Protocol API

PROFINET IO Controller

V3.2.0

Hilscher Gesellschaft für Systemautomation mbH
www.hilscher.com
DOC150403API06EN | Revision 6 | English | 2017-09 | Released | Public

Table of contents

1 Introduction 4
1.1 About this document 4
1.2 List of revisions 4
1.3 Technical data 5
1.3.1 Supported protocols 5
1.3.2 Features of the loadable firmware 5
1.3.3 Restrictions 7
1.3.4 Additional features 8
1.4 References to documents 8
1.5 Legal Notes 9
1.6 Third party software licenses 12

2 Getting started 13
2.1 Configuration of PROFINET IO Controller 13
2.1.1 Overview 13
2.1.2 AR table 14
2.1.3 IOCR tables 14
2.1.4 Submodule table 15
2.1.5 Record storage 15
2.1.6 Packet configuration sequence 16
2.2 Process data handling 17
2.2.1 Output data handling 17
2.2.2 Input data handling 20
2.2.3 Process data timing in isochronous applications 23
2.3 Configuration / Network state 24

3 Application interface 25
3.1 Configuration 25
3.1.1 Configure OEM Parameter service 25
3.1.2 Configure IO Controller service 33
3.1.3 Configure IO Controller parameter service 41
3.1.4 Configure IO Device service 46
3.1.5 Configure AR parameters service 53
3.1.6 Configure IOCR service 57
3.1.7 Configure submodule service 65
3.1.8 Configure record service 71
3.1.9 Configure topology service 76
3.1.10 Download finished service 79
3.1.11 Load Remanent service 81
3.1.12 Configure record structures 83
3.2 Acyclic requests 118
3.2.1 Read submodule record service 118
3.2.2 Write submodule record service 124
3.2.3 Read implicit record service 128
3.2.4 Acknowledge Alarm service 134
3.2.5 DCP Set Name service 137
3.2.6 DCP Set IP service 140
3.2.7 DCP Set Signal Service 144
3.2.8 DCP Reset Factory Settings service 147
3.2.9 DCP Get service 151
3.2.10 Get Logbook Service 156
3.2.11 Get AR Vendor Block Response Service 160
3.2.12 Set AR Status service 163
3.2.13 Establish Device Access AR 165
3.2.14 Release Device Access AR 169
3.3 Network scan 171
3.4 Device Access AR (DA-AR) 174
3.4.1 Overview 174
3.4.2 Usage 175
3.5 Acyclic indications 176
3.5.1 Receive Alarm service 176
3.5.2 Receive Diagnosis service 181
3.5.3 Store Remanent service 188
3.6 Get configuration services 190

© Hilscher, 2015–2017

3.6.1 Get IO Controller service ... 191
3.6.2 Get IO Controller parameter list service ... 199
3.6.3 Get IO Controller parameter service ... 201
3.6.4 Get number of configured objects service ... 206
3.6.5 Get IO Device service ... 209
3.6.6 Get AR parameter list service ... 216
3.6.7 Get AR parameter service ... 218
3.6.8 Get IOCR service ... 223
3.6.9 Get submodule service ... 231
3.6.10 Get record service ... 238
3.7 Legacy services ... 241
3.7.1 Read Record service (legacy) ... 241
3.7.2 Write Record service (legacy) ... 245
3.7.3 Read Record Implicit service (legacy) ... 249
3.8 Common services ... 254
3.8.1 Get Slave Handle service ... 254
3.8.2 Get Slave Connection Info service ... 257
3.8.3 Link Status Changed service ... 261
3.9 DPM ... 264
3.9.1 Extended Status Block ... 264
3.9.2 Process data timing information ... 265
3.10 Fragmented packet transfer ... 267

4 Media Redundancy ... 269
4.1 Overview ... 269
4.2 Topologies ... 270
4.3 Configuration parameters ... 271

5 LED ... 273

6 Certification requirements for applications ... 274

7 Status codes / Error codes ... 276
7.1 Error Codes of PROFINET IO Controller ... 276
7.2 Sockets ... 279
7.3 PROFINET Status Code ... 279
7.3.1 The ErrorCode Field ... 280
7.3.2 The ErrorDecode Field ... 280
7.3.3 The ErrorCode1 and ErrorCode2 Fields ... 281
7.3.4 ErrorCode1 and ErrorCode2 for ErrorDecode = PNIO ... 282
7.3.5 ErrorCode1 and ErrorCode2 for ErrorDecode is Manufacturer Specific ... 287

8 Appendix ... 288
8.1 List of tables ... 288
8.2 List of figures ... 288
8.3 Contacts ... 290

© Hilscher, 2015–2017

1 Introduction

1.1 About this document

This manual describes the application interface of the PROFINET IO Controller protocol stack. The aim of this manual is to support the integration of netX-based devices into applications.

1.2 List of revisions

Rev	Date	Name	Revision
4	2016-04-20	BM, AM, HH	Firmware/stack version 3.1.0 Section added.
5	2016-12-07	AM, RA, HH	Firmware/stack version 3.2.0 Section *PNM_AP_CFG_IOC_REQ_T request* updated. Section *Load Remanent service* added. Section *Store Remanent service* added. Section *Process data timing in isochronous applications* added. Section *DPM* added. Section *PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_T structure* added. Section *PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_T structure* added. MRP information updated. Section *Media Redundancy* added. LWIP licensing information added. Section *Configure OEM Parameter service* updated. Section PNM_AP_CFG_IOD_ADDRESS_MODE_E enumeration added. Section *Fragmented packet transfer* added.
6	2017-09-06	BM, AM	Firmware/stack version 3.2.0 Section *Get configuration services* added Fix wrong numbers for `ulStructID` in RCX_GET_SLAVE_CONN_INFO_CNF_T confirmation Section *Configure OEM Parameter service* updated. Section *Get Slave Connection Info service* updated. Fix wrong submodule handle values for internal submodules. Add newly implemented controller local record object indices. Section *PROFINET Status Code* added.

Table 1: List of revisions

© Hilscher, 2015–2017

1.3 Technical data

The data below applies to IO Controller firmware and stack version V3.2.0.

1.3.1 Supported protocols

- RTC - Real time protocol cyclic
- RTA - Real time protocol acyclic
- DCP - Discovery and Configuration Protocol
- CL-RPC - Connectionless Remote Procedure Call
- LLDP - Link Layer Discovery Protocol
- PTCP - Precision Transparent Clock Protocol

1.3.2 Features of the loadable firmware

Feature	Value	Remark
Maximum number of ARs	128	This is the total maximum number of ARs supported by the controller.
Maximum number of ARs for RT communication	128	-
Maximum number of ARs for IRT communication	64	The netX supports up to 64 Input and Output IOCRs in IRT Mode
Process Data Input	5652 byte	The Input Area of DPM consists of the input process data and three 16 byte status bit lists. The Process data includes the PROFINET Provider and Consumer States.
Output data, total	5700 byte	The Process data includes the PROFINET Provider and Consumer States.
Supported send clock for RT mode	1 ms, 2 ms, 4 ms	If IRT mode is used for at least one AR, the minimum possible send clock for RT mode is additionally limited to greater or equal values than the send clock of that IRT AR.
Supported send clock for IRT mode	250 µs, 500 µs, 1 ms, 2 ms, 4 ms	
Performance Limits of ARs	max. 8 ARs if any Sendclock < 500 µs max. 16 ARs if any Sendclock < 1 ms max. 64 ARs if any Sendclock < 2 ms	The maximum number of possible ARs is limited by the minimum send clock value used in the configuration.
Maximum amount of data for acyclic read/write record access	65536 byte	This limit applies to Acyclic Implicit Read, Acyclic Read and Acyclic Write services. When the size of confirmation/request packets exceeds the mailbox size a fragmented packet transfer is required. Only one fragmented instance per service may be active at the same time.
Number of IOCRs per AR	1x Input IOCR 1x Output IOCR	
Maximum amount of data per IOCR	1440	This is the physical limit of the Ethernet frame size. The IOCR's data consists of process data and the associated provider and consumer states. Thus the amount of available process data per IOCR depends on the number of submodules configured for that IOCR.

© Hilscher, 2015–2017

Feature	Value	Remark
Maximum number of submodules	2048	The globally defined maximum number of submodules the controller can handle. The submodules can be arbitrarily distributed across the configured ARs and IOCRs. The number of submodules per IOCR is only limited by the IOCR data size
Maximum number of ARVendorBlock	256	The firmware supports a global number of up to 256 ARVendorBlocks which. Each ARVendorBlock is associated with an AR. There is no limitation on the number of ARVendorBlocks per AR.
Maximum size of ARVendorBlockData	512 byte	The ARVendorBlockReq and the ARVendorBlockRes data are each limited to this value.
Maximum amount of record data per AR	16384 byte	The maximum amount of record data to be configured for one AR. This record data is transferred to the device when the AR is established and includes beyond others: PDEV parameterization and device specific record data.
Time for TX data update (Time_TxUpd_Min)	98 µs	This is the time required by the firmware to prepare the transmitt buffers from DPM output area. The time is independent from number of IOCRs and Data length.
Time for RX data processing (Time_RxUpd_Max)	74 µs + 2.3 µs per IOCR	This is the time required by the firmware to prepare the DPM input area from the receive buffers. The time depends on the number of IOCRs and their size. The given time is for 40 byte IOCR data.
Device Access AR CMI Timeout	20 s	The DA-AR uses a CMI Timeout of 20 seconds.
Alarm processing	yes	The PROFINET IO Controller can be configured to perform automatic alarm handling. For each alarm type it can be configured if the alarm is to be handled by the PROFINET IO Controller firmware itself or if the alarm shall be passed to the application for further processing. Using automatic alarm handling reduces the amount of indications passed to the application and can simplify application implementation.
DCP functions via API	Name Assignment IO-Devices (DCP SETNameOfStation) Set IO-Devices IP (DCP SET IP) Signal IO-Device (DCP SET SIGNAL) Reset IO-Device to factory settings (DCP Reset FactorySettings) Bus scan (DCP IDENTIFY ALL) DCP GET	-
PROFINET specification	Implementation based on PROFINET IO specification v2.3 ED2 MU3 Legacy startup (communication to IO Devices according to PROFINET IO specification V2.2) is supported	-
Automatic Name Assignment	yes	The PROFINET IO Controller implements automatic name assignment. That means, that the controller can automatically assign the PROFINET name of station according configured topology information to any unnamed device in the network. With that feature it is possible to e.g. exchange faulty PROFINET IO Devices without factory new devices without additional effort.

© Hilscher, 2015–2017

Feature	Value	Remark
Media Redundancy protocol	Media Redundancy Client since V3.1.0.0 Media Redundancy Manager since V3.2.0.0	The PROFINET controller implements the Media Redundancy Protocol required to configure a ring topology. To use MRP Manager functionality an additional license bit in SecurityMemory is required

Table 2: Technical data

1.3.3 Restrictions

- The size of the bus configuration file is limited by the size of the RAM Disk (1 Megabyte)
- The configuration database interface and its structure and definitions apply for netX products which use the PROFINET IO Controller - firmware only.
- Structures and functions described in this document apply only to hardware from 3rd party vendors insofar as original Hilscher firmware is concerned. Therefore, whenever the term "netX firmware" is used throughout this manual it refers to ready-made firmware provided by Hilscher. Although 3rd party vendors are free to implement the same structures and functions in their product, no guarantee for compatibility of drivers etc. can be given.
- The following limitations apply
  - The usable (minimum) cycle time depends on the number of used IO Devices, the number of used input and output data. See technical data for details
  - RT over UDP not supported
  - Multicast communication not supported
  - DHCP is not supported (neither for PROFINET IO Controller nor for the IO-Devices)
  - Only one IOCR per IO-Device per direction
  - only one instance of DeviceAccess AR can be used at the same time
  - MRPD is not supported
  - planing of IRT is not done by the PROFINET IO Controller protocol stack
  - Sync Slave is not supported
  - MRP Manager requires an additional license in security memory
  - only one fragmented acyclic services can be used at the same time
  - Multiple MRP Managers are not supported
  - only one DCP Service can be used in parallel
  - Multiple Sync Masters are not supported

© Hilscher, 2015–2017

1.3.4 Additional features

Automatic alarm handling

The PROFINET IO Controller can be configured to perform automatic alarm handling. For each alarm type it can be configured if the alarm is to be handled by the PROFINET IO Controller firmware itself or if the alarm shall be passed to the application for further processing. Using automatic alarm handling reduces the amount of indications passed to the application and can simplify application implementation.

However automatic alarm handling needs to be used with care. Process relevant alarms (e.g. alarm type Process Alarm) should never be handled automatically.

Fast startup

The PROFINET IO Controller supports Fast Startup (FSU). This feature can be configured independently per AR.

Automatic name assignment

The PROFINET IO Controller implements automatic name assignment. That means, that the controller can automatically assign the PROFINET name of station according a previously configured topology information to any unnamed device in the network. With that feature it is possible to e.g. exchange faulty PROFINET IO Devices without factory new devices without additional effort.

Device Access AR

The PROFINET IO Controller implements Device Access AR. This AR Type is usually used to write I&M record objects by the engineering software.

1.4 References to documents

This document refers to the following documents:

[1] PROFIBUS International: Technical Specification for PROFINET IO: Application Layer protocol for decentralized periphery, Version 2.3Ed2MU3, March 2016, Order No. 2.722, English.

[2] Hilscher Gesellschaft für Systemautomation mbH: DPM Interface Manual for netX based Products, 2012, Revision 12. English.

 © Hilscher, 2015–2017

1.5 Legal Notes

Copyright

© Hilscher Gesellschaft für Systemautomation mbH

All rights reserved.

The images, photographs and texts in the accompanying materials (in the form of a user's manual, operator's manual, Statement of Work document and all other document types, support texts, documentation, etc.) are protected by German and international copyright and by international trade and protective provisions. Without the prior written consent, you do not have permission to duplicate them either in full or in part using technical or mechanical methods (print, photocopy or any other method), to edit them using electronic systems or to transfer them. You are not permitted to make changes to copyright notices, markings, trademarks or ownership declarations. Illustrations are provided without taking the patent situation into account. Any company names and product designations provided in this document may be brands or trademarks by the corresponding owner and may be protected under trademark, brand or patent law. Any form of further use shall require the express consent from the relevant owner of the rights.

Important notes

Utmost care was/is given in the preparation of the documentation at hand consisting of a user's manual, operating manual and any other document type and accompanying texts. However, errors cannot be ruled out. Therefore, we cannot assume any guarantee or legal responsibility for erroneous information or liability of any kind. You are hereby made aware that descriptions found in the user's manual, the accompanying texts and the documentation neither represent a guarantee nor any indication on proper use as stipulated in the agreement or a promised attribute. It cannot be ruled out that the user's manual, the accompanying texts and the documentation do not completely match the described attributes, standards or any other data for the delivered product. A warranty or guarantee with respect to the correctness or accuracy of the information is not assumed.

We reserve the right to modify our products and the specifications for such as well as the corresponding documentation in the form of a user's manual, operating manual and/or any other document types and accompanying texts at any time and without notice without being required to notify of said modification. Changes shall be taken into account in future manuals and do not represent an obligation of any kind, in particular there shall be no right to have delivered documents revised. The manual delivered with the product shall apply.

Under no circumstances shall Hilscher Gesellschaft für Systemautomation mbH be liable for direct, indirect, ancillary or subsequent damage, or for any loss of income, which may arise after use of the information contained herein.

Liability disclaimer

The hardware and/or software was created and tested by Hilscher Gesellschaft für Systemautomation mbH with utmost care and is made available as is. No warranty can be assumed for the performance or flawlessness of the hardware and/or software under all application conditions and scenarios and the work results achieved by the user when using the hardware and/or software. Liability for any damage that may have occurred as a result of using the hardware and/or software or the corresponding documents shall be limited to an event involving willful intent or a grossly negligent violation of a fundamental contractual obligation. However, the right to assert damages due to a violation of a fundamental contractual obligation shall be limited to contract-typical foreseeable damage.

It is hereby expressly agreed upon in particular that any use or utilization of the hardware and/or software in connection with

- Flight control systems in aviation and aerospace;
- Nuclear fusion processes in nuclear power plants;
- Medical devices used for life support and
- Vehicle control systems used in passenger transport

shall be excluded. Use of the hardware and/or software in any of the following areas is strictly prohibited:

- For military purposes or in weaponry;
- For designing, engineering, maintaining or operating nuclear systems;
- In flight safety systems, aviation and flight telecommunications systems;
- In life-support systems;
- In systems in which any malfunction in the hardware and/or software may result in physical injuries or fatalities.

