	Description
1	$GPGSA	$GPGSA	Start of sentence
2	a	A	Operating mode: M: Manual, operate in 3-D mode. A: Automatically choose 2-D or 3-D mode.
3	x	2	Fix mode: 1: Fix not available 2: 2-D fix 3: 3-D fix
4	xx,xx, . . .	05	PRN numbers of satellites in use (unused fields null)
5	x.x	13.4	Position dilution of precision
6	x.x	8.9	Horizontal dilution of precision
7	x.x	10.0	Vertical dilution of precision
8	*hh	*3D	Checksum
9	<CR><LF>		End of message termination

10.6. $GPVTG message

The $GPVTG message includes course over ground and ground speed.

Example: **$GPVTG,309.62,T, ,M,0.13,N,0.2,K,A*23**

Table 10-6 GPVTG message data format

Field	Format	Example	Description
1	$GPVTG	$GPVTG	Start of VTG sentence
2	x.x	309,62	Course over ground (True) in degrees
3	a	T	Fixed field (True)
4	-	-	Course over ground (magnetic) in degrees, not output
5	a	M	Fixed field: magnetic
6	x.x	0,13	Speed over ground - horizontal speed in knots
7	a	N	Fixed field: knots
8	x.x	0,2	Speed over ground - horizontal speed in (km/h)
9	a	K	Fixed field: kilometers per hour
	a	A	Mode Indicator: N=No Fix, A=Autonomous Fix, D=Differential Fix, E=Estimated/Dead Reckoning Fix
10	*hh	*23	Checksum
11	<CR><LF>		End of message termination

10.7. $GLGSA message

The $GLGSA message includes GNSS DOP and active satellites.

Example: **$GLGSA,A,3,65,71,73,80,81,88,,,,,,,4.0,2.0,3.5*2E**

Table 10-7 GLGSA message data format

Field	Format	Example	Description
1	$aaaaa	$GLGSA	Start of GSA sentence
2	a	A	Operation mode: Automatically switching between 2D or 3D mode
3	x	3	Navigation mode: 1 = Fix not available, 2 = 2D Fix, 3 = 3D Fix
4..15	xx	65	Satellite numbers separated by commas: Start of repeated block (12 times)
16	x.xx	4.0	Position dilution of precision
17	x.xx	2.0	Horizontal dilution of precision
18	x.xx	3.5	Vertical dilution of precision
19	*hh	*2E	Checksum
20	<CR><LF>	-	End of message termination

10.8. $GLGSV message

The $GLGSV message includes GNSS satellites in view.

Example: **$GLGSV,2,1,06,65,27,299,32,71,37,096,20,73,48,056,27,80,08,021,28*6D**

Example: **$GLGSV,2,2,06,81,13,347,20,88,27,293,25*60**

Table 10-8 GLGSV message data format

Field	Format	Example	Description
1	$xxGSV	$GPGSV	Start of GSV sentence
2	x	2	Number of messages, total number of GSV messages being output
3	x	1	Number of this message
4	xx	06	Number of satellites in view:
5,9,13,17	xx	65	Satellite IDs.
6,10,14,18	xx	27	Elevation (range 0-90) for each satellite ID
7,11,15,19	xxx	299	Azimuth, (range 0-359) for each satellite ID
8,12,16,20		32	Signal strength in dBHz (C/N0, range 0-99) for each satellite ID, blank when not tracking
19	*hh	*6D	Checksum
20	<CR><LF>	-	End of message termination

10.9. $GPGLL message

The $GPGLL message includes the latitude, longitude, UTC time of position fix and status.

Example: **$GPGLL,5040.4025,N,01058.8342,E,113704.665,A*32**

Table 10-9 GPGLL message data format

Field	Format	Example	Description
1	$GPGLL	$GPGLL	Start of sentence
2	ddmm.mmmm	5040.4025	Latitude
3	a	N	Latitude direction (N/S)
4	dddmm.mmmm	01058.8342	Longitude
5	a	E	Longitude direction (W/E)
6	hhmmss.sss	113704.665	UTC Position
7	a	A	Position validity (A: valid, V: invalid or S*: invalid)
8	*hh	*32	Checksum
9	<CR><LF>		End of message termination

10.10. $GPIOP message

The $GPIOP message includes the status of the digital/analog inputs and output ports .

Example: **$GPIOP,00001000,00000010,0.00,0.28,0.00,0.28,11.90,4.15*72**

Table 10-10 GPIOP message data format

Field	Format	Example	Description
1	$GPIOP	$GPIOP	Start of sentence
2	bbbbbbbb	00000100	Inputs (IN7-IN0; IN7=IGN) (1: high, 0: low)
3	bbbbbbbb	00000001	Outputs (OUT3-OUT0): 7-*4 (unused);* (1: on, 0: off)
4	x.xx	0.00	Analog input 0 (V) - IN0
5	x.xx	0.28	Analog input 1 (V) - IN1
6	x.xx	0.00	Analog input 2 (V) - IN2
7	x.xx	0.28	Analog input 3 (V) - IN3
8	x.xx	11.90	Main power voltage (V)
9	x.xx	4.15	Internal battery voltage (V)
10	*hh	*72	Checksum
11	<CR><LF>		End of message termination

10.11. $GPGSM message

The $GPGSM message includes the GSM operator and reception status. Table below shows mode 0 indicating the GSM status.

Example: **$GPGSM,0,1,0,"T-Mobile D",20,5518,4caa*32**

Table 10-11 GPGSM message data format

Field	Format	Example	Description
1	$GPGSM	$GPGSM	Start of sentence
2	x	0	GSM status mode (RAT). (0: GSM (2G), 1: GSM COMPACT, 2: UTRAN, 3: GSM with EDGE availability 4: UTRAN with HSDPA availability, 5: UTRAN with HSUPA availability, 6: UTRAN with HSDPA and HSUPA availability, 7: LTE
3	b	1	Registration (1:registered, 0: unregistered, 2: Not registered, but searching, 3: Registration denied, 4: Unknown 5: roaming)
4	x	0	Phone activity status: (0: ready; 1: unavailable; 2: unknown; 3: ringing; 4: call in progress)
5	"a"	"T-Mobile D"	Network operator name
6	xx	20	GSM field strength (0 to 31) 0: $-$113 dB 31: $-$51 dB
7	a	5518	Area code
8	a	4caa	Cell ID
9	*hh	*32	Checksum
10	<CR><LF>		End of message termination

10.12. $GPAREA message

The *$GPAREA* message includes state of 32 areas. The example below shows that the AVL device is currently inside the **Area0** and outside all other areas.

Example: **$GPAREA,0000 0001*0D**

Table 10-12 GPAREA message data format

Field	Format	Example	Description
1	$GPAREA	$GPAREA	Start of sentence
2	xxxx	0000	Areas 31 to 16 (16 bit -> 2 byte Hexadecimal value) The hexadecimal value represents the index of the entered area.
3	xxxx	0001	Areas 15 to 0 (16 bit -> 2 byte Hexadecimal value) The hexadecimal value represents the index of the entered area.
4	<CR><LF>		End of message termination

10.13. $GP3DP message

The $GP3GP message displays current min, max and average G-forces applied to the device. This message is periodically transmitted (if enabled) to the serial ports of the AVL device.

Example: **$GP3DP,-29,-142,1095,-36,-148,1092,-20,-136,1100,4*31**

Table 10-13 GP3DP message data format

Field	Format	Example	Description
1	$GP3DP	$GP3GP	Start of sentence
2	xxx	-29	Average x-axis value
3	xxx	-142	Average y-axis value
4	xxx	1095	Average z-axis value
5	xxx	-36	Minimum x-axis value
6	xxx	-148	Minimum y-axis value
7	xxx	1092	Minimum z-axis value
8	xxx	-20	Maximum x-axis value
9	xxx	-136	Maximum y-axis value
10	xxx	1100	Maximum z-axis value
11	xxxx	4	The time on which these values are checked, and event is generated.
12	<CR><LF>		End of message termination

10.14. BIN protocol

The binary format sent from the AVL device has the following structure.

Example: **<Start text><Binary protocol><CRLF>**

	Format	Format	End Sequence
	Start text	<Binary protocol>+<altitude>	<CR><LF>
Example	24	1305 820D23331E34231B068B8627 0001000000E6	0D0A
			It is converted in the hexadecimal format

Field	Format	Example	Description
1	Start text	$	The text specified by the PROT.BIN.START parameter, "$"=default. See chapter *5.7.2.*
2	Binary protocol	<binary protocol>	See table below.
3	<CR><LF>	0D0A	End of message termination (2 bytes)

The following table is intended as a quick reference to explain the sent binary data.