You are hereby made aware that the hardware and/or software was not created for use in hazardous environments, which require fail-safe control mechanisms. Use of the hardware and/or software in this kind of environment shall be at your own risk; any liability for damage or loss due to impermissible use shall be excluded.

Warranty

Hilscher Gesellschaft für Systemautomation mbH hereby guarantees that the software shall run without errors in accordance with the requirements listed in the specifications and that there were no defects on the date of acceptance. The warranty period shall be 12 months commencing as of the date of acceptance or purchase (with express declaration or implied, by customer's conclusive behavior, e.g. putting into operation permanently).

The warranty obligation for equipment (hardware) we produce is 36 months, calculated as of the date of delivery ex works. The aforementioned provisions shall not apply if longer warranty periods are mandatory by law pursuant to Section 438 (1.2) BGB, Section 479 (1) BGB and Section 634a (1) BGB [Bürgerliches Gesetzbuch; German Civil Code] If, despite of all due care taken, the delivered product should have a defect, which already existed at the time of the transfer of risk, it shall be at our discretion to either repair the product or to deliver a replacement product, subject to timely notification of defect.

The warranty obligation shall not apply if the notification of defect is not asserted promptly, if the purchaser or third party has tampered with the products, if the defect is the result of natural wear, was caused by unfavorable operating conditions or is due to violations against our operating regulations or against rules of good electrical engineering practice, or if our request to return the defective object is not promptly complied with.

Costs of support, maintenance, customization and product care

Please be advised that any subsequent improvement shall only be free of charge if a defect is found. Any form of technical support, maintenance and customization is not a warranty service, but instead shall be charged extra.

Additional guarantees

Although the hardware and software was developed and tested in-depth with greatest care, Hilscher Gesellschaft für Systemautomation mbH shall not assume any guarantee for the suitability thereof for any purpose that was not confirmed in writing. No guarantee can be granted whereby

© Hilscher, 2015–2017

the hardware and software satisfies your requirements, or the use of the hardware and/or software is uninterruptable or the hardware and/or software is fault-free.

It cannot be guaranteed that patents and/or ownership privileges have not been infringed upon or violated or that the products are free from third-party influence. No additional guarantees or promises shall be made as to whether the product is market current, free from deficiency in title, or can be integrated or is usable for specific purposes, unless such guarantees or promises are required under existing law and cannot be restricted.

Confidentiality

The customer hereby expressly acknowledges that this document contains trade secrets, information protected by copyright and other patent and ownership privileges as well as any related rights of Hilscher Gesellschaft für Systemautomation mbH. The customer agrees to treat as confidential all of the information made available to customer by Hilscher Gesellschaft für Systemautomation mbH and rights, which were disclosed by Hilscher Gesellschaft für Systemautomation mbH and that were made accessible as well as the terms and conditions of this agreement itself.

The parties hereby agree to one another that the information that each party receives from the other party respectively is and shall remain the intellectual property of said other party, unless provided for otherwise in a contractual agreement.

The customer must not allow any third party to become knowledgeable of this expertise and shall only provide knowledge thereof to authorized users as appropriate and necessary. Companies associated with the customer shall not be deemed third parties. The customer must obligate authorized users to confidentiality. The customer should only use the confidential information in connection with the performances specified in this agreement.

The customer must not use this confidential information to his own advantage or for his own purposes or rather to the advantage or for the purpose of a third party, nor must it be used for commercial purposes and this confidential information must only be used to the extent provided for in this agreement or otherwise to the extent as expressly authorized by the disclosing party in written form. The customer has the right, subject to the obligation to confidentiality, to disclose the terms and conditions of this agreement directly to his legal and financial consultants as would be required for the customer's normal business operation.

Export provisions

The delivered product (including technical data) is subject to the legal export and/or import laws as well as any associated regulations of various countries, especially such laws applicable in Germany and in the United States. The products / hardware / software must not be exported into such countries for which export is prohibited under US American export control laws and its supplementary provisions. You hereby agree to strictly follow the regulations and to yourself be responsible for observing them. You are hereby made aware that you may be required to obtain governmental approval to export, reexport or import the product.

© Hilscher, 2015–2017

1.6 Third party software licenses

lwIP IP stack

This software package uses the lwIP software for IP stack functionality. The following licensing conditions apply for this component:

Copyright (c) 2001-2004 Swedish Institute of Computer Science.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

© Hilscher, 2015–2017

2 Getting started

The following section gives an overview about general topics when using the PROFINET IO Controller.

2.1 Configuration of PROFINET IO Controller

2.1.1 Overview

Prior to communication with any PROFINET IO Device the PROFINET IO Controller must be configured. For that purpose the following possibilities exist:

- Configuration by the application using the packet interface
- Configuration by configuration file named `config.nxd`

Note: The configuration file format was incompatibely changed between firmware version 3.1.x and 3.2.x. Firmware version 3.1.x used “config.dat” whereas firmware version 3.2.x is using “config.nxd”.

The configuration data will be internally stored into tables like structures as shown in the following figure.

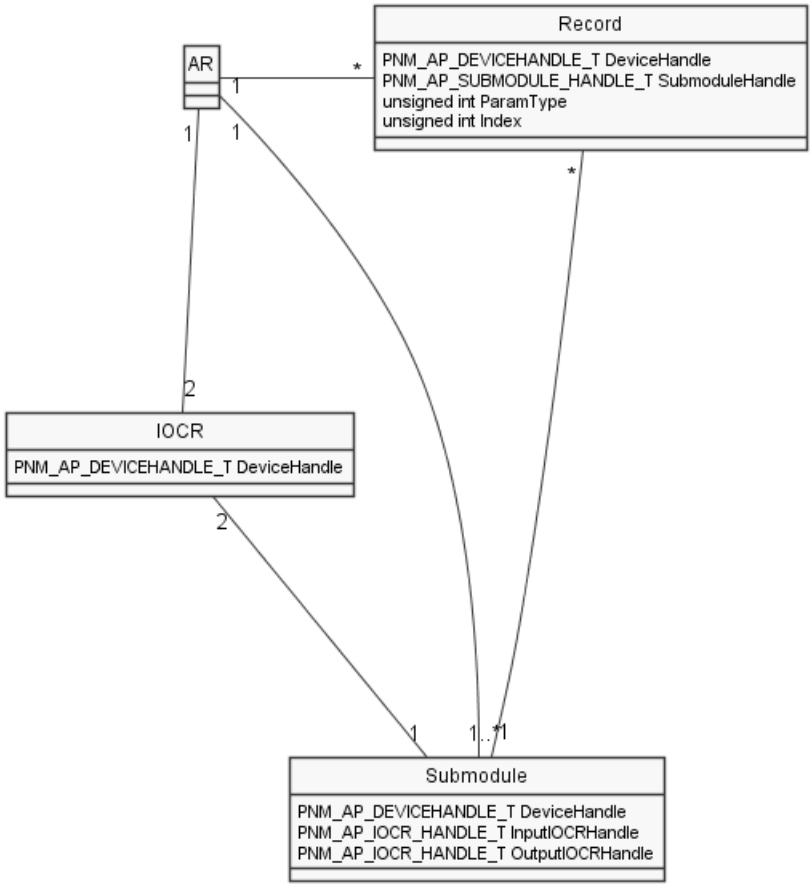

Figure 1: Structure of configuration database

© Hilscher, 2015–2017

2.1.2 AR table

This table is used to store basic information of each AR. A table entry is addressed by a device handle of type PNM_AP_DEVICEHANDLE_T. The following values are valid values for the device handle:

Value	Definition	Description
0	-	Implicit Handle referencing to the controller itself.
1-128	-	Handle of an AR of a PROFINET IO Device. The maximum value depends on the number of supported ARs.

Table 3: Valid values of the device handle

2.1.3 IOCR tables

The PROFINET IO Controller uses two tables to store the information related to an IOCR: one table to hold information about the input IOCRs and one table to hold the information about the output IOCRs. Both IOCR types are addressed by an IOCR handle of type PNM_AP_IOCR_HANDLE_T. The following table shows valid values for the IOCR Handle:

Value	Definition	Description
0x1000 to 0x103F	PNM_AP_IOCR_HANDLE_INPUT_FIRST to -	Handle of an input IOCR in RT or IRT mode
0x1040 to 0x107F	- to PNM_AP_IOCR_HANDLE_INPUT_LAST	Handle of an input IOCR in RT mode
0x2000 to 0x203F	PNM_AP_IOCR_HANDLE_OUTPUT_FIRST to -	Handle of an output IOCR in RT or IRT mode
0x2040 to 0x207F	- to PNM_AP_IOCR_HANDLE_OUTPUT_LAST	Handle of an output IOCR in RT mode

Table 4: Valid values of the IOCR handle

Note: Due to the internal firmware structure a special constraint applies to the frame id of any input IOCRs and the frame id of IRT output IOCRs. Thus the frame id is automatically generated from the IOCR handle.

Depending on the IOCR's handle, the IOCR Type and the direction the PROFINET IO Controller will internally generate a frame id for the associated IOCR.

IOCR Handle	RTC Class	Frame Id	Remark
0x1000 to 0x103F	RTC1 Legacy	0xC000 to 0xC03F	-
	RTC1	0x8000 to 0x803F	-
	RTC3	0x0100 to 0x013F	-
0x1040 to 0x107F	RTC1 Legacy	0xC040 to 0xC07F	-
	RTC1	0x8040 to 0x807F	
0x2000 to 0x203F	RTC1 Legacy	-	The frame id to use is specified by the PROFINET IO Device on Startup
	RTC1	-	The frame id to use is specified by the PROFINET IO Device on Startup
	RTC3	0x0200 to 0x023F	-
0x2040 to 0x207F	RTC1 Legacy	-	The frame id to use is specified by the PROFINET IO Device on Startup
	RTC1	-	The frame id to use is specified by the PROFINET IO Device on Startup

Table 5: Internally generated frame id values

© Hilscher, 2015–2017

2.1.4 Submodule table

The controller maintains submodule handle to hold any information related to a particular submodule. The table entries are addressed by a submodule handle of type PNM_AP_SUBMODULE_HANDLE_T. The following values are valid submodule handles:

Value	Definition	Description
0x0001 to 0x0800	-	Handle to a submodule associated with a particular PROFINET IO Device.
0xFFF0	-	Handle to virtual submodule referencing the PROFINET IO Controller PD Interface submodule
0xFFF1	-	Handle to virtual submodule referencing the PROFINET IO Controller PD Port 1 submodule
0xFFF2	-	Handle to virtual submodule referencing the PROFINET IO Controller PD Port 2 submodule

Table 6: Valid values for a submodule handle

2.1.5 Record storage

The record storage is a memory buffer where all record parameters of the submodules of one AR are stored. Each AR is associated its own record storage. The record data is ordered and distinguished by submodule and record index.

© Hilscher, 2015–2017

2.1.6 Packet configuration sequence

In order to configure the PROFINET IO Controller using the Packet Application Interface, the following steps have to be performed:

Step	Configuration	Service / Section	Page
1	**Configure basic Controller parameters** This service shall be used to configure Network Parameters, PROFINET IDs and basic DPM settings of the PROFINET IO Controller firmware.	*Configure IO Controller service*	33
2	**Configure Controller PDEV parameters** This step is optional. It should be used to configure PD Parameters of the Controllers Port Submodules and PTCP and IRT Settings of the Controllers Interface Submodule.	*Configure IO Controller parameter service*	41
3	**Configure all application relations of the Controller** This service shall be used to configure basic settings of an AR like Network Parameters, PROFINET IDs and AR Type.	*Configure IO Device service*	46
4	**Configure all IOCRs associated with the ARs from previous step** This service shall be used to configure each IOCR associated with an AR. It specifies IOCR related parameters like IOCR type, Frame length and Position of the IOCR data within the DPM Input and Output Areas.	*Configure IOCR service*	57
5	**Configure all submodules associated with the ARs and IOCRs from the previous step** This service configures the PROFINET IDs of a submodule and the data offsets within the associated IOCRs. The data offset specifies the position of the submodules data and data states within the DPM Input and Output Areas	*Configure submodule service*	65
6	**Configure Submodule Record Parameters for all Submodules** This service is optional. It should be used to configure manufacturer specific record data and PROFINET IO related record data (PD related records like Neighborhood Checks, IRT and PTCP Configuration).	*Configure record service*	71
7	**Configure Network Topology** This service is optional. The step is required if the Controller shall automatically assign the PROFINET NameOfStation to unnamed devices.	*Configure topology service*	76
8	**Issue Configuration Done Service** As last step this service shall be used by the application. The controller will perform a final verification of the configuration and activate communication if requested.		

Table 7: Packet configuration sequence

© Hilscher, 2015–2017

2.2 Process data handling

2.2.1 Output data handling

The transmit data is data which is sent from a device to the bus. This means for the PROFINET IO Controller that the transmit data corresponds to the output process Data. In contrast to this, the transmit data is the input process data for the PROFINET IO Device. Remind that the viewpoint is always the PROFINET IO Controller application.

Due to the implementation, the data sent to one remote station (the data within one frame) will be always consistent. Additional consistency is imposed for IRT data.

2.2.1.1 Output data timing

The following figure shows the timing relations between the application, the DPM, the protocol stack and the network. In case of pure RT communication, the “Red Phase” does not exist in the network. However even in this case, the firmware internal event “start of red phase” exists in the firmware.

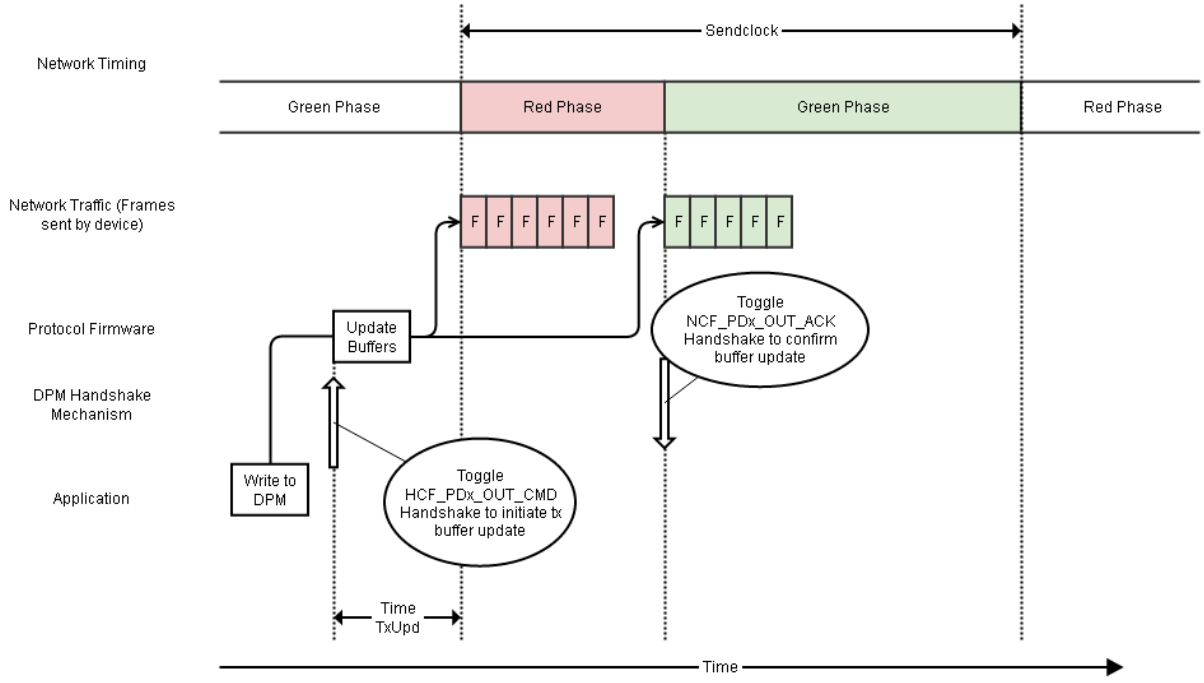

Figure 2: Timing diagram for output data

The output data is handled as follows for the PROFINET protocol stack:

1. The application writes the data into the DPM output area. Afterwards the application toggles the HCF_PDx_OUT_CMD handshake bit.
2. When the HCF_PDx_OUT_CMD handshake bit is toggled, the firmware will start updating its transmit buffers from the DPM output area. If the update is finished before the next send clock starts, the new data will be transferred in the next red phase. If the update is not finished before the next red phase starts, data from the previous cycle will be used again in the next red phase and the new data will be used in the next to next red phase.
3. The firmware will confirm the buffer update on the next green phase begin after finishing the transmitt buffer update by toggling the NCF_PDx_OUT_ACK handshake bit.

© Hilscher, 2015–2017

The firmware requires a certain amount of time to prepare the transmit buffers from the DPM output area. This time is denoted as Time_TxUpd_Min. The application must toggle the HCF_PDx_OUT_CMD handshake bit at least Time_TxUpd_Min before the next send clock starts. Otherwise the transmit buffer update cannot be finished before the next red phase starts. If this happens, the firmware will send the same process data as in the previous cycle to the network and delay the new process data by an additional send clock. The time Time_TxUpd_Min depends on the amount and structure of the process data, the firmware, version and implementation type. Details can be found in the coresponding protocol API manual.

Due to this impementation the process data sent to the network in the red phase will be always consistent across all frames in one red phase. That means either all or none of transmitted frames of a red phase contain the new data.

Note: Due to this implementation the NCF_PDx_OUT_ACK handshake bit will be toggled atmost once per cycle, when the green phase starts. Thus the firmware enforces that the application cannot update the process data more than once per sendclock. This leads to the following behavior:

1. The firmware is implicitly protected from being overloaded by the application due to too frequent process data update.
2. The application is implicitly synchronized to the network timing. This can be regarded as a light form of bus synchronization of the application process.

© Hilscher, 2015–2017

2.2.1.2 Output data in context of the cifX API

When using the cifX API, the function xChannelIOWrite is used to update the output process data. The next figure shows the sequence when this function is onvoked by the application

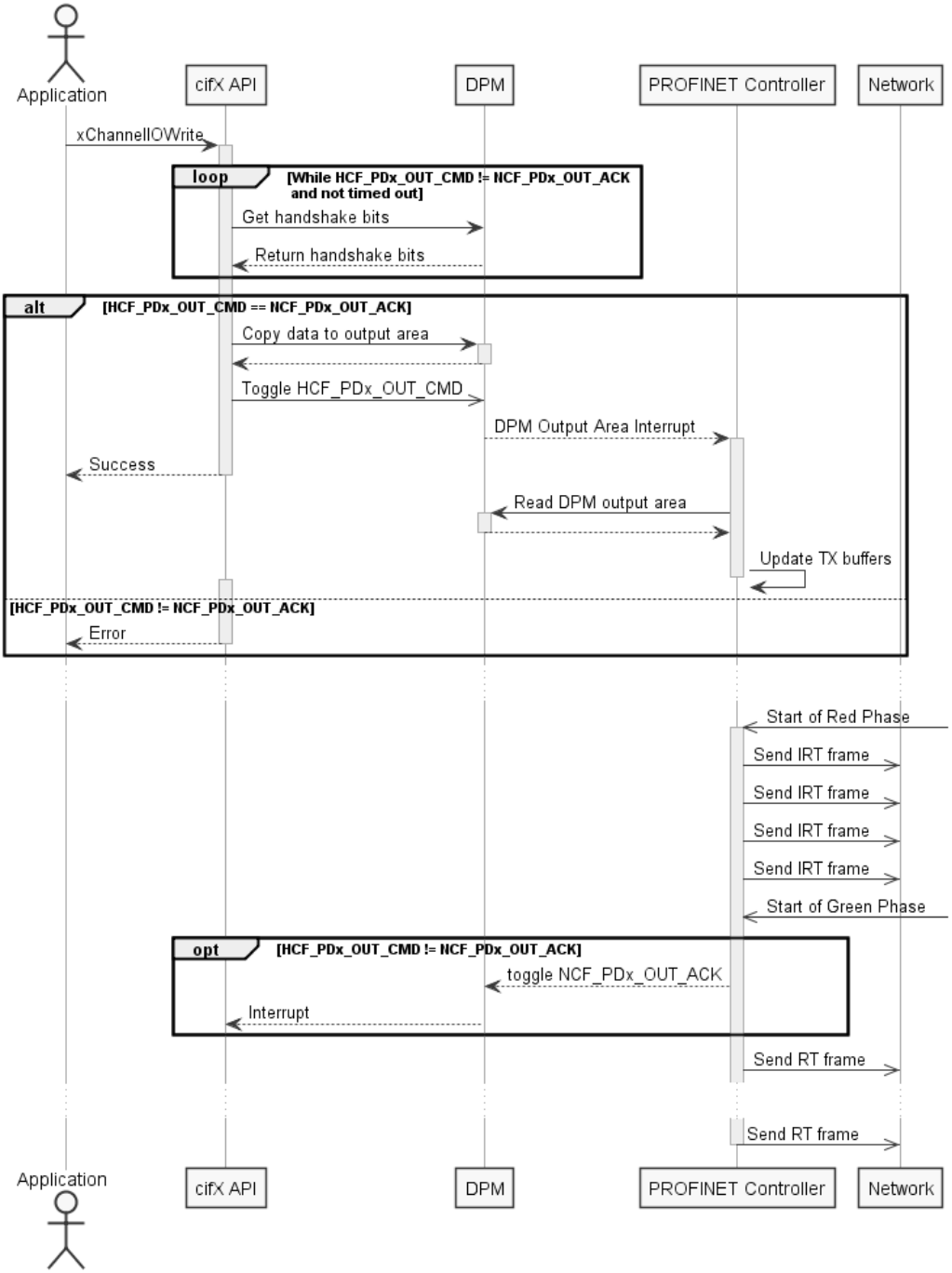

Figure 3: Sequence when using xChannelIOWrite

© Hilscher, 2015–2017

2.2.2 Input data handling

The receive data is data which is received at the device from the network. This means for the PROFINET IO Controller the receive data corresponds to the input process data. In contrast to this, the receive data is the output process data for the PROFINET IO Device. Remind that the viewpoint is always the PROFINET Controller IO application.

2.2.2.1 Input data timing

The following figure shows the timing relations between the application, the DPM, the protocol stack and the network. In case of pure RT communication, the “Red Phase” does not exist in the network. However even in this case, the firmware internal event “start of red phase” exists in the firmware.

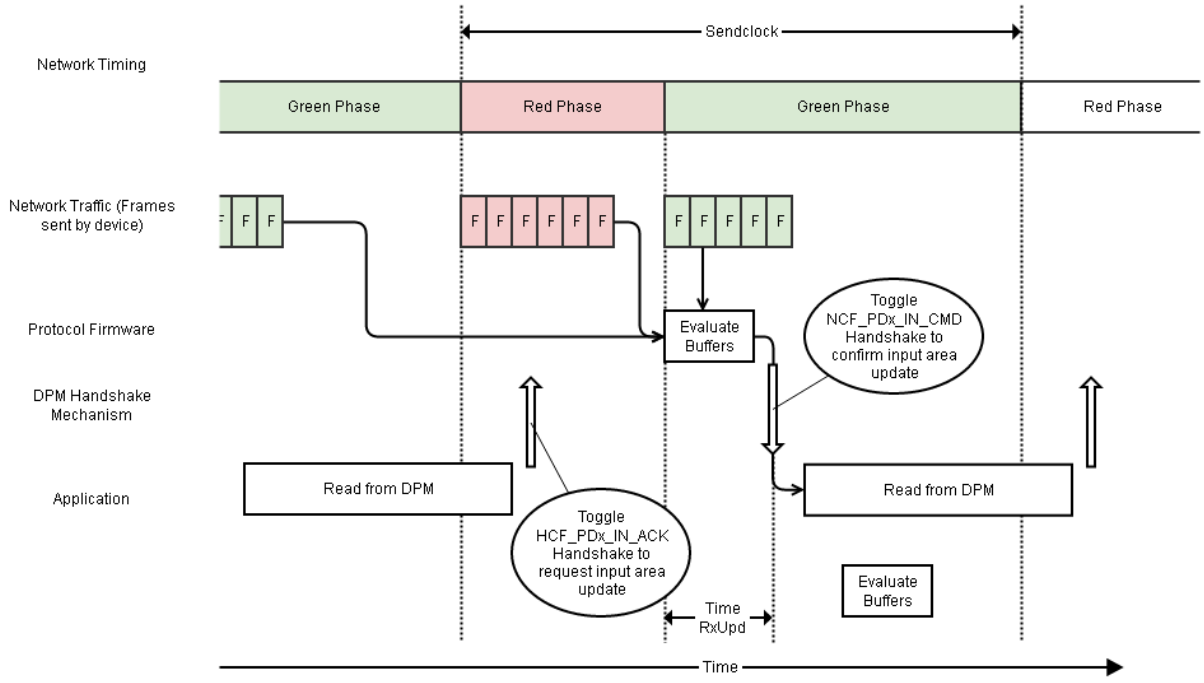

Figure 4: Timing diagram for input data

The receive data is handled as follows for the PROFINET protocol stack:

1. The application toggles the HCF_PDx_IN_ACK handshake bit to request the DPM input area update. The handshake toggle will be notified by the firmware but will not cause an immediate action.
2. When the next green phase starts the firmware will check if the HCF_PDx_IN_ACK handshake bit had been toggled before. If that is the case, the DPM input area will be updated with the received process data from the preceding red phase and the previous and possibly current green phase. This depends on receive timing in green phase. The time needed by the firmware for doing so is denoted as Time_RxUpd. After performing the update, the firmware will confirm the DPM input area update by toggling the NCF_PDx_IN_CMD handshake bit.
3. The application can now access the DPM input area to get the received data.

© Hilscher, 2015–2017

In order to get the receive process data within the same cycle, the application must toggle the HCF_PDx_IN_ACK handshake bit before the next green phase starts. Otherwise the process data will be updated on the next-to-next green phase begin. The firmware requires a certain amount of time to evaluate the receive buffers into the DPM input area. The maximum time required for this task is denoted as Time_RxUpd_Max. It depends on the amount and structure of the process data, the firmware, version and implementation type.

Note: Due to this implementation the NCF_PDx_IN_CMD handshake bit will be toggled at most once per cycle, after the begin of the green phase when all received process data had been evaluated. Thus the firmware enforces that the application cannot retrieve the process data more than once per sendclock. This leads to the following behavior:

1. The firmware is implicitly protected from being overloaded by the application due to too frequent process data evaluation requests
2. The application is implicitly synchronized to the network timing. This can be regarded as a light form of bus synchronization of the application process

2.2.2.2 Input data in context of the cifX API

If using the cifX API, the function xChannelIORead is used to retrieve the input process data. The next figure shows the sequence when this function is invoked by the application.

© Hilscher, 2015–2017

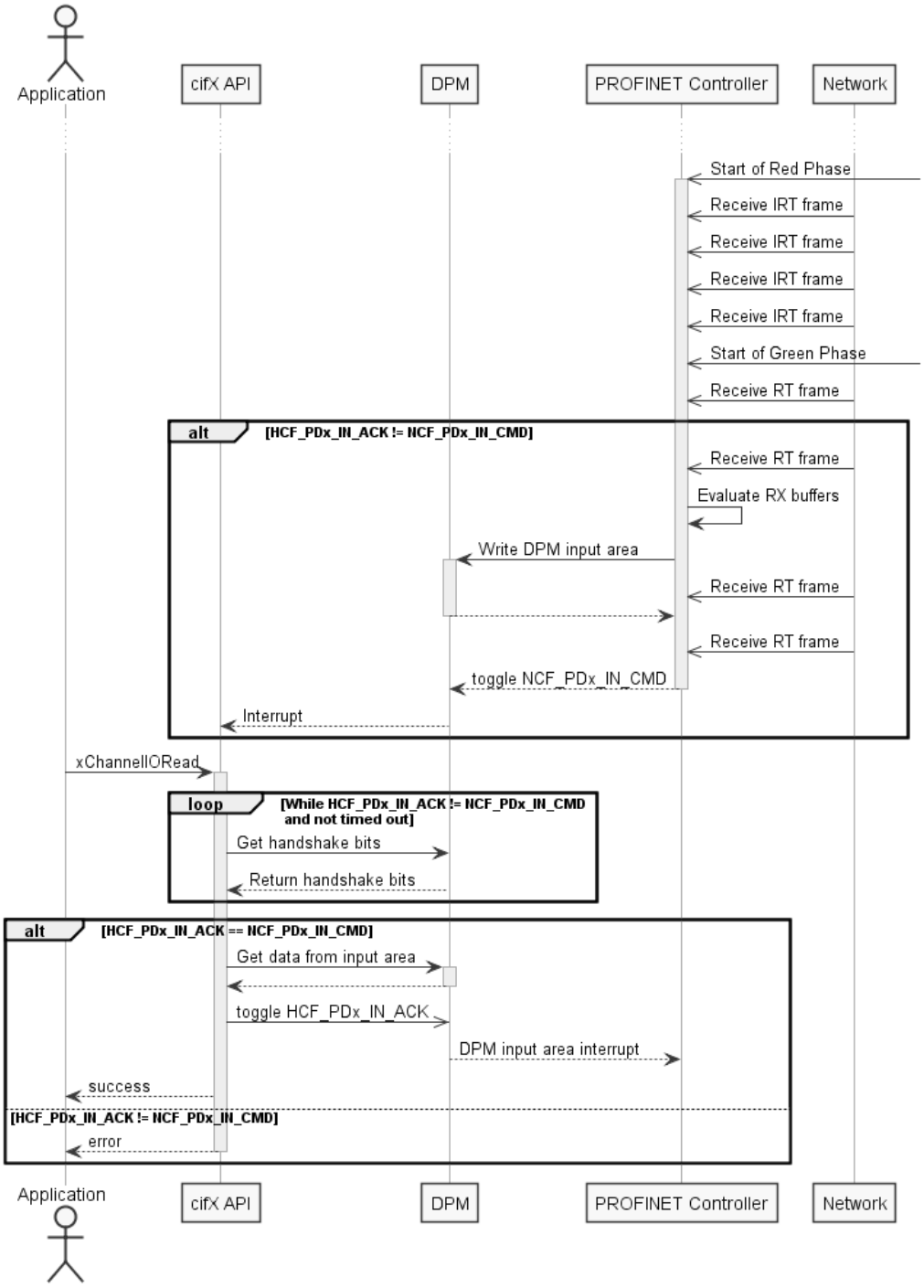

Figure 5: Sequence when using xChannelIORead

Note:	If using the function xChannelIORead it must be considered, that function copies the current content of the DPM input area to the application. The update of the DPM input area is requested after copying the data.

Note:	The DPM handling follows the buffered host controlled scheme. This means for the receive data, that the application has to call xChannelIORead initially in order to toggle the HCF_PDx_IN_ACK if the application uses interrupt mode. Otherwise no initial interrupt will be generated.

© Hilscher, 2015–2017

2.2.3 Process data timing in isochronous applications

For isochronous applications timing constraints apply to the PROFINET devices, the PROFINET Controller Firmware and the Host application. These timings are calculated by the engineering system and are included in the configuration of the PROFINET controller firmware.

Note: This feature is available since version V3.2.0.0

They following figure shows the timing relations for this use case. Note that in isochronous applications each network cycle consist of a red and a green period. The isochronous data is exchanged between the PROFINET Controller and Devices in the red period. (Realtime Class 3 traffic)

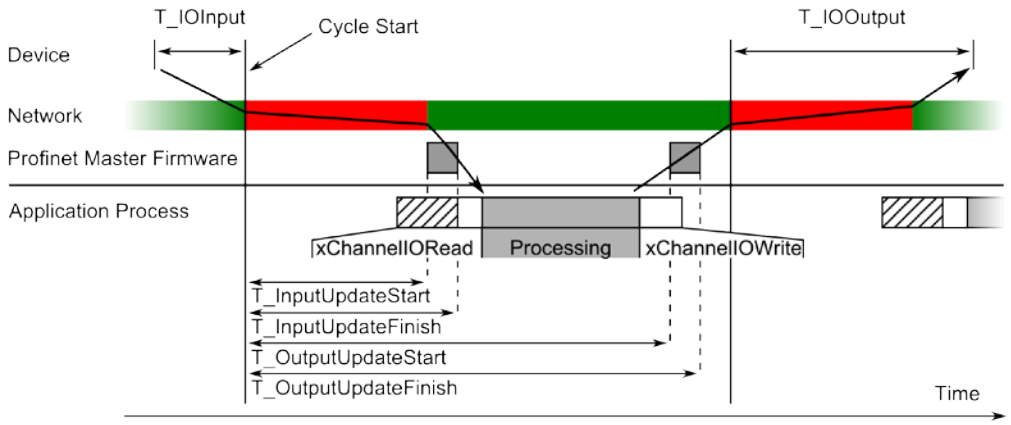

Figure 6: Process data timing

In isochronous applications, the PROFINET Devices are required to aquire the input values at a time point T_IOInput before the next cycle starts. Likewise the output values are to be applied at time T_IOOutput after the cycle start. While the T_IOInput time only depends on the capabilities of a particular device the T_IOOutput value also depends on length of the red period. The length of the red period in turn depends on the number of devices in the configuration, the amount of process data to be exchanged and the network topology.