Protocol 0x1000: <binary protocol>=<DATE><VALID><TIME><LAT><LON><SPEED><COURSE>

Protocol 0x4000: <binary protocol+altitude>=<DATE><VALID><TIME><LAT><LON><SPEED><COURSE><ALTITUDE>

Field	Name	Format	Bits	Bit Selection	Range	Example	Description
1	DATE	dd mm yy	16	11…15 = Day (5 bits) 7…10 = Month (4 bits) 0…6 = Year (7 bits)	1..31 1..12 00..99	02 06 05	DATE format including the current Day, Month and year. (e.g. 02.06.05)
2	VALID	v		31 in the TIME format	0..1	1	The current GPS validity bit 31 of the TIME format (1=valid; 0=invalid), see field 3.
3	TIME	v hh mm ms	32	31 = See field 2. 22…26= Hours (5 bits) 16…21= minutes (6 bits) 0..15= msec. (16 bits)	0..1 0..23 00..59 0..59999	1 08 13 9011	TIME format including the current GPS validity (1=valid; 0=invalid), hour, minutes and milliseconds.
4	LAT	xxxxxxx	32	0..31=Latitude (32 bits)	0..4294967295	506733339*	LAT format [+90? to –90?] with resolution of 0.0000001. The value represents a two's complement number of latitude - see below.

Field	Name	Format	Bits	Bit Selection	Range	Example	Description
5	LON	xxxxxxx	32	0.31=Longitude (32 bits)	0..4294967295	109807143*	LON format [+180? to –180?] with resolution of 0.0000001. The value represents a two's complement number of latitude - see below.
6	SPEED	xxxx	16	0..15=Speed (16 bits)	0..65535	1	Speed Over Ground in m/sec with resolution of 0.01
7	COURSE	xxxx	16	0..15=course (16 bits)	0..65535	0	Course Over Ground [0? to 360?] with resolution of 0.01
8	ALTITUDE	xxxx	16	0..15=Altitude(16 bits)	-32768 ...32767	230*	GPS altitude in meter. Byte21: MSB of two's complement integral value; Byte22: LSB of two's complement integral value. This field is available in the protocol 0x4000 only.

* The most significant (leftmost) bit indicates the sign of the LAT, LON and Altitude value:

- If the sign bit (leftmost) is 0, then the LAT, LON and Altitude are positive (e.g. 109807143 =00011010000101110000110001001 11).
- If the sign bit (leftmost) is 1, then the LAT, LON and Altitude are negative. LAT, LON and Altitude should be inverted by applying bitwise NOT (e.g. **LON** = **in hex** [fbe81682] ? **in decimal** [4226291330] ? **in binary** [11111011111010000001011010000010] ? **bitwise NOT** [00000100000101111110100101111101] ? **add 1** [00000100000101111110100101111110] ? **convert to decimal** [68675966]). To calculate it, invert the 32-bit digits by changing all of the 1 to 0 and all of the 0 to 1, then add 1 to inverted binary value and convert the result to decimal. Finally, multiply it by 0.0000001 and place a "−" sign in front of it. The LON in decimal results [-6.8675966°]. The same procedure should be made for negative LAT.

11: Appendix

11.1. How to update new firmware

Please refer to the AVL Firmware Release Notes, available from the Lantronix Tech Support FOX3 and BOLERO40 Series firmware downloads page.

11.2. Supported protocols

Following are listed the protocols in the hexadecimal format. The value to be set must be in hexadecimal format without leading "0x". Depending on user-defined format in the parameter, the protocols and user text including the dynamic variables received on the server site may look as follow:

```
For example, you have configured the following format syntax (default):
$PFAL,Cnf.Set,DEVICE.PFAL.SEND.FORMAT="$",CKSUM,"","<end>"
and you want to transmit the device position to the TCP server using the following PFAL-Command:
$PFAL,MSG.Send.TCP,8,"GPS positions"
```

The data received on the server side, looks in this order:

```
$GPS positions<CRLF>
$GPRMC,133725.569,A,5040.4365,N,01058.5650,E,0.05,302.98,251004,,*00<CRLF>
<end><CRLF>
```

If you configure the following format syntax:

```
$PFAL,Cnf.Set,DEVICE.PFAL.SEND.FORMAT="$",CKSUM,"&","<end>"
```

then the data received on the server side, looks in this order:

```
$GPS positions&<CRLF>
$GPRMC,133725.569,A,5040.4365,N,01058.5650,E,0.05,302.98,251004,,*00&<CRLF>
<end><CRLF>
```

For more details about the format of the supported protocols refer to chapter *10.1. Description of NMEA output messages*.

Table 11-1 Supported protocols

Protocols In hex format	Meaning
01	Requests *$GPGGA message*. This protocol is transmitted in text format.
02	Requests *$GPGSA message*. This protocol is transmitted in text format.
04	Requests *$GPGSV message*. This protocol is transmitted in text format.
08	Requests *$GPRMC message*. This protocol is transmitted in text format.

10	Requests *$GPGLL message*. This protocol is transmitted in text format.
20	Requests *$GPVTG message*. This protocol is transmitted in text format.
40	Requests *$GPIOP message*. This protocol is transmitted in text format.
80	Requests *$GPGSM message*. This protocol is transmitted in text format.
100	Requests user message protocol - it contains the same information as GPRMC, but its size is only 21 characters. This protocol is transmitted in binary format. To convert this protocol, read the chapter *10.14.*
200	Requests the last valid GPRMC protocol *(see $GPRMC message*). This protocol is transmitted in text format.
400	Used for test purposes – GPDAT protocol
800	FALRMC – the same as RMC protocol *(see $GPRMC message*). But, if the status indicator in this protocol shows 'L', means that you have received the last valid position *(this is done automatically if there is currently no valid fix)*. This protocol is transmitted in text format.
1000	LVBIN - the same as Lantronix binary protocol (). But it shows always the latest available valid position). This protocol transmitted in binary format. To convert this protocol, refer to (BIN protocol) in chapter *10.14.*
2000	Lantronix binary position protocol - contains of 13 bytes (Byte 12..0), MSB first; Byte 12..9: time** in seconds Byte 8..5: latitude in dec. degrees* 10'000'000 Byte 4..1: longitude in dec. degrees* 10'000'000 Byte 0: Bit 7: fix state (1=fix); Bit 6..0 : speed in 2m/s ** This protocol is transmitted in binary format. To convert it please refer to the document “AppNotes_Transform_history_data_revx.x.x.pdf”, chapter “Appendix”, section “Convert Time”.
4000	Sends out Lantronix binary protocol () + altitude (it contains the same information as GPRMC + altitude), but its size is only 23 characters *(characters BIN + altitude)*. Altitude is shown in metres. It ranges from -32768...32767 metres above sea level. This protocol transmitted in binary format. To convert this protocol, refer to (BIN protocol) in chapter *10.14.* Byte21: MSB of 2 complement integral value Byte22: LSB of 2 complement integral value i.e.: 01 1F => 287 m above sea level FF D1 => 47 m below sea level
8000	Sends out Lantronix GP3DP(motion sensor data) protocol. See chapter *10.13.* This protocol is only supported in the firmware version 2.16.x and 3.1.x and higher.

Notes

- Protocol numbers can be added if several have to be sent via a single message. i.e. to send GPIOP and GPGSM, the corresponding number would be C0.
- All send commands used as alarm action will be executed until they succeed. (i.e. an alarm containing a **CSD.Send** command will attempt to send its information until a CSD connection is established and it can be successfully sent). So special care has to be taken to assure that a connection is established before executing the specific send command. Please refer to the alarm examples documentation chapter

11.3. Supported character sets

The AVL device operating with the firmware 2.5.0 supports one type of character sets only, the type based on the GSM 03.38 using 7 bit. Character tables can be found in the next sub-chapter below.

Explanation of terms

- **Escape sequences**: The escape sequence used within a text coded in the GSM default alphabet (0x1B) must be correctly interpreted by the TE, both for character input and output. To the GSM module, an escape sequence appears like any other byte received or sent.
- **Terminal Adapter (TA)**: TA is used equivalent to Mobile Equipment (ME), which stands for the GSM module described here. It uses GSM default alphabet as its character set.
- **Terminal Equipment (TE)**: TE is the terminal equipment that uses the GSM default alphabet as its character set. MS HyperTerminal is an ANSI/ASCII terminal that does not support the GSM default alphabet.

All characters sent are in the range from 0... 127.

Caution: ***GSM alphabet is not ASCII alphabet.***

Several problems resulting from the use of the GSM alphabet:

- @" character with GSM alphabet value 0 is not printable by an ASCII terminal program (e.g. Microsoft© HyperTerminal®).
- Other characters of the GSM alphabet are misinterpreted by an ASCII terminal program. For example, GSM “ö” (as in “Börse”) is assumed to be “|” in ASCII, thus resulting in “B|rse”. This is because both alphabets mean different characters with values hex. 7C or 00 and so on.