The length of the red period also influences the processing time available for the PROFINET Controller Application. As shown in the image, right after the red period ends, the controller firmware will transfer the input process data into the Dual Port Memory. Afterwards the application can access and process the data. Finally the PROFINET Controller Firmware will transfer the output process data into internal buffers ready for transmission in the next red period.

The PROFINET Controller implementation of the Dual Port Memory Process Data Handshake mechanism simplifies the application handling. Calling the cifX Api Function xChannelIORead will block until the process data of the current cycle is available.

The PROFINET Controller firmware implements a monitoring mechanism which takes timestamps at some critical points in this processing loop: T_InputUpdateStart, T_InputUpdateFinish, T_OutputUpdateStart and T_OutputUpdateFinish. This timing information is placed in the extend status block. Additionally the firmware can be configured to provide this information in the input area as well. In both places the timing information is synchronized with the DPM input area handshake.

In order to provide the PROFINET Controller Application with an estimation about the available processing time, the Configuration Parameter Isochronous Controller Data has been defined. It is an artifical Parameter for the Controller generated by the engineering system. (e.g. Sycon.NET) The Application can read this parameter using the Get Ioc Parameter Service.

© Hilscher, 2015–2017

2.3 Configuration / Network state

The state machine shown in the following figure describes the internal behaviour of the PROFINET IO Controller firmware regarding configuration and network state.

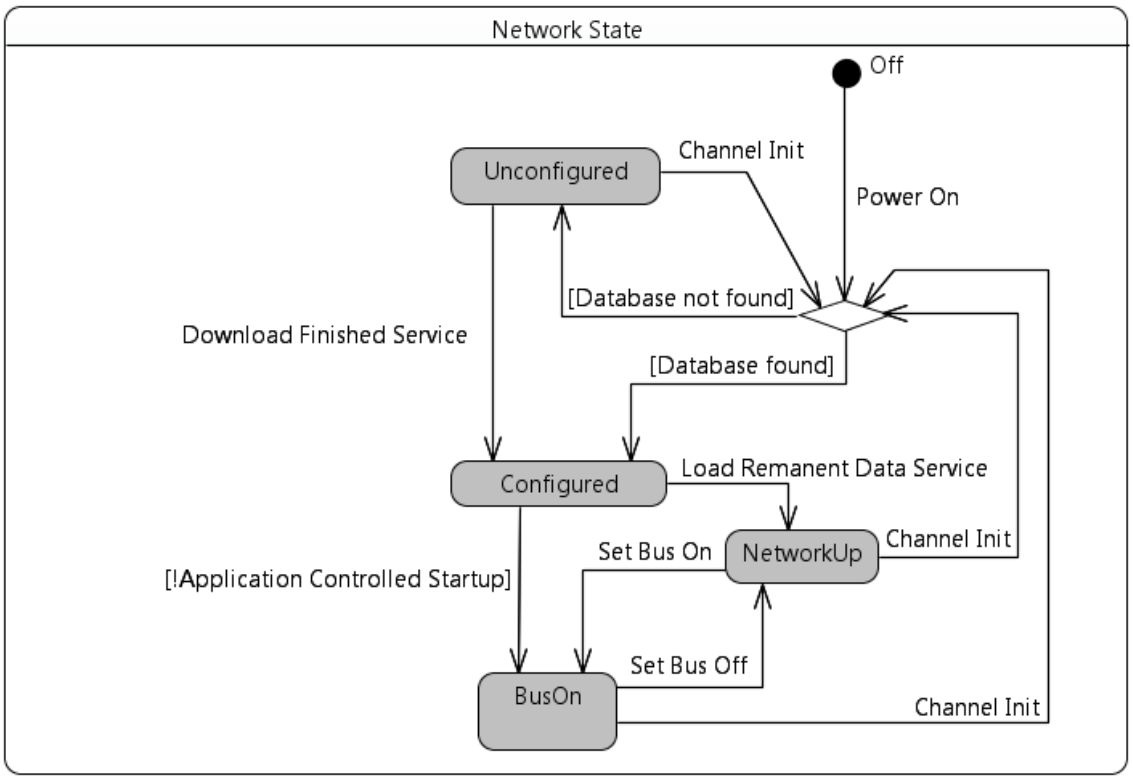

Figure 7 Controller Global/Network state

The following global PROFINET IO Controller states are defined:

- **Unconfigured**: No configuration is available. It will not react to DCP Protocol and establish no application relation with any slave
- **Configured**: A valid configuration was loaded. If Application Controlled Startup was set to false. The controller will continue automatically to state Bus On. The PROFINET IO Controller will not react to DCP Protocol in this state and will not establish any application relation.
- **NetworkUp:** DCP Protocol is enabled but no application relation is established. If DCP Set was enabled in configuration the controller will accept DCP SET service from network which changes either IP or Name Of Station. These parameters are stored as part of the remanent data by host application. When remanent data is restored using Load Remanent service, Name Of Station and IP Settings of Configuration are replaced with these values.
- **BusOn:** DCP Protocol is enabled and application relations are established. Any DCP SET service received from network will be rejected (except DCP Set Signal).

Note: Only a configuration using the setting “Application Controlled Startup” will meet the requirements of PROFINET IO Controller certification.

© Hilscher, 2015–2017

3 Application interface

This section contains the description of all packets of the application interface.

3.1 Configuration

This section contains the description of packets to configure the PROFINET IO Controller stack.

3.1.1 Configure OEM Parameter service

The Configure OEM Parameter service can be used by the application to configure various vendor specific parameters of the firmware. The usage of this service is optional but might be required for proper brand labeling.

Note: The service can only be used if bus communication is deactivated. It can also be performed after *Configure IO Controller service* (page 33) if the start mode is controlled by the application. In general, parameters affected by this service will not be reset to power-on values when the configuration is deleted. If this is desired a system reset or power cycle must be performed. Exception from this rule are parameters which overwrite configuration values.

3.1.1.1 PNM_AP_CFG_OEMPRM_REQ_T request

Packet structure reference

```
typedef struct PNM_AP_CFG_OEMPRM_VERSION_Ttag PNM_AP_CFG_OEMPRM_VERSION_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_OEMPRM_VERSION_Ttag
{
  /** structure version of this structure */
  uint32_t ulStructVersion;
  /** the hardware version number */
  uint16_t usHWVersion;
  /** the software version prefix */
  uint8_t  bPrefix;
  /** padding */
  uint8_t  bPadding;
  /** the major version number */
  uint16_t usVersionMajor;
  /** the minor version number */
  uint16_t usVersionMinor;
  /** the revision version number */
  uint16_t usVersionBugfix;
};

typedef struct PNM_AP_CFG_OEMPRM_SERIAL_NUMBER_Ttag PNM_AP_CFG_OEMPRM_SERIAL_NUMBER_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_OEMPRM_SERIAL_NUMBER_Ttag
{
  /** structure version of this structure */
  uint32_t ulStructVersion;
  /** serial number */
  uint8_t  abSerialNumber[16];
};

typedef struct PNM_AP_CFG_OEMPRM_ORDER_ID_Ttag PNM_AP_CFG_OEMPRM_ORDER_ID_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_OEMPRM_ORDER_ID_Ttag
{
  /** structure version of this structure */
  uint32_t ulStructVersion;
  /** Order ID */
```


© Hilscher, 2015–2017

```
  uint8_t  abOrderID[20];
};

typedef struct PNM_AP_CFG_OEMPRM_ALARM_HANDLING_Ttag PNM_AP_CFG_OEMPRM_ALARM_HANDLING_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_OEMPRM_ALARM_HANDLING_Ttag
{
  /** structure version of this structure */
  uint32_t ulStructVersion;
  /** Automatic Alarm handling flags
   *  The definition of the ulAlarmHandlingFlags are \ref
PNM_AP_CFG_ALARM_HANDLING_FLAGS_Etag enumeration. */
  uint32_t ulAlarmHandlingFlags;
};

typedef struct PNM_AP_CFG_OEMPRM_DEVICE_IDENTITY_Ttag
PNM_AP_CFG_OEMPRM_DEVICE_IDENTITY_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_OEMPRM_DEVICE_IDENTITY_Ttag
{
  /** Version of this structure */
  uint32_t ulStructVersion;
  /** Vendor Id */
  uint16_t usVendorId;
  /** Device Id */
  uint16_t usDeviceId;
  /** Device Instance */
  uint16_t usInstance;
};

enum PNM_AP_CFG_OEMPRM_TYPE_Etag
{
  PNM_AP_CFG_OEMPRM_VERSION                =  1,
  PNM_AP_CFG_OEMPRM_SERIAL_NUMBER          =  2,
  PNM_AP_CFG_OEMPRM_ORDER_ID               =  3,
  PNM_AP_CFG_OEMPRM_ALARM_HANDLING         =  4
  PNM_AP_CFG_OEMPRM_DEVICE_IDENTITY        =  5,
};

typedef enum PNM_AP_CFG_OEMPRM_TYPE_Etag PNM_AP_CFG_OEMPRM_TYPE_E;
/* parameter union */
typedef union PNM_AP_CFG_OEMPRM_UNION_Ttag PNM_AP_CFG_OEMPRM_UNION_T;
union PNM_AP_CFG_OEMPRM_UNION_Ttag
{
  PNM_AP_CFG_OEMPRM_VERSION_T             tVersion;
  PNM_AP_CFG_OEMPRM_SERIAL_NUMBER_T       tSerialNumber;
  PNM_AP_CFG_OEMPRM_ORDER_ID_T            tOrderId;
  PNM_AP_CFG_OEMPRM_ALARM_HANDLING_T      tAlarmHandling;
  PNM_AP_CFG_OEMPRM_DEVICE_IDENTITY_T     tDeviceIdentity;
};

/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_OEMPRM_STRUCT_VERSION_1    (0x0001)

/** request packet data */
typedef struct PNM_AP_CFG_OEMPRM_DATA_Ttag PNM_AP_CFG_OEMPRM_DATA_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_OEMPRM_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t                  ulStructVersion;

  /** identifier see \ref PNM_AP_CFG_PRM_TYPE_E */
  uint16_t                  usPrmType;
  /** Reserved (Padding) */
  uint16_t                  usPadding;
  /** parameter data */
  PNM_AP_CFG_OEMPRM_UNION_T  uData;
} ;
```


© Hilscher, 2015–2017

```
/** request packet */
typedef struct PNM_AP_CFG_OEMPRM_REQ_Ttag PNM_AP_CFG_OEMPRM_REQ_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_OEMPRM_REQ_Ttag
{
  /** packet header */
  PNM_AP_PCK_HEADER_T                    tHead;
  /** packet data */
  PNM_AP_CFG_OEMPRM_DATA_T               tData;
};

/** confirmation packet */
typedef PNM_AP_EMTPY_PCK_T            PNM_AP_CFG_OEMPRM_CNF_T;

/** packet union */
typedef union PNM_AP_CFG_OEMPRM_PCK_Ttag PNM_AP_CFG_OEMPRM_PCK_T;

union PNM_AP_CFG_OEMPRM_PCK_Ttag
{
  /** request packet */
  PNM_AP_CFG_OEMPRM_REQ_T    tReq;
  /** confirmation packet */
  PNM_AP_CFG_OEMPRM_CNF_T    tCnf;
};
```

Packet description

Structure PNM_AP_CFG_OEMPRM_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Ignored. Set to zero for future compatibility.
ulSrcId	UINT32	any	-
ulLen	UINT32	8 + n	Packet data length in bytes. n depends on the OEM parameter type contained in the packet. (see)
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x9412	PNM_AP_CMD_CFG_OEMPRM_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	any	-
Structure PNM_AP_CFG_OEMPRM_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usPrmType	UINT16	1	The the parameter to configure
usPadding	UINT16	0	Padding. Set to zero for future compatibility
uData	UNION		Union container for parameters

Parameter descriptions

Parameter ulStructVersion

Version of this structure. This parameter is used for future extensions of the Configure Controller service.

© Hilscher, 2015–2017

Parameter usPrmType

The usPrmType field is used to specify the parameter to configure. The following value is defined:

Option	Numeric value	Parameter Length (n)	Meaning
PNM_AP_CFG_OEMPRM_VERSION	1	14	The version identification of the firmware shall be configured. The field uData contains version information according structure PNM_AP_CFG_OEMPRM_VERSION_T.
PNM_AP_CFG_OEMPRM_SERIAL_ NUMBER	2	20	The Device unique serial number to be used. The field uData contains serial number according structure PNM_AP_CFG_OEMPRM_SERIAL_NUMBER_T.
PNM_AP_CFG_OEMPRM_ORDER_ID	3	24	The Order number of device. The field uData contains order number according structure PNM_AP_CFG_OEMPRM_SERIAL_NUMBER_T. This information is overwritten when a new configuration is loaded. Therfore this parameter must be set after configuring the profinet IO controller but before enabling the bus.
PNM_AP_CFG_OEMPRM_ALARM_ HANDLING	4	8	Defines alarm handling behavior by PROFINET IO Controller stack. This service allows the application to selct which alarm types can by the application be handeled. The field uData contains alarm handling flags according structure PNM_AP_CFG_ OEMPRM_ALARM_HANDLING_T. This parameter is overwritten when a new configuration is loaded. Therefore is must be set after configuring the profinet IO controller but before enabling the bus.
PNM_AP_CFG_OEMPRM_DEVICE_ID ENTITY	5	10	Overwrite Profinet VendorId, DeviceId and Instance from Configuration Database with custom values. This parameter is overwritten when a new configuration is loaded. Therefore is must be set after configuring the profinet IO controller but before enabling the bus.

Table 8: Definiton of OEM parameter types

Parameter uData

This field contains the configuration parameter. The union PNM_AP_CFG_OEMPRM_UNION_T with the following subfields is defined:

Variable	Type	Value / Range	Description
tVersion	STRUCT	-	Parameter structure for Version
tSerialNumber	STRUCT	-	Parameter structure for serial number.
tOrderID	STRUCT	-	Parameter structure for order number.
tAlarmHandling	STRUCT	-	Parameter structure for alaram handling flags.
tDeviceIdentity	STRUCT	-	Parameter structure for device identity.

These substructures are defined as follows

Parameter uData.tVersion

This parameter defines the version information to be reported by the protocol stack.

© Hilscher, 2015–2017

Note: The version information will be initialized with the Hilscher version number on startup of the firmware.

© Hilscher, 2015–2017

Variable	Type	Value / Range	Description
ulStructVersion	UINT32	1	Version of the structure
usHWVersion	UINT16	0 to 65535	Hardware revision
bPrefix	UINT8	'V', 'R', 'P', 'U', 'T'	Indicator if the device is an officially released version ('V'), an revised version ('R'), a prototype ('P'), a device for field test ('U') or a test device ('T')
bPadding	UINT8	0	Padding. Set to zero for future compatibility
usVersionMajor	UINT16	0 to 999	Major version of the product
usVersionMinor	UINT16	0 to 999	Minor version of the product
usVersionBugfix	UINT16	0 to 999	Bugfix version of the product

Note: If the firmware version of the PROFINET IO Controller is not a release or hotfix version and the application specifies a value of 'V' or 'R' for the prefix, then the firmware will force the prefix to the value 'P'.

Parameter uData.tSerialNumber

This parameter defines the unique production serial number. This serial number will be used for RPCAnnotation, LLDP System Description and LLDP ChassisId in case of enabled Multiple interface node.