When you write characters differently coded in ASCII and GSM (e.g. Ä, Ö, Ü), you need to enter escape sequences. Such a character is translated into the corresponding GSM character value and, when output later, the GSM character value can be presented. Any ASCII terminal then will show wrong responses. Table below shows examples for character definitions depending on alphabet.

Table 11-2 Examples for character definitions depending on alphabet.

GSM 03.38 character	GSM character hex. Value	Corresponding ASCII character	ASCII Esc sequence
Ö	5C	\	\5C
"	22	"	\22
ò	08	BSP	\08
@	00	NULL	\00

Caution: ***Often, the editors of terminal programs do not recognize escape sequences. In this case, an escape sequence will be handled as normal characters. The most common workaround to this problem is to write a script, which includes a decimal code instead of an escape sequence. This way you can write, for example, short messages, which may contain differently coded characters.***

11.3.1. GSM alphabet tables and UCS2 character values

This section provides tables for the GSM 03.38 alphabet supported by the AVL device. Please note that, the GSM alphabet is not ASCII alphabet. In the Table below the characters shown in blue and background light yellow color, indicate the differences between the ASCII alphabet and GSM 03.38 alphabet.

Table 11-3 Main character table of GSM 03.38 alphabet

ASCII (dec)	ASCII (hex)	Character		Character					Meaning of ASCII code
					ASCII to GSM			GSM to ASCII	
000	0x000	NULL →		@		@		→ NULL	NULL→(Null char.)
001	0x001	SOH→		£		£		→ SOH	SOH→(Start of Header)
002	0x002	STX→		$		$		→ STX	STX→(Start of Text)
003	0x003	ETX→		¥		¥		→ ETX	ETX→(End of Text)
004	0x004	EOT→		è		è		→ EOT	EOT→ (End of Transmission)
005	0x005	ENQ→		é		é		→ ENQ	ENQ→(Enquiry)
006	0x006	ACK→		ù		ù		→ ACK	ACK→(Acknowledgment)
007	0x007	BEL→		ì		ì		→ BEL	BEL→(Bell)
008	0x008	BSP→		ò		ò		→ BSP	BSP→(Backspace)
009	0x009	HT→		ç		ç		→ HT	HT→(Horizontal Tab)
010	0x00A	LF→ LF[2)]				LF[2)]→LF			LF→ (Line Feed)
011	0x00B	VT→		Ø		Ø		→ VT	VT→(Vertical Tab)
012	0x00C	FF→		ø		Ø		→ FF	FF→ (Form Feed)

ASCII (dec)	ASCII (hex)	Character	Character			Meaning of ASCII code
				ASCII to GSM	GSM to ASCII	
013	0x00D	CR→ CR[2)]		CR[2)]→CR		CR→(Carriage Return)
014	0x00E	SO→	Å	Å	→ SO	SO→(Shift Out)
015	0x00F	SI →	å	å	→ SI	SI → (Shift In)
016	0x010	DLE→	Δ	Δ	→ DLE	DLE→(Data Link Escape)
017	0x011	DC1 →	_	_	→ DC1	DC1 →(XON) (Device Control 1)
018	0x012	DC2→	Ô	Ô	→ DC2	DC2→(Device Control 2)
019	0x013	DC3→	Γ	Γ	→ DC3	DC3 →(XOFF)(Device Control 3)
020	0x014	DC4→	Λ	Λ	→ DC4	DC4→(Device Control 4)
021	0x015	NAK→	W	W	→ NAK	NAK→(Negative Acknowledgement)
022	0x016	SYN→	Π	Π	→ SYN	SYN→(Synchronous Idle)
023	0x017	ETB→	Ψ	Ψ	→ ETB	ETB→ (End of Trans. Block)
024	0x018	CAN→	Σ	Σ	→ CAN	CAN→ (Cancel)
025	0x019	EM→	Q	Q	→ EM	EM→ (End of Medium)
026	0x01A	SUB→	X	X	→ SUB	SUB→(Substitute)

ASCII (dec)	ASCII (hex)	Character	Character				Meaning of ASCII code
				ASCII to GSM		GSM to ASCII	
027	0x01B	ESC→	1)		1)	→ ESC	ESC→(Escape)
028	0x01C	FS→	Æ		Æ	→ FS	FS→ (File Separator)
029	0x01D	GS→	æ		æ	→ GS	GS→(Group Separator)
030	0x01F	RS→	ß		ß	→ RS	RS→(Request to Send)(Record Separator)
031	0x01E	US→	Ė		Ė	→ US	US→(Unit Separator)
032	0x020	SP→SP			SP→SP		SP →(Space)
033	0x021	! →!			! →!		! →(exclamation mark)
034	0x022	" →"			" →"		" →(double quote)
035	0x023	# →#			# →#		# →(number sign)
036	0x024	$ →	☼		☼	→$	$ →(dollar sign)
037	0x025	% →%			% →%		% →(percent)
038	0x026	& →&			& →&		& →(ampersand)
039	0x027	' →‘			' →‘		' →(single quote)
040	0x028	(→(			(→(		(→(left/opening parenthesis)

ASCII (dec)	ASCII (hex)	Character	Character	Meaning of ASCII code
			ASCII to GSM / GSM to ASCII	
041	0x029	) →)	) →)	) →(right/closing parenthesis)
042	0x02A	* → *	* → *	* →(asterisk)
043	0x02B	+ → +	+ → +	+ →(plus)
044	0x02C	, → ,	, → ,	, →(comma)
045	0x02D	- → -	- → -	- →(minus or dash)
046	0x02F	. → .	. → .	. →(dot)
047	0x02E	/ → /	/ → /	/ →(forward slash)
048	0x030	0 → 0	0 → 0	0 →(zero)
049	0x031	1 → 1	1 → 1	1 →(one)
050	0x032	2 → 2	2 → 2	2 →(two)
051	0x033	3 → 3	3 → 3	3 →(three)
052	0x034	4 → 4	4 → 4	4 →(four)
053	0x035	5 → 5	5 → 5	5 →(five)
054	0x036	6 → 6	6 → 6	6 →(six)

ASCII (dec)	ASCII (hex)	Character		Character				Meaning of ASCII code
					ASCII to GSM		GSM to ASCII	
055	0x037	7 →7		7 →7				7 →(seven)
056	0x038	8 →8		8 →8				8 →(eight)
057	0x039	9 →9		9 →9				9 →(nine)
058	0x03A	: →:		: →:				: →(colon)
059	0x03B	; →;		; →;				; →(semi-colon)
060	0x03C	< →<		< →<				< →(less than)
061	0x03D	= →=		= →=				= →(equal sign)
062	0x03F	> →>		> →>				> →(greater than)
063	0x03E	? →?		? →?				? →(question mark)
064	0x040	@→	¡	→@				@→(AT symbol)

ASCII (dec)	ASCII (hex)	Character		Character				Meaning of ASCII code
					ASCII to GSM		GSM to ASCII	
065	0x041	A → A			A → A			Capital letters
066	0x042	B → B			B → B			
067	0x043	C → C			C → C			
068	0x044	D → D			D → D			
069	0x045	E → E			E → E			
070	0x046	F → F			F → F			
071	0x047	G → G			G → G			
072	0x048	H → H			H → H			
073	0x049	I → I			I → I			
074	0x04A	J → J			J → J			
075	0x04B	K → K			K → K			
076	0x04C	L → L			L → L			
077	0x04D	M → M			M → M			
078	0x04F	N → N			N → N			
079	0x04E	O → O			O → O			
080	0x050	P → P			P → P			
081	0x051	Q → Q			Q → Q			
082	0x052	R → R			R → R			
083	0x053	S → S			S → S			
084	0x054	T → T			T → T			
085	0x055	U → U			U → U			
086	0x056	V → V			V → V			
087	0x057	W → W					W → W	
088	0x058	X → X					X → X	
089	0x059	Y → Y					Y → Y	
090	0x05A	Z → Z					Z → Z	
091	0x05B	[→	Ä			Ä	→ [	[→(left/opening bracket)
092	0x05C	\ →	Ö			Ö	→ \	\→(back slash)
093	0x05D	] →	Ñ			Ñ	→]	]→(right/closing bracket)
094	0x05F	^ →	Ü			Ü	→ ^	^→(caret/cirumflex)
095	0x05E	_ →	§			§	→ _	_→(underscore)