Variable	Type	Value / Range	Description
ulStructVersion	UINT32	1	Version of the structure
abSerialNumber[16]	UINT8[]	0x20 – 0x7E	Serial number coded as visisble character

Note: The serial number shall be left aligned and shall be padded with space(es) (character 0x20) to the full field length

Parameter uData.tOrderId

This Parameter defines the order number that allows a unambiguous identification of the PROFINET IO Controller.

Note: This service is intended to be used only in conjunction with SYCON.net database configuration. The application which use Configure IO Controller service may set the OrderId using abOrderId parameter in PNM_AP_CFG_IOC_REQ_T request packet.

Variable	Type	Value / Range	Description
ulStructVersion	UINT32	1	Version of the structure
abOrderId[20]	UINT8[]	0x20 – 0x7E	Order number coded as visisble character

© Hilscher, 2015–2017

Parameter uData.tAlarmHandling

This parameter specifies which alarms shall be handled by the PROFINET IO Controller automatically and which alarms shall be passed to the application for processing.

Note: This service is intended to be used only in conjunction with SYCON.net database configuration. The application which use *Configure IO Controller service* may deterimine the alarm handling behavior of IO Controller using **ulAlarmHandlingFlags** parameter in PNM_AP_CFG_IOC_REQ_T request packet.

Variable	Type	Value / Range	Description
ulStructVersion	UINT32	1	Version of the structure
ulAlarmHandlingFlags	UINT32	-	Alarm flags coded as a bitmask where each bit corresponds to a particular alarm type. See *Table 9: Definition of alarm handling flags*

Parameter uData.tDeviceIdentity

This parameter set the PROFINET IO VendorId, DeviceId and Instance to the specified values. It can be used to overwrite values from a configuration database.

Note: This service is intended to be used only in conjunction with SYCON.net database configuration. The application which use *Configure IO Controller service* may set the device identity using **usVendorId**, **usDeviceId** parameter in PNM_AP_CFG_IOC_REQ_T request packet.

Variable	Type	Value / Range	Description
ulStructVersion	UINT32	1	Version of the structure
usVendorId	UINT16	1-65535	PROFINET IO Vendor Id to use.
usDeviceId	UINT16	-	PROFINET IO Device Id to use.
usInstance	UINT16	-	PROFINET IO Instance to use. Defaults to 0.

3.1.1.2 PNM_AP_CFG_OEMPRM_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T          PNM_AP_CFG_OEMPRM_CNF_T;
```

Packet description

Structure PNM_AP_CFG_OEMPRM_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	
ulSrc	UINT32	0 ... 2^{32}-1	Ignore
ulDestId	UINT32	0	
ulSrcId	UINT32	mirrored	
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	mirrored	

© Hilscher, 2015–2017

ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x9413	PNM_AP_CMD_CFG_OEMPRM_CNF
ulExt	UINT32	0	
ulRoute	UINT32		Ignore

© Hilscher, 2015–2017

3.1.2 Configure IO Controller service

This service is used to configure basic parameters of the PROFINET IO Controller. The service shall be used exactly once at the beginning of the configuration sequence. The service cannot be used if bus communication is activated or if the configuration is locked.

3.1.2.1 PNM_AP_CFG_IOC_REQ_T request

Packet structure reference

```
enum PNM_AP_CFG_STARTMODE_FLAGS_Etag
{
  /** CONTROLER will start with the data exchange on the bus after the initialization has
been ended. */
  PNM_AP_IOC_FLAG_STARTMODE_AUTOMATIC              = 0x00000000,
  /** Application program must activate the data exchange on the bus. */
  PNM_AP_IOC_FLAG_STARTMODE_APPLICATION_CONTROLLED = 0x00000001,
  /** Controller will force devices to set the configured name of station even if a non
empty nameOfStation is set.
   * Please note that this feature is only available if TOPOLOGY INFORMATION is set using
PNM_AP_CFG_TOPO_DATA_T */
  PNM_AP_IOC_FLAG_FORCE_NAME_ASSIGNMENT            = 0x00000002,
  /** controller will accept name of station / ip address changed from bus
   * if not in state operate
   */
  PNM_AP_IOC_FLAG_ENABLE_DCP_SET                   = 0x00000004,
};

typedef enum PNM_AP_CFG_STARTMODE_FLAGS_Etag PNM_AP_CFG_STARTMODE_FLAGS_E;

enum PNM_AP_CFG_ALARM_HANDLING_FLAGS_Etag
{
  /* includes diagnosis disappears */
  PNM_AP_CFG_ALARM_HANDLING_FLAG_DIAGNOSIS           = 0x00000001,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_PROCESS             = 0x00000002,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_PULL                = 0x00000004,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_PLUG                = 0x00000008,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_PULL_MODULE         = 0x00000010,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_PLUG_WRONG          = 0x00000020,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_STATUS              = 0x00000040,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_UPDATE              = 0x00000080,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_MEDIA_REDUND        = 0x00000100,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_CONTROLLED          = 0x00000200,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_RELEASED            = 0x00000400,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_RETURN_OF_SUBM      = 0x00000800,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_MCR_MISMATCH        = 0x00001000,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_PORT_DATA_CHANGE    = 0x00002000,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_SYNC_DATA_CHANGE    = 0x00004000,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_TIME_DATA_CHANGE    = 0x00008000,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_ISOCHR_MODE_PROBLEM = 0x00010000,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_NETW_COMP_PROBLEM   = 0x00020000,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_DFP_PROBLEM         = 0x00040000,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_MRPD_PROBLEM        = 0x00080000,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_MULT_INTF_MISMATCH  = 0x00100000,
  PNM_AP_CFG_ALARM_HANDLING_FLAG_UPLOAD_AND_RETRIVAL = 0x00200000,
  /* unknown alarm types MUST be handled in application and are NOT handled automatically
by protocol stack */
  PNM_AP_CFG_ALARM_HANDLING_FLAG_MANUFACTURER_ALARMS = 0x80000000,
};

/** valid values for DPM watchdog time */
#define PNM_AP_DPM_WATCHDOG_TIME_OFF     0x00000000L /* Watchdog supervision disabled
*/
#define PNM_AP_DPM_WATCHDOG_TIME_MIN     0x00000014L /* Minimum value for watchdog
supervision */
#define PNM_AP_DPM_WATCHDOG_TIME_MAX     0x0000ffffL /* Maximum value for watchdog
supervision */
```


© Hilscher, 2015–2017

```
typedef enum PNM_AP_CFG_ALARM_HANDLING_FLAGS_Etag PNM_AP_CFG_ALARM_HANDLING_FLAGS_E;

/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_IOC_STRUCT_VERSION_1    (0x0001)
#define PNM_AP_CFG_IOC_STRUCT_VERSION_2    (0x0002)

/* request packet data */
typedef struct PNM_AP_CFG_IOC_DATA_Ttag PNM_AP_CFG_IOC_DATA_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_IOC_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t              ulStructVersion;
  /** The definition of the ulSystemFlags are \ref PNM_AP_CFG_STARTMODE_FLAGS_Etag */
  uint32_t              ulSystemFlags;
  /** Automatic Alarm handling flags

      With the ulAlarmHandlingFlags you can define which alarms should be handled by the
fieldbus stack automatically.
      The definition of the ulAlarmHandlingFlags are \ref
PNM_AP_CFG_ALARM_HANDLING_FLAGS_Etag enumeration.
  */
  uint32_t              ulAlarmHandlingFlags;
  /** DPM Channel Watchdog time in ms

    \arg \c Deactivated 0
    \arg \c Min 0
    \arg \c Max 65535)
  */
  uint32_t              ulDpmWatchdogTime;
  /** VendorID to be used by IO Controller
        \note This parameter is for the OEMs to handover the own VendorID number.
        \par See Spec:
        Coding of fields related to Instance, DeviceID, VendorID
  */
  uint16_t              usVendorID;
  /** DeviceID to be used by IO Controller
        \note This parameter is for the OEMs to handover the Vendor spezific DeviceID
number.
        \par See Spec:
        Coding of fields related to Instance, DeviceID, VendorID
  */
  uint16_t              usDeviceID;
  /** IP address to be used by IO Controller */
  uint32_t              ulIPAddr;
  /** subnet mask to be used by IO Controller */
  uint32_t              ulNetmask;
  /** Gateway address to be used by IO Controller */
  uint32_t              ulGateway;
  /** Device Type to be used by IO Controller, Pad With Zero:
      we don't need a length here, can be computed by searching for zeros
      \par See Spec:
       See Coding of the field DeviceType
  */
  uint8_t               abDeviceType[25];
  /** NameOfStation to be used by IO Controller, Pad With Zero:
      we don't need a length here, can be computed by searching for zeros
      \note The default name should be get from the GSDML attribute \c
DNS_CompatibleName.
      \par See Spec:
       Coding of the field NameOfStationValue
  */
  uint8_t               abNameOfStation[240];
  /** OrderId to be used by IO Controller, Pad With Zero:
      we don't need a length here, can be computed by searching for zeros
      \note  The GSDML Element \c OrderNumber.
      \par See Spec:
       Coding of the field OrderID
  */
```


© Hilscher, 2015–2017

```
  uint8_t              abOrderId[20];

  /** Start offset in DPM input area for slave status bitlists.
      Each bitlist has 128 bits (one for each IO Device) thus requires 16 byte.
      The bitlists are always in the fix order: configured, active, faulty

      The bitlist must be located outside of the process data area, that is,
      either before any process data or behind.
  */
  uint16_t             usBitlistStartOffset;
  /** Fix alignment of structure fields due to definition abDeviceType */
  uint8_t              bPadding;
  /** Start offset in DPM input area for the IRT cycle counter
   *
   * This field is available since structure version 2. Set to 0xFFFF
   * to disable cycle counter in input data area. */
  uint16_t             usIoTimingInfoOffset;
};

/* request packet */
typedef struct PNM_AP_CFG_IOC_REQ_Ttag PNM_AP_CFG_IOC_REQ_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_IOC_REQ_Ttag
{
  /** packet header */
  PNM_AP_PCK_HEADER_T              tHead;
  /** packet data */
  PNM_AP_CFG_IOC_DATA_T            tData;
} ;

/** confirmation packet */
typedef PNM_AP_EMTPY_PCK_T         PNM_AP_CFG_IOC_CNF_T;

/** packet union */
typedef union PNM_AP_CFG_IOC_PCK_Ttag PNM_AP_CFG_IOC_PCK_T;
union PNM_AP_CFG_IOC_PCK_Ttag
{
  /** request packet */
  PNM_AP_CFG_IOC_REQ_T      tReq;
  /** confirmation packet */
  PNM_AP_CFG_IOC_CNF_T      tCnf;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_CFG_IOC_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Ignored. Set to zero for future compatibility.
ulSrcId	UINT32	any	-
ulLen	UINT32	322	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x9400	PNM_AP_CMD_CFG_IOC_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	any	-
Structure PNM_AP_CFG_IOC_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
ulSystemFlags	UINT32	0, 1	System flags
ulAlarmHandlingFlags	UINT32		Automatic alarm handling flags
ulDpmWatchdogTime	UINT32	0, 20 … 65535	DPM channel watchdog time
usVendorID	UINT16		VendorID to be used by IO Controller
usDeviceID	UINT16		DeviceID to be used by IO Controller
ulIPAddr	UINT32		IP address to be used by IO Controller
ulNetmask	UINT32		Subnet mask to be used by IO Controller
ulGateway	UINT32		Gateway address to be used by IO Controller
abDeviceType[25]	UINT8[]		Device Type to be used by IO Controller
abNameOfStation[240]	UINT8[]		NameOfStation to be used by IO Controller
abOrderId[20]	UINT8[]		OrderId to be used by IO Controller
usBitlistStartOffset	UINT16		Start offset in DPM input area for slave status bit lists
bPadding	UINT8	0	Padding to fix structure field alignment. Set to zero for future compatibility.
usIoTimingInfoOffset	UINT16		Start offset in DPM input area for io timing information (This field is available since structure version 2)

© Hilscher, 2015–2017

Parameter descriptions

Parameter ulStructVersion

Version of this structure. This parameter is used for future extensions of the Configure Controller Service.

Parameter ulSystemFlags

The ulSystemFlags field is a bitmask of flags defined as follows:

Option	Numeric value	Meaning
PNM_AP_IOC_FLAG_STARTMODE_AUTOMATIC	0x00000000	The communication will be started automatically after the PROFINET IO Controller had been configured successfully.
PNM_AP_IOC_FLAG_STARTMODE_APPLICATION_CONTROLLED	0x00000001	The application must explicitly enable the bus to start communication.
PNM_AP_IOC_FLAG_FORCE_NAME_ASSIGNMENT	0x00000002	The controller will assign the Name Of Station even to devices which are already assigned a Name Of Station.

Parameter ulAlarmHandlingFlags

This parameter specifies which alarms shall be handled by the PROFINET IO Controller itself and which alarms shall be passed to the application for processing. The field ulAlarmHandlingFlags is defined as a bitmask where each bit corresponds to a particular alarm type. For each alarm to be handled by the PROFINET IO Controller the corresponding bit shall be set. As a special use case it is also possible to handle all manufacturer specific alarms within the PROFINET IO Controller using the bitmask PNM_AP_CFG_ALARM_HANDLING_FLAG_MANUFACTURER_ALARMS.

The following bitmasks are defined:

Flag	Value
PNM_AP_CFG_ALARM_HANDLING_FLAG_DIAGNOSIS	0x00000001
PNM_AP_CFG_ALARM_HANDLING_FLAG_PROCESS	0x00000002
PNM_AP_CFG_ALARM_HANDLING_FLAG_PULL	0x00000004
PNM_AP_CFG_ALARM_HANDLING_FLAG_PLUG	0x00000008
PNM_AP_CFG_ALARM_HANDLING_FLAG_PULL_MODULE	0x00000010
PNM_AP_CFG_ALARM_HANDLING_FLAG_PLUG_WRONG	0x00000020
PNM_AP_CFG_ALARM_HANDLING_FLAG_STATUS	0x00000040
PNM_AP_CFG_ALARM_HANDLING_FLAG_UPDATE	0x00000080
PNM_AP_CFG_ALARM_HANDLING_FLAG_MEDIA_REDUND	0x00000100
PNM_AP_CFG_ALARM_HANDLING_FLAG_CONTROLLED	0x00000200
PNM_AP_CFG_ALARM_HANDLING_FLAG_RELEASED	0x00000400
PNM_AP_CFG_ALARM_HANDLING_FLAG_RETURN_OF_SUBM	0x00000800
PNM_AP_CFG_ALARM_HANDLING_FLAG_MCR_MISMATCH	0x00001000
PNM_AP_CFG_ALARM_HANDLING_FLAG_PORT_DATA_CHANGE	0x00002000
PNM_AP_CFG_ALARM_HANDLING_FLAG_SYNC_DATA_CHANGE	0x00004000
PNM_AP_CFG_ALARM_HANDLING_FLAG_TIME_DATA_CHANGE	0x00008000
PNM_AP_CFG_ALARM_HANDLING_FLAG_ISOCHR_MODE_PROBLEM	0x00010000
PNM_AP_CFG_ALARM_HANDLING_FLAG_NETW_COMP_PROBLEM	0x00020000
PNM_AP_CFG_ALARM_HANDLING_FLAG_DFP_PROBLEM	0x00040000
PNM_AP_CFG_ALARM_HANDLING_FLAG_MRPD_PROBLEM	0x00080000
PNM_AP_CFG_ALARM_HANDLING_FLAG_MULT_INTF_MISMATCH	0x00100000
PNM_AP_CFG_ALARM_HANDLING_FLAG_UPLOAD_AND_RETRIVAL	0x00200000

© Hilscher, 2015–2017

Flag	Value
PNM_AP_CFG_ALARM_HANDLING_FLAG_MANUFACTURER_ALARMS	0x80000000

Table 9: Definition of alarm handling flags

Parameter `ulDpmWatchdogTime`

This parameter specifies the DPM channel watchdog timeout in units of milliseconds.

- A value of 0 means DPM Channel Watchdog supervision is explicitly disabled.
- A value between 20 (Minimum) and 65535 (Maximum) means DPM Channel Watchdog supervision is prepared with the specified watchdog time. In order to activate the watchdog, the application must start the supervision using the standard DPM Watchdog Mechanism after the successful configuration of the PROFINET IO Controller.

Parameter `usVendorID`

This parameter specifies the PROFINET Vendor Id the controller shall use. The vendor id is an unique identification number of the controller's vendor assigned by the PNO.

Note: This parameter is for the OEMs to handover the own VendorID number.

Parameter `usDeviceID`

This parameter specifies the PROFINET Device Id the controller shall use. The device id is a unique device id assigned by the vendor of the device.

Note: This parameter is for the OEMs to handover the Vendor specific DeviceID number.

Parameter `ulIPAddr`

IP address to be used by IO Controller

Parameter `ulNetmask`

Subnet mask to be used by IO Controller

Parameter `ulGateway`

Gateway address to be used by IO Controller

Parameter `abDeviceType`

The device type is a short textual description of the device. It is used in device identification when performing network scans or when reading out identification information from a PROFINET Device or Controller. The value shall be padded with zero(es) to the full field length.

© Hilscher, 2015–2017

Parameter `abNameOfStation`

The name of station is the bus address of a PROFINET Device or Controller. The name of station must be unique across the communication network. According to the PROFINET specification the characters a-z, 0-9, '-' and '.' are allowed where '.' has the special meaning of a label separator.. The name must fulfill further conditions:

- it must not look like an IP address, e.g. 1.2.3.4
- a label must consist of at least one and at most 63 characters
- the name must not start with 'port-abc' or 'port-abc-defgh' where a,b,c,d,e,f,g,h are digits 0-9

The field shall be padded with zero(es) to full field length

Parameter abOrderId

The vendor's order id of the PROFINET IO Controller. The field shall be padded with zero(es) to full field length.

Parameter `usBitlistStartOffset`

The PROFINET IO Controller manages three bit lists within the DPM Input Area. Each bit list is 128 bits (=16 byte) wide and reflects the status when the DPM Input area was updated by the PROFINET IO Controller. The parameter `usBitlistStartOffset` specifies the offset in byte from the beginning of the DPM Input Area where these lists shall placed to. The lists are always present and cannot be deactivated. The following bit lists are defined:

Name	Offset in DPM Input Area	Description
Configured Bitlist	usBitListStartOffset	A bit is set to true if associated AR had been configured
Active Bitlist	usBitListStartOffset + 16	A bit is set to true if associated AR is communicating
Diagnosis	usBitListStartOffset + 32	A bit is set to true if associated AR is communicating and either a diagnosis is pending or a configuration difference was detected

The offset of the bit corresponding to a particular AR can be derived from the device handle as follows:

```
ByteOffset =         (usDeviceHandle - 1) / 8
BitMask    = 0x1 << ((usDeviceHandle -1 ) % 8)
```

Note: The bit list area and the input IOCR data are placed within the DPM Input Area. Thus the application must ensure that the bit lists memory area does not overlap with input IOCR data blocks.

Parameter `usIoTimingInfoOffset`

The PROFINET IO Controller firmware provides timing information about process data handling using the PNM_AP_IOTIMINGINFO_T structure. This information can be placed in DPM input area if required. This parameter configures the start offset of this data within the DPM input area. If the offset is larger than the size of the DPM input area or the data does not fully fit into the DPM input area the feature is disabled. Thus using a value 0xFFFF will disable this feature.

Note: This information is also provided in the DPM extended status block

© Hilscher, 2015–2017

3.1.2.2 PNM_AP_CFG_IOC_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T          PNM_AP_CFG_IOC_CNF_T;
```

Packet description

Structure PNM_AP_CFG_IOC_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		
ulSrc	UINT32		Ignore
ulDestId	UINT32	0	
ulSrcId	UINT32	mirrored	
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x9401	PNM_AP_CMD_CFG_IOC_CNF
ulExt	UINT32	0	
ulRoute	UINT32		Ignore

© Hilscher, 2015–2017

3.1.3 Configure IO Controller parameter service

This service can be used to configure parameters of the PROFINET IO Controller.

3.1.3.1 PNM_AP_CFG_IOC_PRM_REQ_T request

Packet structure reference

```
typedef union PNM_AP_CFG_IOC_PRM_UNION_Ttag PNM_AP_CFG_IOC_PRM_UNION_T;
union PNM_AP_CFG_IOC_PRM_UNION_Ttag
{
  PNM_AP_CFG_PRM_PDINTERFACEADJUST_T         tIntfAdjust;
  PNM_AP_CFG_PRM_SYNC_T                      tSyncData;
  PNM_AP_CFG_PRM_IRDATA_GLOBAL_T             tIrtGlobal;
  PNM_AP_CFG_PRM_IRDATA_PHASES_T             tIrtPhases;
  PNM_AP_CFG_PRM_IRDATA_FRAME_T              tIrtFrames;
  PNM_AP_CFG_PRM_PDINTFMRPDATACHECK_T        tIntfMRPDataCheck;
  PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST_T       tIntfMRPDataAdjust;
  PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST_T       tPortMRPDataAdjust;
  PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_T         tPortDataAdjust;
  PNM_AP_CFG_PRM_PDPORTDATA_CHECK_T          tPortDataCheck;
  PNM_AP_CFG_PRM_PDNCDATACHECK_T             tNCDataCheck;
  PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_T tIsochronousControllerData;
};

/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_IOC_PRM_STRUCT_VERSION_1    (0x0001)

/** request packet data */
typedef struct PNM_AP_CFG_IOC_PRM_DATA_Ttag PNM_AP_CFG_IOC_PRM_DATA_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_IOC_PRM_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t                  ulStructVersion;

  /** identifier see \ref PNM_AP_CFG_PRM_TYPE_E */
  uint16_t                  usPrmType;
  /** Port specifier.
   *
   * @details Some records are related to a specific port of the controller.
   * Allowed values: 0 = Interface, 1 = Port 0, 2 = Port 1 */
  uint8_t                    bPortId;
  /** Reserved (Padding) */
  uint8_t                    bPadding;
  /** parameter data */
  PNM_AP_CFG_IOC_PRM_UNION_T  uData;
} ;

/** request packet */
typedef struct PNM_AP_CFG_IOC_PRM_REQ_Ttag PNM_AP_CFG_IOC_PRM_REQ_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_IOC_PRM_REQ_Ttag
{
  /** packet header */
  PNM_AP_PCK_HEADER_T                    tHead;
  /** packet data */
  PNM_AP_CFG_IOC_PRM_DATA_T              tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_CFG_IOC_PRM_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	-
ulSrcId	UINT32	any	-
ulLen	UINT32		Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x9402	PNM_AP_CMD_CFG_IOC_PRM_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	any	-
Structure PNM_AP_CFG_IOC_PRM_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usPrmType	UINT16	1 to 8, 12, 13	Parameter type (identifier)
bPortId	UINT8	0 to 2	Reference of controller port to apply this parameters to
bPadding	UINT8	0	Padding. Set to zero for future compatibility.
uData	PNM_AP_CFG_IOC_PRM_UNION_T		Parameter data

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used for future extensions to this structure.

© Hilscher, 2015–2017

Parameter usPrmType, uData

The parameter usPrmType specifies the type of the parameter data within this request packet. This means in particular which field within uData is to be used.

Name	Numeric value	Union element	Packet length	Usage
PNM_AP_CFG_PRM_PDINTERFACEADJUST	13	tIntfAdjust	8 + 8	uData.tIntfAdjust contains a PD Interface Adjust Configuration to be written to an Interface Submodule. See *PNM_AP_CFG_PRM_PDINTERFACEADJUST_T structure* (page 110) for details.
PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST	8	tIntfMRPAdjust	8 + 4 + 276	uData.tIntfMRPDataAdjust contains a PD Interface MRP Data Adjust Configuration to be written to a Interface Submodule. See *PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST_T structure* (page 101) for details on the structure.
PNM_AP_CFG_PRM_PDINTFMRPDATACHECK	7	tIntfMRPCheck	8 + 4 + 28	uData.tIntfMRPDataCheck contains a PD Interface MRP Data Check Configuration to be written to a Interface Submodule. See *PNM_AP_CFG_PRM_PDINTFMRPDATACHECK_T structure* (page 99) for details on the structure
PNM_AP_CFG_PRM_PDIRDATA_PDIRBEGINENDDATA	5	tIrtPhases	4 + 16 * Number of Phases	uData.tIrtPhases contains a PD IR Data Adjust BeginEnd configuration to be written to a Port submodule. See *PNM_AP_CFG_PRM_IRDATA_PHASES_T structure* (page 95) for details on the structure. Note: For structural reasons the PNM_AP_CFG_PRM_IRDATA_PHASES_T is associated with a Port Submodule. Nevertheless will the parameter write occur on the associated interface submodule (as defined by PROFINET Specification).
PNM_AP_CFG_PRM_PDIRDATA_PDIRFRAMEDATA	4	tIrtFrames	8 + 36	uData.tIrtFrames contains a PD IR Data Adjust Frame configuration to be written to an Interface submodule. See *PNM_AP_CFG_PRM_IRDATA_FRAME_T structure* (page 93) for details on the structure.
PNM_AP_CFG_PRM_PDIRDATA_PDIRGLOBALDATA	3	tIrtGlobal	8 + 36 + 16 * 2	uData.tIrtGlobal contains a PD IR Data Adjust Global configuration to be written to the interface submodule of the PROFINET Controller Firmware. See *PNM_AP_CFG_PRM_IRDATA_GLOBAL_T structure* (page 90)for details on the structure.
PNM_AP_CFG_PRM_PDNCDATACHECK	12	tNCDataCheck	8 + 20	uData.tNCDataCheck contains a PD NC Data check configuration to be written to an Interface/Port Submodule. See *PNM_AP_CFG_PRM_PDNCDATACHECK_T structure* (page 108) for details on the structure.
PNM_AP_CFG_PRM_PDPORTDATAADJUST	2	tPortDataAdjust	8 + 36	uData.tPortDataAdjust contains a PD Port Data Adjust configuration to be written to a Port submodule. See *PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_T structure* (page 86) for details on the structure
PNM_AP_CFG_PRM_PDPORTDATACHECK	1	tPortDataCheck	8 + 272	uData.tPortDataCheck contains a PD Port Data Check configuration to be written to a Port submodule. See *PNM_AP_CFG_PRM_PDPORTDATA_CHECK_T structure* (page 84) for details on the structure

© Hilscher, 2015–2017

Name	Numeric value	Union element	Packet length	Usage
PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST	9	tPortMRPDataAdjust	8 + 20	uData.tPortMRPDataAdjust contains a PD Port MRP Data Adjust Configuration to be written to a Port Submodule. See *PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST_T structure* (page 104) for details on the structure.
PNM_AP_CFG_PRM_PDSYNCDATA	6	tSyncData	8 + 280	uData.tSyncData contains a Sync Data for SyncID 0 Configuration to be written to an Interface Submodule. See *PNM_AP_CFG_PRM_SYNC_T structure* (page 96) for details on the structure. Due to the limitations of the PROFINET Controller Firmware this parameter must describe a sync master configuration.
PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_T	17	tIsochronousControllerData	8 + 24	tIsochronousControllerData contains a Isochronous Controller Data parameter. See PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_T structure (page 116) for details on the structure.

Table 10: Parameter usPrmType

Parameter bPortId

This parameter defines to which PD instance the parameter shall be applied:

Numerical value	Meaning
0	Interface (submodule) of PROFINET IO Controller
1	Port 1 (submodule) of PROFINET IO Controller
2	Port 2 (submodule of PROFINET IO Controller

© Hilscher, 2015–2017

3.1.3.2 PNM_AP_CFG_IOC_PRM_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T           PNM_AP_CFG_IOC_PRM_CNF_T;
```

Packet description

Structure PNM_AP_CFG_IOC_PRM_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure TLR_PACKET_HEADER_T			
ulDest	UINT32		
ulSrc	UINT32		Ignore
ulDestId	UINT32	0	
ulSrcId	UINT32	mirrored	
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x9403	PNM_AP_CMD_CFG_IOC_PRM_CNF
ulExt	UINT32	0	
ulRoute	UINT32		Ignore

© Hilscher, 2015–2017

3.1.4 Configure IO Device service

This service is used by the application to configure the basic properties of an AR.

3.1.4.1 PNM_AP_CFG_IOD_REQ_T request

Packet structure reference

```
enum PNM_AP_CFG_IOD_FLAG_Etag
{
  /** the IO-Device supports WriteMultiple */
  PNM_AP_CFG_IOD_FLAG_WRITE_MULTIPLE_SUPPORTED = 0x01,
  /** IO Controller firmware shall not check parameters inside this request */
  PNM_AP_CFG_IOD_FLAG_DISABLE_PRM_CHECK        = 0x02,
  /** The AR is initially disabled. If this flag is set, the profinet controller firmware
   * will not establish this AR on startup */
  PNM_AP_CFG_IOD_FLAG_DISABLED                 = 0x04,

  PNM_AP_CFG_IOD_FLAG_MAX,
};

/** Device address modes */
enum PNM_AP_CFG_IOD_ADDRESS_MODE_Etag
{
  /** The controller identifies the device using its name of station
   * or alias name and will assign name of station and ip if required.
   *
   * This is the default operaiton mode as defined in specification */
  PNM_AP_CFG_IOD_ADDRESS_MODE_DCP_IDENTIFY_ASSIGN = 0x00,
  /** The controller discovers the device using the configured name of station
   * but does not configure the ipsuite. The controller will use the ip address
   * reported by the device to establish communication. The controller will
   * not establish communication if no ip address was preconfigured in the device
   */
  PNM_AP_CFG_IOD_ADDRESS_MODE_DCP_DISCOVER        = 0x01,
  /** The controller discovers the device using its ip address by means of arp
   * no dcp is sent
   */
  PNM_AP_CFG_IOD_ADDRESS_MODE_ARP_DISCOVER        = 0x02,
};

typedef enum PNM_AP_CFG_IOD_ADDRESS_MODE_Etag PNM_AP_CFG_IOD_ADDRESS_MODE_E;

typedef enum PNM_AP_CFG_IOD_FLAG_Etag PNM_AP_CFG_IOD_FLAG_E;

/* AR Types */
enum PNM_AP_CFG_IOD_AR_TYPE_Etag
{
  /* standard AR Type for RT communication */
  PNM_AP_CFG_IOD_AR_TYPE_SINGLE           = 0x01,
  /* IO Supervisor AR (with IO-data), not supported */
  PNM_AP_CFG_IOD_AR_TYPE_SUPERVISOR       = 0x06,
  /* standard AR Type for IRT communication */
  PNM_AP_CFG_IOD_AR_TYPE_RTC3             = 0x10,
  /* SystemRedundancy AR, not supported */
  PNM_AP_CFG_IOD_AR_TYPE_SYSTEMREDUNDANCY = 0x20,
};

typedef enum PNM_AP_CFG_IOD_AR_TYPE_Etag PNM_AP_CFG_IOD_AR_TYPE_E;

/* AR Properties */
enum PNM_AP_CFG_IOD_AR_PROP_FLAGS_Etag
{
  /* a supervisor is allowed to take over submodules of this AR */
  PNM_AP_CFG_IOD_AR_PROP_FLAG_SUPERVISOR_TAKEOVER_ALLOWED     = 0x00000008,
  /** startup according to Profinet specification 2.3, if not set the legacy startup is
used */
  PNM_AP_CFG_IOD_AR_PROP_FLAG_STARTUPMODE_ADVANCED            = 0x40000000,
```


© Hilscher, 2015–2017

```
  /* alarm type PULL MODULE is supported */
  PNM_AP_CFG_IOD_AR_PROP_FLAG_PULL_MODULE_ALARM_ALLOWED       = 0x80000000,
  /* AR is an device access AR, only valid in conjunction with
   * PNM_AP_CFG_IOD_AR_TYPE_SUPERVISOR */
  PNM_AP_CFG_IOD_AR_PROP_FLAG_DEVICEACCESS                    = 0x00000100,
};

typedef enum PNM_AP_CFG_IOD_AR_PROP_FLAGS_Etag PNM_AP_CFG_IOD_AR_PROP_FLAGS_E;

typedef uint16_t PNM_AP_DEVICEHANDLE_T;

enum PNM_AP_DEVICEHANDLE_Etag
{
  /** Device handle of the controller itself */
  PNM_AP_DEVICEHANDLE_IOC        = 0x0000,

  /** First valid devicehandle to use for IO AR */
  PNM_AP_DEVICEHANDLE_IOAR_FIRST = 0x0001,
  /** Last valid devicehandle to use for IO AR */
  PNM_AP_DEVICEHANDLE_IOAR_LAST  = 0x0100,