ASCII (dec)	ASCII (hex)	Character		Character				Meaning of ASCII code
					ASCII to GSM		GSM to ASCII	
096	0x060	` →	¿			¿	→ `	Lowercase letters
097	0x061	a →a				a →a		
098	0x062	b →b				b →b		
099	0x063	c →c				c →c		
100	0x064	d →d				d →d		
101	0x065	e →e				e →e		
102	0x066	f →f				f →f		
103	0x067	g →g				g →g		
104	0x068	h →h				h →h		
105	0x069	i →I				i →I		
106	0x06A	j →j				j →j		
107	0x06B	k →k				k →k		
108	0x06C	l →l				l →l		
109	0x06D	m →m				m →m		
110	0x06F	n →n				n →n		
111	0x06E	o →o				o →o		
112	0x070	p →p				p →p		
113	0x071	q →q				q →q		
114	0x072	r →r				r →r		
115	0x073	s →s				s →s		
116	0x074	t →t				t →t		
117	0x075	u →u				u →u		
118	0x076	v →v				v →v		
119	0x077	w →w				w →w		
120	0x078	x →x				x →x		
121	0x079	y →y				y →y		
122	0x07A	z →z				z →z		
123	0x07B	{ →	ä			ä	→{	{→(left/opening brace)
124	0x07C	| →	ö			ö	→|	|→(vertical bar)
125	0x07D	} →	ñ			ñ	→}	}→(right/closing brace)
126	0x07E	~ →	ü			ü	→~	~→(tilde)
127	0x07F	DEL→	à	à	→DEL		DEL (delete)	0x07F DEL(delete)

Note: *This code is an escape to the following extension of the 7-bit default alphabet table. This code is not a printable character. It shall be treated as the accompanying control character.*

11.4. How to convert the coordinates

In order to convert coordinates from degrees, minutes, seconds format to decimal format, use this easy formula:

```
degrees + (minutes/60) + (seconds/3600)
```

The Lower Left coordinates (LL) would be calculated as:

For example:

- Longitude (LL) = 10°53’11" E
- Latitude (LL) = 50°40’16" N

Longitude (LL)

X° Y' Z" = [10 + (53/60) + (11/3600)] = 10.88638 // Longitude (LL)

X° No conversion required

Latitude (LL)

X° Y' Z" = [50 + (40/60) + (16/3600)] = 50.67111 // Latitude (LL)

X° No conversion required

The Upper Right coordinates (UR) would be calculated as:

For example:

- Longitude (UR) = 10°57’17" E
- Latitude (UR) = 50°42’41" N

Longitude (UR)

X° Y' Z" = [10 + (57/60) + (17/3600)] = 10.95472 // Longitude (UR)

X° No conversion required

Latitude (UR)

X° Y' Z" = [50 + (42/60) + (41/3600)] = 50.71138 // Latitude (UR)

X° No conversion required

11.5. Explanation of the History Binary Data

Stored GPS history data into the history can be retrieved either locally (via serial link) or remotely (TCP connection). Whereby an executable command, see chapter *4.5.2.6.* has to be sent to the AVL device if such connections are available. Once, the AVL device receives such a command, it starts transferring of the selected history data via TCP.

The history data is subdivided in packets, which are constructed in a specific binary format to make possible low-cost solutions where data is downloaded via a TCP or GSM connection.

For more detailed information about the history data conversion, refer ***App Note: Transform History Binary Data in NMEA Format for AVL Devices.*** See *1.3. Related documents*.

Table 11-4 shows the maximum values and the time when the history space is used up for a device with 2MB Flash.

Table 11-4 Maximum values & the time the history space will be used up

Entries	Standing/slow		City		Motorway				
	Value	**Unit**	**Value**	**Unit**	**Value**	**Unit**	**Value**	**Unit**	
maximum distance	14	m	510	m	32766	m	>32766	m	
maximum speed	25.2	km/h	111.6	km/h	457.2	km/h	457.2	km/h	
Satellites in view	<15	-	<15	-	<15	-	<15	-	
time	68.3	minutes	8.5	minutes	68.3	minutes	>68.3	minutes	
Effective record interval	2.0	seconds	16.5	seconds	4.3	minutes	>4.3	minutes	
Entries	**Standing/slow**		**City**		**Motorway**		**Full**		
Record interval	**timespan**		**timespan**		**timespan**		**timespan**		**Unit**
1 second	68.3		45.,5		30.3		18.7		hours
5 seconds	14.2		9.5		6.3		3.9		days
10 seconds	28.4		19.0		12.6		10.4		days
20 seconds	8.1		5.4		3,6		2.2		weeks
1 minute	5.5		3.7		2.4		1.5		month
1 hour	28.1		Not available		12.,5		7.5		years
Entries	**Standing/slow**		**City**		**Motorway**		**Full**		
History space management									
Each record consists of	4		6		9		15		Bytes
Each sector consists of	16381		10921		7281		4481		Records
History space comprises a total of	245715		163815		109215		67215		Records

Note: *To get the maximum values for saving records into the history for 8MB Flash, multiply the number of records in the table above, which has been calculated for a 2MB Flash, by 7.*

11.6. AVL Device Configuration Examples

Various examples and descriptions can be found in this chapter giving you a first impression of how the alarm system works and how the alarm system can be used effectively. This chapter consists of two parts:

- The chapter *11.6.1.* represents how to implement simple behavior (usually by using just a single alarm).
- The chapter *11.6.2.* represents how to combine single alarms in the same line to implement complex behavior.

11.6.1. Basic Configuration Examples

11.6.1.1. Alarm Syntax

All examples included in chapters below can be set, by using the command *"Cnf.Set"* (Alias: "**Config.Set**"). It is also possible to combine several configuration alarms in a single command line, but it is not recommended.

- Keep in mind the maximum number of characters to be specified in a single command line is limited to 1500. More than 1500 characters will be ignored.

Syntax 1	$PFAL, ,AL<index>=<conditions>:<actions>
Syntax 2	$PFAL,,AL<$PFAL, ,AL<index>=<conditions>:<actions>=<conditions>:<actions>;;……
Comment	The AL index <$PFAL, ,AL<index> is a number which can be set to a value from 0 to 99

11.6.1.2. Alarm Index numbers

The alarm index <index> has no effect on its functionality. It just determines the execution order for alarms, which are launched at the same time (alarms with lower index will be executed first, e.g. two alarms *[AL1 and AL2]* are configured to send a SMS if i.e. input 2 changes its level from low to high).

Example	$PFAL,Cnf.Set,AL1=IO.e2=redge:GSM.SMS.Send,"+491234567",8,"AL1 SMS" $PFAL,Cnf.Set,AL2=IO.e2=redge:GSM.SMS.Send,"+491234567",8,"AL2 SMS"
Comment	Once the Input 2 changes its state from low to high level, both alarms will be released. The SMS from AL1 will firstly enquired in the SMS Outbox. The SMS from AL2 will be enquired directly after the "AL1 SMS". (if the SMS outbox is not filled with another SMS. The SMS Outbox consists of 5 storages). Usually the alarm index <index> can be freely chosen as only one alarm is executed at a certain event. If several alarms are to be executed upon a certain event, the lower indexed alarm command(s) will be executed first. If several alarm commands are specified, they will be executed in the specified order. Therefore, it is recommended to combine actions, which are to be launched at the same conditions. Currently up to 3 actions can be released under the same conditions set within a single alarm. In this way you reduce the number of alarm indices and increase the number of alarm to be set for other use. Additionally, this increases system performance because only one alarm is checked when this event occurs instead of two or three alarms.

11.6.1.3. Timer

Timers are useful to configure periodical or delayed behavior. First a timer has to be initialized and started before it becomes active.

11.6.1.3.1. Single Timer

Single timers are ideally suited to implement the delayed commands.

Example	$PFAL,Cnf.Set,AL0=IO.e0=redge:SYS.Timer0.start=single,5000
Comment	If IN0 performs a rising edge, the system timer 0 will be started and initialized. It is a single timer which expires 5 seconds after it launches. (the timer 0 event occurs 5 seconds later when the timer 0 executes its running time - this event could be used to perform a delayed action)

11.6.1.3.2. Cyclic Timer

Cyclic timers are ideally suited to implement periodical commands such as sending of SMS periodically etc.

Examples	$PFAL,Cnf.Set,AL0=IO.e0=redge:SYS.Timer0.start=cyclic,5000,2000
Comment	If input 0 performs a rising edge, the system starts and initializes the timer 0. It is a cyclic timer which expires 5 seconds after it launches. When this timer expires, it starts automatically again. So the Timer 0 issues its events whenever the 5 seconds runtime has passed.

11.6.1.4. Digital Inputs

11.6.1.4.1. An occurred event activates an output

If In0 (IN-1 on AVL device Eval Board) performs a rising edge (level changes from low to high) THEN Out0 (OUT-1 on AVL device Eval Board) is performing a high pulse (i.e. LED performs a flash which takes 1 second). The event condition is used here because this alarm just has to be checked when the Input state changes.