  /** First valid devicehandle to use for DA AR */
  PNM_AP_DEVICEHANDLE_DAAR_FIRST = 0xF000,
  /** Last valid devicehandle to use for DA AR */
  PNM_AP_DEVICEHANDLE_DAAR_LAST  = 0xF000,
};


/* RTA Retries */
#define PNM_AP_CFG_IOD_RTA_RETRIES_MIN            (0x0003)  /**< minimum value allowed
for RTA retries */
#define PNM_AP_CFG_IOD_RTA_RETRIES_MAX            (0x000F)  /**< maximum value allowed
for RTA retries */

/* RTA timeout factor */
#define PNM_AP_CFG_IOC_RTA_TIMEOUT_FACT_MIN       (0x0001)
#define PNM_AP_CFG_IOC_RTA_TIMEOUT_FACT_MAX       (0x0064)

/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_IOD_STRUCT_VERSION_1    (0x0001)

/** request packet data */
typedef struct PNM_AP_CFG_IOD_REQ_DATA_Ttag PNM_AP_CFG_IOD_REQ_DATA_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_IOD_REQ_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t              ulStructVersion;
  /** unique handle of this IO-Device defined by the sender of this packet */
  PNM_AP_DEVICEHANDLE_T usDeviceHandle;
  /** AR Type see \ref PNM_AP_CFG_IOD_AR_TYPE_E */
  uint8_t               bARType;
  /** Address mode see \ref PNM_AP_CFG_IOD_ADDRESS_MODE_E */
  uint8_t               bAddressMode;
  /** flags see \ref PNM_AP_CFG_IOD_FLAG_E */
  uint32_t              ulFlags;
  /** AR UUID - has to be unique over all configured IO-Devices
      \par See Spec:
       Coding of the field ARUUID
  */
  PNM_UUID_T            tArUuid;
  /** AR Properties see \ref PNM_AP_CFG_IOD_AR_PROP_FLAGS_E
      \par See Spec:
       Coding of the field ARProperties
  */
  uint32_t              ulArProperties;
  /** VendorID */
  uint16_t              usVendorID;
  /** DeviceID */
  uint16_t              usDeviceID;
  /** InstanceID (GSDML parameter ObjectUUID Instance) */
```


© Hilscher, 2015–2017

```
  uint16_t              usInstanceID;
  /** max. alarm data length */
  uint16_t              usMaxAlarmDataLength;
  /* RTA Timeout Factor
     \par See Spec:
      Coding of the field RTATimeoutFactor
  */
  uint16_t              usRTATimeoutFact;
  /** RTA Retries
      \par See Spec:
       Coding of the field RTARetries
  */
  uint16_t              usRTARetries;
  /** NameOfStation of the IO Device
      zero padded: length determined by searching first zero
      \note  The GSDML Element \c OrderNumber.
      \par See Spec:
       Coding of the field NameOfStationValue
  */
  uint8_t               abNameOfStation[240];
  /** IP address */
  uint32_t              ulIPAddr;
  /** network mask */
  uint32_t              ulNetworkMask;
  /** Gateway address */
  uint32_t              ulGatewayAddr;
};

/** request packet */
typedef struct PNM_AP_CFG_IOD_REQ_Ttag PNM_AP_CFG_IOD_REQ_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_IOD_REQ_Ttag
{
  /** packet header */
  PNM_AP_PCK_HEADER_T             tHead;
  /** packet data */
  PNM_AP_CFG_IOD_REQ_DATA_T       tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_CFG_IOD_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility.
ulSrcId	UINT32	any	Ignored by protocol stack
ulLen	UINT32	296	Packet data length in bytes
ulId	UINT32	any	Ignored by protocol stack
ulSta	UINT32	0	-
ulCmd	UINT32	0x9404	PNM_AP_CMD_CFG_IOD_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	any	-
Structure PNM_AP_CFG_IOD_REQ_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usDeviceHandle	PNM_AP_DEVICEHANDLE_T	1 … 128	Unique handle of this IO-Device defined by the sender of this packet
bARType	UINT8	0x01, 0x10	AR Type
bAddressMode	UINT8	0 .. 1	IP address mode
ulFlags	UINT32		Flags controlling AR behavior
tArUuid	PNM_UUID_T		UUID associated with this AR
ulArProperties	UINT32		Properties of this AR.
usVendorID	UINT16	GSDML	PROFINET Vendor ID of the Device
usDeviceID	UINT16	GSDML	PROFINET Device ID of the Device
usInstanceID	UINT16	GSDML	PROFINET Instance ID of the Device
usMaxAlarmDataLength	UINT16	200 … 224	Maximum alarm data length
usRTATimeoutFact	UINT16	1 … 0xFFFF	RTA Timeout Factor
usRTARetries	UINT16	3 … 10	RTA Retries
abNameOfStation[240]	UINT8[]		NameOfStation of the IO Device
ulIPAddr	UINT32		IP address
ulNetworkMask	UINT32		Network mask
ulGatewayAddr	UINT32		Gateway address (i.e. IP address of gateway)

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulStructVersion`

Structure version of the configuration structure. Used for future extensions.

Parameter `usDeviceHandle`

This parameter specifies the handle of the AR to be configured.

Parameter `bARType`

This parameter describes the type of the AR

Name	Numeric value	Meaning
PNM_AP_CFG_IOD_AR_TYPE_SINGLE	0x01	AR Type for RT communication
PNM_AP_CFG_IOD_AR_TYPE_RTC3	0x10	AR Type for IRT communication

Parameter `bAddressMode`

This parameter describes the possible modes of ip address resolution

Name	Numeric value	Meaning
PNM_AP_CFG_IOD_ADDRESS_MODE_DCP_IDENTIFY_ASSIGN	0 (default)	In this mode, the controller discovers the device using its station name or alias name (only if topology info available) and will assign the configured ip address if required.
PNM_AP_CFG_IOD_ADDRESS_MODE_DCP_DISCOVER	1	The controller identifies the device using its station name or alias name and will use the ip address reported by the device. The ip address of device will not be reassigned in this mode. Please note that the Controller will not establish the communication if the ip address set in IO Device is set to zero or unreachable.

Parameter `ulFlags`

The parameter `ulFlags` is a bitmask of flag bits specifying additional properties of the AR. The following table lists the defined bits

Name	Numeric value	Meaning
PNM_AP_CFG_IOD_FLAG_WRITE_MULTIPLE_SUPPORTED	0x01	Use multiple write to parameterize the device. This flag should be set if the PROFINET IO Device associated with this AR supports multiple write. (Extracted from GSDML) The PROFINET IO Controller will pack multiple parameter writes into one RPC Service to save Startup Time.
RESERVED	0x02	This value is reserved for future usage. Set to 0 for compatibility reasons.

Parameter `tArUUID`

Unique identifier of the AR. The AR UUID shall be generated by the engineering software or application. It is a unique id of the AR and must be non-zero.

© Hilscher, 2015–2017

Parameter ulArProperties

The parameter ulARProperties is a bitmask describing PROFINET Properties of this AR. A combination of the following values is allowed:

Name	Numeric value	Meaning
PNM_AP_CFG_IOD_AR_PROP_FLAG_SUPERVISOR_TAKEOVER_ALLOWED	0x00000008	A supervisor is allowed to take over submodules of this AR.
PNM_AP_CFG_IOD_AR_PROP_FLAG_STARTUPMODE_ADVANCED	0x40000000	AR start-up is performed according to PROFINET specification 2.3 Advanced Startup Mode. (Extract from GSDML file). If the bitmask is not set, then the PROFINET V2.2 Startup Mode is used.
PNM_AP_CFG_IOD_AR_PROP_FLAG_PULL_MODULE_ALARM_ALLOWED	0x80000000	The alarm type PULL MODULE is supported by the device. (Extract from GSDML file)

Parameter usVendorID

VendorID to be used by IO Controller for addressing the IO Device. (Extract from GSDML file)

Parameter usDeviceID

DeviceID to be used by IO Controller for addressing the IO Device (Extract from GSDML file)

Parameter usInstanceID

The instance ID of the device. (Extract from GSDML file, Attribute "ObjectUUID Instance")

Parameter usMaxAlarmDataLength

The maximum length of payload data to be transported in an alarm. The minimum value is 200 byte and must be supported by any PROFINET Device. The upper limit of the PROFINET IO Controller is 224 byte.

Parameter usRTATimeoutFact

This parameter is used to specify the timeout of acyclic alarm services. When one side of the AR sends an alarm it is expected that the other side acknowledges the alarm within usRTATimeoutFact * 100ms. If the alarm is not acknowledged, up to usRTARetries will be performed. If still no acknowledge is received, the AR will be aborted. Default value is 1. For maximum compatibility values of the range 1 to 100 should be used.

Parameter usRTARetries

Number of times an alarm shall be retried before giving up and aborting the AR, Default value is 3

Parameter abNameOfStation[240]

The parameter abNameOfStation is the PROFINET Name Of Station to be used for this AR. The PROFINET IO Controller will establish the AR with a PROFINET IO Device with that NameOfStation. If multiple devices with same NameOfStation exists no connection will be made. The filed should be zero padded to full field length.

Parameter ulIPAddr

IP address to be assigned to the PROFINET IO Device. This IP Address Setting is to be specified by the application and will be assigned to the PROFINET IO Device by the PROFINET IO Controller on Startup.

© Hilscher, 2015–2017

Parameter `ulNetmask`

Network Mask to be assigned to the PROFINET IO Device. This IP Address Setting is to be specified by the application and will be assigned to the PROFINET IO Device by the PROFINET IO Controller on Startup.

Parameter `ulGatewayAddr`

Gateway Address to be assigned to the PROFINET IO Device. This IP Address Setting is to be specified by the application and will be assigned to the PROFINET IO Device by the PROFINET IO Controller on Startup.

3.1.4.2 PNM_AP_CFG_IOD_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T          PNM_AP_CFG_IOD_CNF_T;
```

Packet description

Structure `PNM_AP_CFG_IOD_CNF_T`			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure `PNM_AP_PCK_HEADER_T`			
`ulDest`	UINT32		-
`ulSrc`	UINT32		-
`ulDestId`	UINT32	0	
`ulSrcId`	UINT32	mirrored	-
`ulLen`	UINT32	0	Packet data length in bytes
`ulId`	UINT32	mirrored	-
`ulSta`	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
`ulCmd`	UINT32	0x9405	`PNM_AP_CMD_CFG_IOD_CNF`
`ulExt`	UINT32	0	-
`ulRoute`	UINT32		Ignore

© Hilscher, 2015–2017

3.1.5 Configure AR parameters service

This service can be used to configure additional AR parameters.

3.1.5.1 PNM_AP_CFG_AR_PRM_REQ_T request

Packet structure reference

```
typedef uint16_t PNM_AP_ARVENDORBLOCKHANDLE_T;

enum PNM_AP_ARVENDORBLOCKHANDLE_Etag
{
  /** The first handle value to use for AR Vendorblocks*/
  PNM_AP_ARVENDORBLOCKHANDLE_FIRST = 0x0001,
  /** The last handle value to use for AR Vendorblocks*/
  PNM_AP_ARVENDORBLOCKHANDLE_LAST  = 0x0100,
};

/* AR Vendor Block, refer to GSDML ARVendorBlock */
/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_Ttag PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_Ttag
{
  /** structure version of this structure */
  uint32_t                      ulStructVersion;
  /** handle of this AR VendorBlock, required to read out the Response at runtime */
  PNM_AP_ARVENDORBLOCKHANDLE_T  usArVendorBlockHandle;
  /** AP Structure Identifier according to GSDML */
  uint16_t                      usAPStructureIdentifier;
  /** API according to GSDML */
  uint32_t                      ulApi;
  /** Array of the VendorBlock data, sizeof array depend on parameter usDataLen  */
  uint8_t                       abData[ARRAYS_OF_LENGTH_ZERO];
} ;

/* IR Data */
/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_AR_PRM_IRDATA_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_AR_PRM_IRDATA_Ttag PNM_AP_CFG_AR_PRM_IRDATA_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_AR_PRM_IRDATA_Ttag
{
  /** structure version of this structure */
  uint32_t              ulStructVersion;
  /* RTC3 domain UUID of this IRT IO device
     needs to match the UUID of IO Controllers Sync parameters
     needs to match the UUID of IO Devices PDSyncData parameters
     \par See Spec:
      Coding of the field IRDataUUID
  */
  PNM_UUID_T            tIrDataUuid;
};

#ifdef PNM_SUPPORT_PARAMETER_SERVER
#warning "usage of Parameter Server is not supported..."
/* Parameter Server */
/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_AR_PRM_PARAMETERSERVER_BLOCK_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_AR_PRM_PARAMETERSERVER_BLOCK_Ttag
PNM_AP_CFG_AR_PRM_PARAMETERSERVER_BLOCK_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_AR_PRM_PARAMETERSERVER_BLOCK_Ttag
{
  /* external Parameter server ObjUUID */
  PNM_UUID_T            tPrmServerUuid;
  uint32_t              ulPrmServerProp;
```


© Hilscher, 2015–2017

```
  uint32_t              ulPrmServerActivityTimeoutFactor;
  /** NameOfStation, zero padded: length determined by searching
   * first zero */
  uint8_t               abPrmServerNameOfStation[240];
};
#endif

/* FastStartUp */
/* allowed values for field ulFSParamMode */
enum PNM_AP_CFG_AR_PRM_FSPARAM_MODE_Etag
{
  PNM_AP_CFG_AR_PRM_FSPARAM_MODE_DISABLE = 1,
  PNM_AP_CFG_AR_PRM_FSPARAM_MODE_ENABLE  = 2,
};
typedef enum PNM_AP_CFG_AR_PRM_FSPARAM_MODE_Etag PNM_AP_CFG_AR_PRM_FSPARAM_MODE_E;

/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_AR_PRM_FSU_PARAMETERS_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_PRM_AR_FSU_PARAMETERS_Ttag PNM_AP_CFG_PRM_AR_FSU_PARAMETERS_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_PRM_AR_FSU_PARAMETERS_Ttag
{
  /** structure version of this structure */
  uint32_t              ulStructVersion;
  /** FSParameterMode see \ref PNM_AP_CFG_AR_PRM_FSPARAM_MODE_E
      \par See Spec:
       Coding of the field FSParameterMode
  */
  uint32_t              ulFSParamMode;
  /** FSParameter UUID, irrelevant if FSParamMode is set to off
      \par See Spec:
       Coding of the field FSParameterUUID
  */
  PNM_UUID_T            tFSParamUuid;
};

/* identifier */
typedef enum
{
  PNM_AP_CFG_AR_PRM_IDENT_ARVENDORBLOCKREQ = 0,
  PNM_AP_CFG_AR_PRM_IDENT_IRDATA           = 1,
  PNM_AP_CFG_AR_PRM_IDENT_PARAMETERSERVER  = 2,
} PNM_AP_CFG_AR_PRM_IDENTIFIER_E;

typedef union
{
  PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_T          tArVendorBlock;
  PNM_AP_CFG_AR_PRM_IRDATA_T                 tIrData;
#ifdef PNM_SUPPORT_PARAMETER_SERVER
  PNM_AP_CFG_AR_PRM_PARAMETERSERVER_BLOCK_T  tPrmServerBlock;
#endif
} PNM_AP_CFG_AR_PRM_UNION_T;

/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_AR_PRM_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_AR_PRM_DATA_Ttag PNM_AP_CFG_AR_PRM_DATA_T;
/** request packet data */
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_AR_PRM_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t               ulStructVersion;
  /** IO-Device handle the submodule belongs to */
  PNM_AP_DEVICEHANDLE_T   usDeviceHandle;
  /** identifier see \ref PNM_AP_CFG_AR_PRM_IDENTIFIER_E */
  uint16_t               usIdentifier;
  /** parameter data */
  PNM_AP_CFG_AR_PRM_UNION_T uData;
};
```


© Hilscher, 2015–2017

```
typedef struct PNM_AP_CFG_AR_PRM_REQ_Ttag PNM_AP_CFG_AR_PRM_REQ_T;

/** request packet */
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_AR_PRM_REQ_Ttag
{
  /** packet header */
  PNM_AP_PCK_HEADER_T                tHead;
  /** packet data */
  PNM_AP_CFG_AR_PRM_DATA_T           tData;
};
```

Packet description

Structure PNM_AP_CFG_AR_PRM_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	-
ulSrcId	UINT32	any	-
ulLen	UINT32		Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x9406	PNM_AP_CMD_CFG_AR_PRM_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	any	-
Structure PNM_AP_CFG_AR_PRM_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usDeviceHandle	PNM_AP_DEVICEHANDLE_T	1 … 128	Handle of AR this AR Parameter belongs to
usIdentifier	UINT16		Identifier for the type
uData	PNM_AP_CFG_AR_PRM_UNION_T		Parameter data

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used for future extensions.

Parameter usDeviceHandle

Reference to the AR for which to configure the AR parameter for

© Hilscher, 2015–2017

Parameters `usIdentifier`, `uData`

The values of `usIdentifier` specifies which kind of AR parameter is configured and which field of `uData` is valid

Name	Numeric value	Packet length	Meaning
PNM_AP_CFG_AR_PRM_IDENT_ARVENDORBLOCKREQ	0	8 + 12 + Length of ARVendorBlockReq data	uData.tArVendorBlock contains an AR Vendor Block Request to configure for the AR

Structure PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_T

Variable	Type	Value / Range	Description
ulStructVersion	UINT32	1	Structure version of this structure
usArVendorBlockHandle	UINT16	1 to 256	Handle to this AR vendor block. This a configuration wide unique handle. That means different ARs must use different handles here.
usAPStructureIdentifier	UINT16	0 to 0x7FFF, 0x8000	AP Structure Identifier according to GSDML
ulApi	UINT32	any	API according to GSDML
abData[]	UINT8[]		Byte array containing the AR VendorBlock data. Size of this array depends on parameter tHead.ulLen. Allowed length of array: 0 (no data) - 512

3.1.5.2 PNM_AP_CFG_AR_PRM_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T            PNM_AP_CFG_AR_PRM_CNF_T;
```

Packet description

Structure PNM_AP_CFG_AR_PRM_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x9407	PNM_AP_CMD_CFG_AR_PRM_CNF
ulExt	UINT32	0	
ulRoute	UINT32		Ignore

© Hilscher, 2015–2017

3.1.6 Configure IOCR service

This service shall be used to configure the IOCRs of PROFINET IO Controller. Typically each AR is associated with one input IOCR and one output IOCR.