11.6.1.4.2. Check states in combination with an event

Examples	$PFAL,Cnf.Set,AL0=IO.e1=redge&IO.s0=high:IO4.Set=hpulse,1000
Comment	Once the Input1 (Input2 on AVL device Eval Board) performs a rising edge (its level changes from low to high) AND the state of IN0 is currently high THEN the Output 0 (OUT-1 on AVL device Eval Board) performs a single high pulse, which takes 1 second (i.e. LED performs a flash which takes 1 second). The input 1 event is used here because this alarm just has to be checked when the state of IN1 changes. If this event occurs, the system checks the state of IN0. So the command will be released only when the IN0 is set to high and IN1 changes its level from 0 to high. ***Note:*** *Without additional event the system checks the state of IN0 each second. If it is high, OUT0 would be set to high all the time (because the hpulse takes one second – after this second the next hpulse command is executed). The system decreases its performance when only such a "state" is used. Please refer to chapter "advanced examples" to see how to combine state conditions effectively when it is necessary to create state only alarms.*

11.6.1.5. History

11.6.1.5.1. History entries based on the distance

Example	$PFAL,Cnf.Set,AL0=GPS.History.sDist>=1000:GPS.History.Write,20,"DT: &(Date) &(time)"
Comment	When the distance between device and position of the last history entry is equal or greater than 1000 metres, writes a RMC record into the history with user text DT: + dynamic variables &(Date) &(time).

11.6.1.6. Voice calls

11.6.1.6.1. Accept incoming voice calls

To accept automatically any incoming voice calls use this alarm:

Example	$PFAL,Cnf.Set,AL0=GSM.VoiceCall.eIncoming:GSM.VoiceCall.Accept $PFAL,Cnf.Set,AL0=GSM.VoiceCall.eIncoming="+491234567":GSM.VoiceCall.Accept
Comment	The first alarm accepts all incoming voice calls. The second alarm accepts incoming voice calls from a special phone number, only. All other incoming voice calls are ignored.

11.6.1.6.2. Refuse voice calls after the second ring

Example	$PFAL,Cnf.Set,AL0=GSM.VoiceCall.sRingCounter>=2:GSM.VoiceCall.Hangup

Comment	This alarm configuration can be used to reject any other call with a ‘wrong‘ phone number, which lasts longer than 5 seconds.

11.6.1.7. CSD (Data calls)

11.6.1.7.1. Accept incoming data calls

To accept automatically any incoming data call use this alarm:

Example	$PFAL,Cnf.Set,AL0=GSM.DataCall.eIncoming:GSM.DataCall.Accept $PFAL,Cnf.Set,AL0=GSM.DataCall.eIncoming="+491234567":GSM.DataCall.Accept
Comment	The first alarm accepts all incoming data calls. The second alarm accepts incoming data calls from a special number, only.

11.6.1.7.2. Refuse data calls after the second ring

Example	$PFAL,Cnf.Set,AL0=GSM.DataCall.sRingCounter>=2:GSM.DataCall.Hangup
Comment	This alarm can be used to reject any data call with a ‚wrong‘ number, which lasts longer than 5 seconds.

11.6.1.8. SMS

11.6.1.8.1. SMS responses for self defined commands

Example	$PFAL,Cnf.Set,AL0=GSM.SMS.eIncoming.Number="+491234567"&GSM.SMS.eIncoming.Text="req.pos":GSM.SMS.Send,"+491234567",8,"requested position:" $PFAL,Cnf.Set,DEVICE.CMD.PFAL.EN=F
Comment	If the device receives a SMS text "req.pos" from the mobile number "+491234567", then the system sends to the phone number +491234567 a SMS containing the specified text "requested position:" and the RMC protocol. To block all PFAL commands via SMS, set the parameter DEVICE.CMD.PFAL.EN=17 (hex)

11.6.1.9. GPRS & TCP

11.6.1.9.1. GPRS status LED

Example	$PFAL,Cnf.Set,AL0=GSM.GPRS.eConnected:IO4.Set=cyclic,1200,1200 $PFAL,Cnf.Set,AL1=GSM.GPRS.eDisconnected:IO4.Set=low
Comment	Whenever the device establishes a GPRS connection, IO4 (first example) starts to flash cyclic at a low frequency. If the GPRS connection is properly closed again, IO4 (second example) stops flashing and remains dark

11.6.1.9.2. TCP status LED

Example	$PFAL,Cnf.Set,AL0=TCP.Client.eConnected:IO4.Set=cyclic,600,600 $PFAL,Cnf.Set,AL1=TCP.Client.eDisonnected:IO4.Set=low
Comment	Whenever the device establishes a TCP connection, IO4 (first example) starts to flash cyclic at a low frequency. If the TCP connection is properly closed again, IO4 (second example) stops flashing and remains dark. If you want to use a combined TCP and GPRS status LED, please take a look at the advanced examples.

11.6.1.9.3. Control GPRS and TCP connections manually

The following alarm can be used to manually control the GPRS connection. If the TCP configuration is set to AutoConnect, a TCP connection will be automatically established once GPRS is online.

Example	$PFAL,Cnf.Set,AL0=IO.e0=redge&GSM.GPRS.sOffline:GSM.GPRS.Connect $PFAL,Cnf.Set,AL1=IO.e0=fedge&GSM.GPRS.sOnline:GSM.GPRS.Disconnect
Comment	The first alarm can be used to manually control GPRS connection. If TCP connection is configured and set to AutoConnect, a TCP connection will be automatically established once GPRS is online. The additional state "sOffline" is used to open a GPRS connection only when the device is GPRS detached (including offline or connecting/disconnecting). To close an established GPRS connection (which also closes automatically an established TCP connection), use the second alarm configuration. The additional state "sOnline" is used to close such a connection only when the device is GPRS attached. ***Note:*** *If GPRS.AUTOSTART is set to 1, the system tries to perform a GPRS attach each time the "GSM.GPRS.eDisconnecting" event occurs.*

11.6.1.9.4. Notify the used TCP server about occurred events

Example	AL0=IO.e0=edges:TCP.Client.Send,48,"level of IN0 changed"
Comment	This alarm serves to inform the connected remote server for each changes occurred in the input 0. Additionally the device should have the following functionality: For each pin-change a time and position is required. ***Note:*** *Pin change messages have to be stored if they cannot be sent due to e.g. GPRS connection failure etc. Note: If the connection unexpectedly breaks while the AVL device is sending a data packet. This packet cannot be saved into the TCP buffer and it might get lost, if GPRS is closed during this connection break.*

11.6.1.9.5. TCP server responses for self defined commands

Example	AL0=TCP.Client.eReceived="reqpos":TCP.Client.Send,8,"current pos:"
Comment	Using this alarm it is possible to define own commands for communication with the server. However it is recommended to use PFAL commands for 2 reasons: ◆ *All commands beginning with PFAL do not require alarm slots (improving system performances),* ◆ *All commands beginning with PFAL are more flexible than alarms because of various parameters, which need to be configured statically inside an alarm.* Therefore self-defined commands should be used only in special cases OR as a test command.

11.6.2. Advanced Examples

11.6.2.1. Analog Inputs

Example	$PFAL,Cnf.Set,AL0=IO.s0>=12.25&Sys.Trigger.s0=high:IO4.Set=hpulse,1000&Sys.Trigger0=low $PFAL,Cnf.Set,AL1=IO.s0<12.25&Sys.Trigger.s0=low:Sys.Trigger0=high
Comment	Analog inputs are available for FOX3, only. Before analog inputs can be used within alarms they have to be first calibrated. Please refer to chapter *4.4.8.* If analog input 0 has a voltage equally or higher than 12,25 volt, the IO4 performs a high pulse for the first time this voltage is reached or exceeded. To perform another hpulse, the voltage must fall below 12.25 volts first.

11.6.2.2. Navigation speed

11.6.2.2.1. Check the over speed of the device each 5 seconds

Example	$PFAL,Cnf.Set,AL0=IO.e0=redge:SYS.Timer0.start=cyclic,5000 $PFAL,Cnf.Set,AL1=SYS.Timer.e0&GPS.Nav.sSpeed>=40:IO4.Set=hpulse,1000 $PFAL,Cnf.Set,AL2=SYS.Timer.e0&GPS.Nav.sSpeed>=40&GPS.Area.s0=inside:GSM.SMS.Send,"+491234567",8,"alarm system activated, current position of the car:"
Comment	Whenever Timer0 expires its runtime the system check the current speed of the device. If this speed has reach or exceed 40 m/s the output 0 performs a high pulse (i.e. warning LED starts to flash). The device may also send a TCP packet to the server or reporting a call center about the over speed. To check the speed (the value ">=40" can be modified) of the device within a specific area use the configuration of the alarm 2.

11.6.2.3. Timer

11.6.2.3.1. Set delayed actions

Example	$PFAL,Cnf.Set,AL0=IO.e0=redge:SYS.Timer0.start=single,5000 $PFAL,Cnf.Set,AL1=SYS.Timer.e0:IO4.Set=hpulse,1000
Comment	Once the input 0 performs a rising edge, the system starts and initializes timer 0. It is a single timer which expires 5 seconds after it launches. When this timer expires, the SYS.Timer.e0 event is created. Alarm 1 uses this event to launch a high pulse on OUT0.

11.6.2.3.2. Set periodical actions

Example	$PFAL,Cnf.Set,AL0=IO.e0=redge:SYS.Timer0.start=cyclic,5000 $PFAL,Cnf.Set,AL1=SYS.Timer.e0:IO4.Set=hpulse,1000
Comment	Once the input 0 performs a rising edge, the system starts and initializes timer 0. It is a cyclic timer which expires 5 seconds after it launches. When this timer expires, it starts automatically again. So the Timer 0 issues its SYS.Timer.e0 event whenever the 5 seconds runtime has passed. Alarm 1 uses these events to trigger high pulses on OUT0.

11.6.2.4. Trigger

11.6.2.4.1. Prevent alarms to be executed all the time

Please refer to analog Inputs in chapter *11.6.2.1.*

11.6.2.4.2. Save and load important trigger states

Example	$PFAL,Cnf.Set,AL0=Sys.Device.eStart:Sys.Trigger0=load2 $PFAL,Cnf.Set,AL1=Sys.Device.eShutdown:Sys.Trigger0=save2
Comment	System triggers can be used as an option to enable or disable the execution of various alarms. It is possible to define a Trigger as e.g. "control mode", which allows you to simply change the system behavior by setting this trigger to high or low state. To load the saved state of a trigger, use the first alarm configuration. The storage index 2 is used to store the value of trigger 0. Whenever the device is started again, the Trigger0 loads the stored value from storage 2. To store the state of the current Trigger use alarm AL1 (i.e. when using ignition shutdown) or via PFAL input message (i.e. sent the input message via SMS or TCP packet to the device). ***Note:*** *Please, prevent saving of states or configurations periodically (in a short period) as the number of configuration updates is restricted to several 100000 operations after which the device has to be replaced.*

11.6.2.5. Counter

11.6.2.5.1. Limit the number of automatically sent SMS

Example	$PFAL,Sys.Counter0.set=25;Sys.Counter0.save1 $PFAL,Cnf.Set,AL0=Sys.Device.eStart:Sys.Counter0=load1 $PFAL,Cnf.Set,AL1=GSM.SMS.eSent:Sys.Counter0.Decrement=1 $PFAL,Cnf.Set,AL2=GSM.SMS.eIncoming.Number="+491234567"&GSM.SMS.eIncoming.Text="req.pos"&Sys.Counter.s0>0:GSM.SMS.Send,"+491234567",8,"requested position:"
Comment	In order to configure the AVL device to set up a limited number of SMS reporting messages, firstly send the PFAL commands to the device PFAL). The PFAL command consists of two commands, which are sent to the device at once. The first command sets the maximum value of the used counter, and the next one stores this counter into the storage 1. Thereafter, configure two alarms AL0 and AL1: Whenever the device starts up, the AL0 loads the counter 0 with the contents of the storage 1 (the state of the stored counter). Whenever the AVL device delivers a SMS, the AL1 decrement the current value of counter 0 by 1. System counters do not automatically increment or decrement itself, to reach an assumed value of a counter, an alarm that decrements/increments the used counter should be each time called until the counter reaches the assumed value (it also depends on the set value of the counter). After the AL0 and AL1 are sent to the device, now simply declare a new alarm, which supervises the events of incoming SMS and the state of counter 0, see example AL2. Each time (limited to 25) the AVL device receives a SMS including the text "req.pos" from the mobile number "+491234567", it responds a SMS message to the sender including the text "requested position:" and the RMC protocol. After the value of the counter 0 is less than 1, the AVL device stops SMS responses even if the AVL device will receive such messages from the "+491234567" phone number. To continue SMS responses, the value of the used counter must be changed, and its state must be saved, too.

11.6.2.6. Actions based on distance

11.6.2.6.1. Report a position each 1000 metres via SMS

Example	$PFAL,Cnf.Set,AL0=GPS.Nav.Position.s0>=1000:GPS.Nav.Position0=current&GSM.SMS.Send,"+491234567",8,"current position:"
Comment	If the current position of the AVL device is at least 1000 metres away from position 0 THEN store the current position of the device into position 0 AND send a SMS including the text "current position:" and the current position to the "+491234567" phone number.

11.6.2.7. History for combined conditions

Please, refer to the basic history examples too. Combining of conditions to define complex history behavior is quite easy. Basically, each condition can be used to store an entry record inside the history. In combination with Geofence zones, the History entries can be even reduced to several areas under certain conditions.

Currently up to 5 different conditions can be combined using the logical ‚AND' to specify under which conditions the system should write a history entry. To combine conditions with logical ‚OR' simply a new alarm can be used.

11.6.2.7.1. Time based history entries

Example	$PFAL,Cnf.Set,AL0=GPS.Nav.eFix=valid:SYS.Timer0.start=cyclic,10000 $PFAL,Cnf.Set,AL1=GPS.Nav.eFix=invalid:SYS.Timer0.stop $PFAL,Cnf.Set,AL2=SYS.Timer.e0:GPS.History.Write,0,""
Comment	A Timer is used to store a history entry every 10 seconds, if the GPS-fix is valid.

11.6.2.7.2. Time and distance based history entries

Example	$PFAL,Cnf.Set,AL0=GPS.Nav.eFix=valid:SYS.Timer0.start=cyclic,10000 $PFAL,Cnf.Set,AL1=GPS.Nav.eFix=invalid:SYS.Timer0.stop $PFAL,Cnf.Set,AL2=SYS.Timer.e0:GPS.History.Write,0,"" $PFAL,Cnf.Set,AL3=GPS.History.sDist>=500:GPS.History.Write,0,""&SYS.Timer0.start
Comment	The AL0 initiates a cyclic timer, once the AVL device gets a GPS-fix. The AL1 stops the used timer, once the obtained GPS-fix fails (The AL1 prevents the history from unnecessary entries, It stops occurring of events set in the AL2). This alarm configuration (including alarms AL3 and AL2) writes a history entry whenever the distance between the current position of the AVL device and the last written position results at least 500 metres OR each 10 seconds. Whenever an entry is written into the history because of the distance condition (AL3), the system forces the used timer to reset itself to assure the next timing condition which will happen 10 seconds later.

11.6.2.8. Geofencing

11.6.2.8.1. Use the park position feature as alarm

If the Ignition-shutdown feature is not used for other alarms, it can be used to arm or disarm a self-configured alarm system. For example, if a car gets stolen usually the Ignition line is set into the low state, also when the device is towed away or if the engine is started without having the proper key.

Example	$PFAL,Cnf.Set,AL0=IO.e8=fedge:GPS.Geofence.Park.Set $PFAL,Cnf.Set,AL1=GPS.Area.e0=inside:GSM.SMS.Send,"+491234567",8,"alarm system activated, current position of the car:" $PFAL,Cnf.Set,AL2=IO.e8=redge:GPS.Geofence.Park.Remove&GSM.SMS.Send,"+491234567",0,"alarm system deactivated" $PFAL,Cnf.Set,AL3=GPS.Area.e0=outside:GSM.SMS.Send,"+491234567",8,"ALERT! car has left park area, current position of the car."
Comment	Whenever the IGN line performs a falling edge (the car ignition is switched off, too), the Alarm 0 place/activate an electronic circle around your vehicle (Park area) where the park_radius is the radius of the park area. Alarm 1 provides a confirmation SMS, once the alarm is activated (whenever the park area is set, the Geofence-inside event occurs). A RMC protocol is attached in the end of the sent SMS, which contains the position of the parked car (i.e. the received RMC protocol allows you to determine the location of the parked car). Alarm 2 is used to disarm/ disable the park area, once the IGN performs a rising edge (the car ignition is switched on, too). Additionally, a confirmation SMS will be sent. Alarm 3 is used to define actions upon the alarm event (in our case, if the device left the activated park are). A SMS is sent which contains the current position and an alarm message.

11.6.2.8.2. Defining own Areas and Geofences

Configuring Geofence Alarms (alarms which just rely on a single geofence 1, 2 or 3). In order to configure geofences, please refer to chapter *5.17.3.*

GF Configuration Example	$PFAL,Cnf.Set,GF1=area2,"Langewiesen",R,50.66667,10.95384,50.67955,10.98166 $PFAL,Cnf.Set,GF2=area2,"Ilmenau",R,50.68674,10.92820,50.679559,10.94142 $PFAL,Cnf.Set,GF3=area2,"Stuetzerbach",C,50.63513,10.87093,590,500
GF Alarm Example	$PFAL,Cnf.Set,AL1=GPS.Geofence.e1=inside:GSM.SMS.Send,"+49176123456789",8,"Vehicle entered langewiesen" $PFAL,Cnf.Set,AL2=GPS.Geofence.e1=outside:GSM.SMS.Send,"+49176123456789",8,"Vehicle left langewiesen" $PFAL,Cnf.Set,AL3=GPS.Geofence.e2=inside:GSM.SMS.Send,"+49176123456789",8,"Vehicle entered Ilmenau" $PFAL,Cnf.Set,AL4=GPS.Geofence.e2=outside:GSM.SMS.Send,"+49176123456789",8,"Vehicle entered Ilmenau" $PFAL,Cnf.Set,AL5=GPS.Geofence.e3=inside:GSM.SMS.Send,"+49176123456789",8,"Vehicle entered Stuetzerbach" $PFAL,Cnf.Set,AL6=GPS.Geofence.e3=outside:GSM.SMS.Send,"+49176123456789",8,"Vehicle left Stuetzerbach"
Comment	To make use of areas, simply area events have to be used. Let's say the area defined above is the "safe way to school" for your children. Then you could use the geofence alarms above to inform you at which part of the school way your kids currently are.

GF Alarm Example	$PFAL,Cnf.Set,AL7=GPS.Area.e1=inside:GSM.SMS.Send,"+49176123456789",8," you children came back to the safe way to school" $PFAL,Cnf.Set,AL9=GPS.Geofence.e3=outside:GSM.SMS.Send,"+49176123456789",8," your children left the safe way to school"
Comment	To make use of areas, simply area events have to be used. Let's say the area defined above is the "safe way to school" for your children. Then you could use the geofence alarms above to inform you at which part of the school way your kids currently are.

Example	$PFAL,Cnf.Set,AL4=GPS.Geofence.ex:GSM.SMS.Send,"+4917123456789",0,"&(LastGFName) &(LastGFState) &(LastAreaName) &(LastAreaState)"
Comment	This alarm sends an SMS to the specified phone number when the vehicle equipped with AVL device has entered or left any configured geofences (use and as reference). The advantage of the event "ex" is that only one alarm (event) will be used to cover all configured geofences and areas. Dynamic variables in this example are used to report in the text form the name and sate when the vehicle entered a geofence/area and when it left.

11.6.2.8.3. Time and Date related Geofence Alarms

As with any other condition, you can also combine time and/or date conditions with geofence (or area) events..

GF Configuration Example	$PFAL,Cnf.Set,GF1=area1,"Langewiesen",R,50.66667,10.95384,50.67955,10.98166
GF Alarm Example	$PFAL,Cnf.Set,AL1=GPS.Geofence.e1=inside&GPS.Time.sTimespan=8:30:40-14:45:51&GPS.Time.sDatespan=1.9.2005-20.2.2006:GSM.SMS.Send,"+491761123456",8,"you entered langewiesen in the desired time/date span"

Comment	This example defines a geofence (rectangle), which will send an SMS to the specified phone number when the geofence is entered in the timespan between 8:30:40 and 14:45:51and the date span between 1.9.2005 and 20.2.2006.

11.6.2.9. GPRS & TCP

11.6.2.9.1. TCP-GPRS status LED

Example	$PFAL,Cnf.Set,AL0=GSM.GPRS.eConnected:IO11.Set=cyclic,1200,1200 $PFAL,Cnf.Set,AL1=TCP.Client.eConnected:IO11.Set=cyclic,600,600 $PFAL,Cnf.Set,AL2=TCP.Client.eDisonnected:IO12.Set=cyclic,1200,1200 $PFAL,Cnf.Set,AL3=GSM.GPRS.eDisconnected:IO12.Set=low
Comment	In the examples above following set alarms are implemented: AL0 : Once the AVL device gets GPRS attached, the LED 11 (on the device case) flashes periodically at low frequency. AL1: Once the AVL device gets TCP connected, the LED 11 (on the device case) flashes periodically at high frequency. AL2: Once the AVL device gets TCP disconnected, the LED 12 (on the device case) flashes periodically at low frequency. AL3: Once the AVL device gets GPRS detached, the LED 12 (on the device case) is dark.

11.6.2.9.2. TCPStorage: send special device information to server periodically

TCPStorage can be used to collect data internally before sending it to the TCP server. This prevents the need to send a small TCP packet every time new data is to be transmitted.

Background: each TCP packet requires special information characters, which have to be transmitted anytime – so sending many small packets increases the amount of data and therefore the transmission costs. This cost can be optimized by sending only larger packets, which may contain several "small" data sets. After the TCP storage is configured according to its purpose (the correct buffer size – e.g. 512 Bytes, automatic mode), it can be used for this example, which shows how to implement a solution for specific user requests. For example, the **task** is to transmit the following data periodically to a connected server in order to create a vehicle event log for a server application (i.e. webpage which allows users to view this gathered information).

The server based vehicle Log Report should displays the following detailed data, such as:

- Event time
- Location (street address, city, state)
- Vehicle moved status
- Stop duration for a selected vehicle (available for each stop)
- Miles
- Speed

The time between 2 log reports is 1 minute.

Special consideration when using firmware features

Example	$PFAL,Cnf.Set,AL0=TCP.Client.eConnected:SYS.Timer1.start=cyclic,60000 $PFAL,Cnf.Set,AL1=TCP.Client.eDisconnected:SYS.Timer1.stop $PFAL,Cnf.Set,AL2=GPS.Nav.sSpeed<2&Sys.Trigger.s0=low:Sys.Trigger0=high&SYS.Timer0.start=cyclic,1000&Sys.Counter0.Set=0 $PFAL,Cnf.Set,AL3=SYS.Timer.e0:Sys.Counter0.Increment=1 $PFAL,Cnf.Set,AL4=GPS.Nav.sSpeed>=2&Sys.Trigger.s0=high:Sys.Trigger0=low&SYS.Timer0.stop&[SEND DATA TO SERVER]&SYS.Timer1.start $PFAL,Cnf.Set,AL5=SYS.Timer.e1:[SEND DATA TO SERVER] $PFAL,TCP.Storage.AddProtocol,0,"[&(Time),&(Date),&(Lat),&(Lon),&(Alt),&(Speed),&(NavDist),&(Trigger0)]" $PFAL,Cnf.Set,AL4=GPS.Nav.sSpeed>=2&Sys.Trigger.s0=high:Sys.Trigger0=low&SYS.Timer0.stop&SYS.Timer1.start&TCP.Storage.AddProtocol,0,"[&(Time),&(Date),&(Lat),&(Lon),&(Alt),&(Speed),&(NavDist),&(Trigger0),&(Counter0)]" $PFAL,Cnf.Set,AL5=SYS.Timer.e1:TCP.Storage.AddProtocol,0,"[&(Time),&(Date),&(Lat),&(Lon),&(Alt),&(Speed),&(NavDist),&(Trigger0)]"
Comment	Timers allow the application to raise an event on a specified interval, so that timers can be used to send data periodically at specified time intervals. In order to enable it, start for example a cyclic timer that starts its executation whenever the connection to the server results true and stops its execution otherwise. Next, customize a protocol that contains in a small size (offering cost-efficient) all information you need for each logging report. The customized protocol can be constructed, using the "dynamic protocol" feature. To do this you have to gather all information your application needs. For example: *event time* ➔ &(Time) and/or &(Date) *Location* ➔ &(Lat), &(Lon) and if needed the &(Alt) too *Speed* ➔ &(Speed) *Miles*: ➔ &(NavDist) //distance counter Nav.Dist can be used. This number is shown in metres – so the server has to calculate it to miles/kilometres. The moving vehicle status is not available directly ➔ it has to be created anywhere. Stop duration for a selected vehicle èhas also to be created anywhere. ➔ Furthermore 2 solutions: Device creates this state and transmits it to server Server creates this state (i.e. out of position and time data) In our case let's demonstrate an example how to monitor the stops of the vehicle. Logging a position each minute might be too less awarding short stops for only a few seconds. Using the speed of the device allows creating of stop conditions. When the speed is <2m/s a cyclic Timer (1sec) and Counter will start up. Additionally, a Trigger can be used to display both states "standing" and "moving" while a Counter contains the stop duration in seconds. Standing: If device stops moving, set the state to low (stopping), then start counting of seconds and set the "stop time counter" to 0, (for each second gone, increment the counter). Moving: If device starts moving, set the state to high (moving), then stop counting of seconds, and send to server) Additionally: reset cyclic send counter (this assures that the next "usual" log report is written after 1 minute after device starts moving - if this is not used, both cases are unsynchronized) Now the [SEND DATA TO SERVER] – part: Basically the command for adding a dynamic protocol to the TCP storage is: $PFAL,TCP.Storage.AddProtocol,0,"my text" As all information is available now, the dynamic protocol can be generated. Additionally the counter &(Counter0) is shown, when the device starts moving again. Finally AL4 and AL5 could look this way:

	$PFAL,Cnf.Set,AL4=GPS.Nav.sSpeed>=2&Sys.Trigger.s0=high:Sys.Trigger0=low&SYS.Timer0.stop&SYS.Timer1.start&TCP.Storage.AddProtocol,0,"[&(Time),&(Date),&(Lat),&(Lon),&(Alt),&(Speed),&(NavDist),&(Trigger0),&(Counter0)]"
	$PFAL,Cnf.Set,AL5=SYS.Timer.e1:TCP.Storage.AddProtocol,0,"[&(Time),&(Date),&(Lat),&(Lon),&(Alt),&(Speed),&(NavDist),&(Trigger0)]"

11.6.2.10. Using commands inside alarms

Send commands have a special option, which allows them to be re-executed inside an alarm until they succeed. This behavior assures, that important information cannot get lost due to i.e. bad GSM coverage. Usually Send commands "just" enqueue the information to be sent to a buffer. SMS send, for example, stores the SMS in the SMS outbox. TCP send stores the TCP packets in the TCP buffer (which sometimes is also called TCP history buffer). That means, if a buffer is used up, the sent action is going to fail, only. This causes the alarm to be ‚blocked' until enough space inside the buffer is available to store the message. Such a ‚blocked' alarm attempts to send/ enqueue the information once per second. If this succeeds, the alarm will be deactivated, and it can be activated again.

11.6.2.11. SMS send

Note that, a "blocked" alarm cannot be activated again –even if all its conditions are evaluated to True. Therefore, it is i.e. not possible to send a SMS reporting message each second. The first few SMS would be stored in the *SMS Outbox* until it is used up). Even though the system tries to send SMS away, the *Outbox* will be used up faster than it can be freed by sending. Once outbox is used up, all alarms which attempt to send SMS will be ‚blocked'. They cannot be activated anymore before successfully enquiring their SMS to the *Outbox*.

This result in alarms, which will be launched, not each second anymore – just e.g. each 9 seconds *(whenever the system successfully sends an SMS)*.

11.6.2.12. CSD send

Special care has to be taken when sending data via CSD (data call). It is strongly recommended to include a state to each alarm to check an established CSD connection, which attempts to send CSD data.

Unless SMS or TCP send commands, the CSD does not use an Outbox buffer to store messages. It attempts to send the data directly. Therefore, alarms can be easily blocked, if there is no CSD connection available.

11.6.2.13. Storing information to non-volatile memory

Prevent the saving of states or configurations periodically via alarms or very often. The system is limited to several 100'000 configuration write operations. If the limit (which may vary slightly from device to device) is exceeded, the device has to be replaced. This also includes the saving of states of triggers/counters/timers or positions. Special algorithms are used to increase the number of write operations, when writing history entries. Therefore this functionality is not restricted as much as the configuration writes operations (writing records into the history can be done for many years before the device has to be replaced – it's more likely that writing configurations from time to time exceeds this number faster than writing history entries).

11.7. ISP, GPRS configuration parameters of German service providers

The following table presents GPRS parameters of selected German service providers and operators:

Table 11-5 Service provider information, valid 16.10.2001

	T-D1	Vodafone	E-Plus	Genion 02
APN	internet.t-mobile	web.vodafone.de	internet.eplus.de	internet
ReQoS	Contact your SIM card provider to get the correct settings for Required Quality of Service (ReQoS)			
MiQoS	Contact your SIM card provider to get the correct settings for Minimum Quality of Service (MiQoS)			
USERNAME	d1		eplus	
PASSWORD	gprs		gprs	

11.8. Used abbreviations

Table 11-6 Used Abbreviations

Altitude	The distance between the current position and the nearest point on WGS-84 (World Geodetic System 1984) reference ellipsoid. Altitude is usually expressed in meter an is positive outside the ellipsoid.
APN	Access Point Name
Current position	The GPS is based on satellite ranging - calculating the distances between the device and the position of 3 or more satellites (4 or more if elevation is desired) and then applying some good old mathematics. Assuming the positions of the satellites are known, the location of the device can be calculated by determining the distance from each of the satellites to the device. GPS takes these 3 or more known references and measured distances and "triangulates" an additional position.
CHAP	The MD5 Challenge Handshake authentication Protocol (MD5-CHAP) is used to periodically verify the identity of the peer using a three-way handshake. This is done upon initial link establishment and may be repeated any time after the link has been established.
DOP	Dilution of Precision. Errors caused by bad geometry of the Satellites. The higher the number, the more "noise" in the position reading.
Dynamic IP address	A dynamic IP address is what most of GPRS devices using dial up services. This type of IP address will usually change each and every time you log on to the GPRS services.
FLASH memory	A particular type of memory that is permanent and the data written on it will be not lost when the device is turned off. It can be updated and upgraded by special program.
GPRS	General Packet Radio Service
GPS	The Global Positioning System (GPS) is a location system based on a constellation of about 24 satellites orbiting the earth at altitudes of approximately 11,000 miles. GPS satellites are orbited high enough to avoid the problems associated with land based systems, yet can provide accurate positioning 24 hours a day, anywhere in the world.
GPS Time	The number of seconds since Saturday/Sunday Midnight UTC, with time zero beginning this midnight. Used with GPS Week Number to determine a specific point in GPS Time.
GSM	Global Standard for Mobile Communications
History	A limited part (section) on the on-board FLASH memory that serves to save the user-selected position records (the current data position received from the GPS satellites) and those data can be available for evaluation in a later time.
IMEI	International Mobile Equipment Identity

I/O	Input/Output
IP	Internet Protocol. Providers offer two types of IP addresses –static IP address and dynamic IP address.
Latitude	Halfway between the poles lies equator. Latitude is the angular measurement of a place expressed in degrees north or south of the equator. Latitude runs from 0º at the equator to 90º north or 90º south at the poles. When not prefixed with letters E or W, it is assumed positive north of Equator and negative south of Equator. Lines of latitude run in an east-west direction. They are called parallels.
Locally	The AVL device unit is directly connected to the host device (PC/Laptop etc.) through the serial interface and via supported messages the target device can be configured (e.g. the default settings can be changed and new configured).
Longitude	Lines of longitude, called meridians, run in a north-south direction from pole to pole. Longitude is the angular measurement of a place east or west of the prime meridian. This meridian is also known as the Greenwich Meridian, because it runs through the original site of the Royal Observatory, which was located at Greenwich, just outside London, England. Longitude runs from 0º at the prime Meridian to 180º east or west, halfway around the globe. When not prefixed with letters E or W, it is assumed positive east of Greenwich and negative west of Greenwich. The International Date Line follows the 180º meridian, making a few jogs to avoid cutting through land areas.
NMEA	National Marine Electronics Association
PAP	The Password authentication Protocol (PAP) is an authentication protocol that passes the user name and password in plaintext.
PDOP	The Positional Dilution Of Precision is a numeric value that refers to the geometric suitability of a particular group of GPS satellites. Each satellite in the GPS constellation orbits the earth repeatedly every day. At any given time, there can be from 5 to 12 satellites overhead, each moving in its own predictable path. As the satellites orbit, and the relationship from one to the next changes, the geometry of a solution derived from that group is also affected. Since three-dimensional GPS positions are determined by measured distances from four or more satellites, and as the GPS constellation is in a state of constant movement, PDOP provides a relative gauge by which to measure the end result while the user is in the field. PDOP values will fluctuate constantly throughout the day; however, the fluctuation is generally slight, and for the purposes of most GIS field data collection, it is unnoticeable. A PDOP reading of 2-5 is typical.
PDP	Packet Data Protocol
PPP	Point to Point Protocol
SMTP	Simple Mail Transport Protocol
SMS	Short Message Service
SRAM	Static Random Access Memory, a type of memory that temporarily keeps its information. The content of SRAM is available as long as the internal software of the used device is running. Should the device be reset, switched off, or goes into sleep mode, the SRAM loses the contents forever.
static IP Address	A static IP Address is one, which never changes. This number is assigned by your ISP and the number stays the same until you change Internet providers and request a new one.
TCP	Transmission Control Protocol
UTC	Universal Time Coordinated
WWW	World Wide Web

X, Y, Z positions	Coordinates of user's position in ECEF (meters). The Earth-centered Earth-Fixed (ECEF) is a Cartesian coordinate system with its origin located at the center of the Earth. The coordinate system used by GPS to describe 3-D location. For WGS-84 reference ellipsoid. ECEF coordinated have the Z-axis aligned with the Earth's spin axis, The X-axis through the insertion of the Prime meridian and the Equator and the Y-axis is rotated 90 degrees East of the X-axis about the Z-axis.