Note: Please remind that the PROFINET IO Controller firmware assigns the PROFINET frame id of an IOCR internally. The application cannot choose a particular frame id. Details on the relationship between IOCR handle and frame id can be found in section *Configuration of PROFINET IO Controller* on page 13.

3.1.6.1 PNM_AP_CFG_IOCR_REQ_T request

Packet structure reference

```
typedef uint16_t PNM_AP_IOCR_HANDLE_T;

enum PNM_AP_IOCR_HANDLE_Etag
{
  /** minimum value for IOCR handle with type Input */
  PNM_AP_IOCR_HANDLE_INPUT_FIRST = 0x1000,
  /** maximum value for IOCR handle with type Input */
  PNM_AP_IOCR_HANDLE_INPUT_LAST  = 0x107F,

  /** minimum value for IOCR handle with type Output */
  PNM_AP_IOCR_HANDLE_OUTPUT_FIRST = 0x2000,
  /** maximum value for IOCR handle with type Output */
  PNM_AP_IOCR_HANDLE_OUTPUT_LAST  = 0x207F,
};

typedef enum PNM_AP_IOCR_HANDLE_Etag PNM_AP_IOCR_HANDLE_E;

/* IOCR Type */
typedef enum
{
  /* this is an Input CR */
  PNM_AP_CFG_IOCR_TYPE_INPUT             = 1,
  /* this is an Output CR */
  PNM_AP_CFG_IOCR_TYPE_OUTPUT            = 2,
  /* not supported, for future use */
  PNM_AP_CFG_IOCR_TYPE_MULTICAST_PROVIDER = 3,
  /* not supported, for future use */
  PNM_AP_CFG_IOCR_TYPE_MULTICAST_CONSUMER = 4,
  /* invalid value */
  PNM_AP_CFG_IOCR_TYPE_MAX,
} PNM_AP_CFG_IOCR_TYPE_E;

/* IOCR Properties */
typedef enum
{
  PNM_AP_CFG_IOCR_PROP_RT_CLASS_1_LEGACY  = 0x01,    /* RT_CLASS_1, use legacy FrameIDs
(0xC000-0xF7FF) */
  PNM_AP_CFG_IOCR_PROP_RT_CLASS_1         = 0x02,    /* RT_CLASS_1, use FrameIDs (0x8000-
0xBBFF) */
  PNM_AP_CFG_IOCR_PROP_RT_CLASS_3         = 0x03,    /* RT_CLASS_3 (IRT), use FrameIDs
(0x0100-0x06FF) */
  PNM_AP_CFG_IOCR_PROP_RT_CLASS_UDP       = 0x04,    /* not supported, for future use */
} PNM_AP_CFG_IOCR_PROP_E;

/** Definition of the frame ids the controller will use
 *
 * The following array defines the base value for frameid assignment
 * within the controller firmware. The frame id of a given IOCR will
 * be the IOCR Handle minus PNM_AP_IOCR_HANDLE_XXX_FIRST plus
 * the base value selected according IOCR properties */
enum PNM_AP_CFG_IOCR_FRAMEIDBASE_Etag
```


© Hilscher, 2015–2017

```
{
  /** Base value for RTC3 Input IOCR FrameIds
   *
   * The PnC XC Code will only accept RTC3 CPM with 0x100 <= Frame Id < 0x180.
   * This is hardcoded within XC !
   */
  PNM_AP_CFG_IOCR_FRAMEIDBASE_RTC3INPUT   = 0x0100,
  PNM_AP_CFG_IOCR_FRAMEIDBASE_RTC3OUTPUT  = 0x0200,
  PNM_AP_CFG_IOCR_FRAMEIDBASE_RTC1        = 0x8000,
  PNM_AP_CFG_IOCR_FRAMEIDBASE_RTC1LEGACY  = 0xC000,
};

typedef enum PNM_AP_CFG_IOCR_FRAMEIDBASE_Etag PNM_AP_CFG_IOCR_FRAMEIDBASE_E;

/* SendClockFactor */
#define PNM_AP_CFG_IOCR_SEND_CLOCK_DEFAULT         (0x0020)  /**< default value for
SendClockFactor */
#define PNM_AP_CFG_IOCR_SEND_CLOCK_MIN_RTC1        (0x0008)  /**< minimum value allowed
for SendClockFactor for RT_CLASS_1 */
#define PNM_AP_CFG_IOCR_SEND_CLOCK_MIN_RTC3        (0x0008)  /**< minimum value allowed
for SendClockFactor for RT_CLASS_3 */
#define PNM_AP_CFG_IOCR_SEND_CLOCK_MAX             (0x0080)  /**< maximum value allowed
for SendClockFactor */

/* ReductionRatio */
/* the possible values depend on SendClock */
#define PNM_AP_CFG_IOCR_REDUCT_RATIO_MIN           (0x0001)  /**< minimum value allowed
for ReductionRatio */
#define PNM_AP_CFG_IOCR_REDUCT_RATIO_MAX_RTC1      (0x0200)  /**< maximum value allowed
for ReductionRatio for RT_CLASS_1 */
#define PNM_AP_CFG_IOCR_REDUCT_RATIO_MAX_RTC3      (0x0010)  /**< maximum value allowed
for ReductionRatio for RT_CLASS_3 */

/* Phase */
/* the possible values depend on SendClock and ReductionRatio */
#define PNM_AP_CFG_IOCR_PHASE_VALUE_MIN            (0x0001)  /**< minimum value allowed
for Phase */
#define PNM_AP_CFG_IOCR_PHASE_VALUE_MAX            (0x4000)  /**< maximum value allowed
for Phase */

/* FrameSendOffset */
#define PNM_AP_CFG_IOCR_FRAME_SEND_OFFSET_RTC1     (0xFFFFFFFF) /**< best practice,
default value */

/* DataHold Factor */
#define PNM_AP_CFG_IOCR_DATAHOLD_DEFAULT           (0x0003)  /**< default value for
datahold factor */
#define PNM_AP_CFG_IOCR_DATAHOLD_MIN               (0x0001)  /**< minimum value allowed
for datahold factor */
#define PNM_AP_CFG_IOCR_DATAHOLD_MAX               (0x1E00)  /**< maximum value allowed
for datahold factor */

/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_IOCR_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_IOCR_DATA_Ttag PNM_AP_CFG_IOCR_DATA_T;

/** request packet data */
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_IOCR_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t                ulStructVersion;
  /** this IOCRs unique handle */
  PNM_AP_IOCR_HANDLE_T    usIocrHandle;
  /** IO-Device handle the IOCR belongs to */
  PNM_AP_DEVICEHANDLE_T   usDeviceHandle;

  /** Flags for future use, set to 0 */
  uint32_t                ulFlags;
  /** IOCR properties see \ref PNM_AP_CFG_IOCR_PROP_FLAG_E
```


© Hilscher, 2015–2017

```
      \par See Spec:
       Coding of the field IOCRProperties
  */
  uint32_t               ulIocrProp;
  /** IOCR type see \ref PNM_AP_CFG_IOCR_TYPE_E
      \par See Spec:
       Coding of the field IOCRType
  */
  uint16_t               usIocrType;
  /** IOCR multicast MAC address for future use, currently unsupported, set to 0
      \par See Spec:
       Coding of the field IOCRMulticastMACAdd
  */
  uint8_t                abMcastMACAddr[6];
  /** IOCR data length, the length of the cyclic frame represented by this IOCR
      \par See Spec:
       Coding of the field DataLength
  */
  uint16_t               usDataLen;
  /** IOCR sendclock factor - see above
      \par See Spec:
       Coding of the field SendClockFactor
  */
  uint16_t               usSendClockFact;
  /** IOCR reduction ratio - see above
      \par See Spec:
       Coding of the field ReductionRatio
  */
  uint16_t               usReductRatio;
  /** IOCR phase - see above
      \par See Spec:
       Coding of the field Phase
  */
  uint16_t               usPhase;
  /** IOCR sequence number
      \par See Spec:
       Coding of the field Sequence
  */
  uint16_t               usSequence;
  /** IOCR datahold factor - see above
      \par See Spec:
       Coding of the field DataHoldFactor
  */
  uint16_t               usDataHoldFact;
  /** IOCR FrameSendOffset - see above
      \par See Spec:
       Coding of the field FrameSendOffset
  */
  uint32_t               ulFrameSendOffs;

  /** base address for the whole IO data of this IOCR in IO Controllers DPM input/output
area */
  uint16_t               usDpmOffset;

  /** TODO: support vlan tag setting? */
};

typedef struct PNM_AP_CFG_IOCR_REQ_Ttag PNM_AP_CFG_IOCR_REQ_T;
/** request packet */
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_IOCR_REQ_Ttag
{
  /** packet header */
  PNM_AP_PCK_HEADER_T          tHead;
  /** packet data */
  PNM_AP_CFG_IOCR_DATA_T       tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_CFG_IOCR_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility.
ulSrcId	UINT32	any	-
ulLen	UINT32	42	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x9408	PNM_AP_CMD_CFG_IOCR_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	0	-
Structure PNM_AP_CFG_IOCR_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usIocrHandle	PNM_AP_IOCR_HANDLE_T	0x1000...0x107F, 0x2000...0x207F	The handle of the IOCR to configure
usDeviceHandle	PNM_AP_DEVICEHANDLE_T	Valid device handle	The handle of the AR to associate this IOCR to
ulFlags	UINT32	0	
ulIocrProp	UINT32	1 … 3	Properties of the IOCR
usIocrType	UINT16	1 … 3	Type of the IOCR
abMcastMACAddr[6]	UINT8[]	00:00:00:00:00:00	Multicast MAC Address to be used for this IOCR. Reserved for future usage. Set to zero for compatibility.
usDataLen	UINT16	1… 1440 (% 4 == 0)	Length of the data part of the IOCR without Frame Header and APDU Status (C_SDU Length). Must be a multiple of 4.
usSendClockFact	UINT16	8 … 128	Send Clock Factor to use for this IOCR. Only values with power of 2 are allowed.
usReductRatio	UINT16	1 … 512	Reduction Ratio for this IOCR. Only values with power of 2 are allowed.
usPhase	UINT16	1 … 512	Phase for this IOCR. Must be smaller or equal usReductionRatio
usSequence	UINT16	0	Currently unused. Set to zero for future compatibility
usDataHoldFact	UINT16	1 … 7680	Data Hold Factor for this IOCR
ulFrameSendOffs	UINT32	0 … 3999999	Frame send offset for RTC Class 3 IOCRs. Set to zero for non RTC Class 3 IOCRs
usDpmOffset	UINT16	0 … 5700 (% 4 == 0)	Start address for IOCR data in DPM Input or Output Area. Must be 4-byte aligned.

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulStructVersion`

Structure version of this structure. Used for future extensions of this structure.

Parameter `usIocrHandle`

The parameter usIocrHandle is a reference to the IOCR object to be configured. The following values are defined

Minimum value	Maximum value	Usage
0x1000	0x103F	Input IOCR for RT Class 1 / RT Class 3
0x1040	0x107F	Input IOCR for RT Class 1
0x2000	0x203F	Output IOCR for RT Class 1 / RT Class 3
0x2040	0x207F	Output IOCR for RT Class 1

The PROFINET IO Controller will internally generate the frame id of the Input IOCRs and RT Class 3 Output IOCRs based on the frame id value. For details see section *Configuration of PROFINET IO Controller* on page 13.

Parameter `ulIocrProp`

The IOCR properties parameter `ulIocrProp` is a bit field describing the IOCR in more detail. The following values are currently defined

Symbolic name	Numeric value	Usage
`PNM_AP_CFG_IOCR_PROP_RT_CLASS_1_LEGACY`	0x01	RT_CLASS_1
`PNM_AP_CFG_IOCR_PROP_RT_CLASS_1`	0x02	RT_CLASS_1
`PNM_AP_CFG_IOCR_PROP_RT_CLASS_3`	0x03	RT_CLASS_3 (IRT)

Parameter `usIocrType`

This parameter determines the type of the IOCR. Due to the internal firmware structure the type is determined by the IOCR's handle value.

Symbolic name	Numeric value	Usage
`PNM_AP_CFG_IOCR_TYPE_INPUT`	1	For Input IOCR with 0x1000 <= usIocrHandle <= 0x107F
`PNM_AP_CFG_IOCR_TYPE_OUTPUT`	2	For Output IOCR with 0x2000 <= usIocrHandle <= 0x207F

Parameter `abMcastMacAddr`

This parameter is reserved for future usage. Set to zero.

Parameter `usDataLen`

The length of data part of the IOCR is defined by this parameter. It determines how many byte the PROFINET IO Controller firmware will copy to / from the DPM Input / Output Area for this IOCR. Internally, the data exchange is restricted to 4 byte- aligned operations, thus the parameter usDataLen must be a multiple of 4.

The PROFINET IO Controller will internally pad the IOCR data block (CSDU) to the minimum size of 40 byte if a value smaller than 40 byte is specified. This does not affect the used memory space in DPM area. E.g. if a usDataLen = 20 byte is specified, a block of 20 byte will be used in DPM and the IOCR C_SDU length will be 40.

© Hilscher, 2015–2017

When computing the length of an IOCR, the process data, provider states and consumer states must be considered.

Parameter `usSendClockFact, usReductRatio and usPhase`

These parameters are used to configure the cycle time of the IOCR and thus the associated AR and device. PROFINET uses the concept of a Macro and Micro Cycle to provide some kind of flexibility for Process Data Timing. The sendclock factor determines the base clock for the IOCR (The Micro Cycle). The Send Cycle interval is calculated as follows:

usSendClockFact * 31.25 µs

All RT Class 3 IOCRs must use the same send clock factor. This micro cycle is then multiplied by the reduction ratio usReductRatio to define the Data Cycle (Macro Cycle). The reduction ratio may be different for different IOCRs. The data cycles defines the process data exchange interval and is calculated as follows

usReductRatio * usSendclockFact * 31.25 µs

A network telegram will be issued for an IOCR every data cycle. Finally the parameter usPhase configures in which Send Cycle of a Data Cycle the IOCR shall be transmitted. This is especially useful in case of RT Class 3 IOCR as the Data Cycles of all RT Class 3 devices are synchronized. It is thus possible to distribute the network load more homogenous.

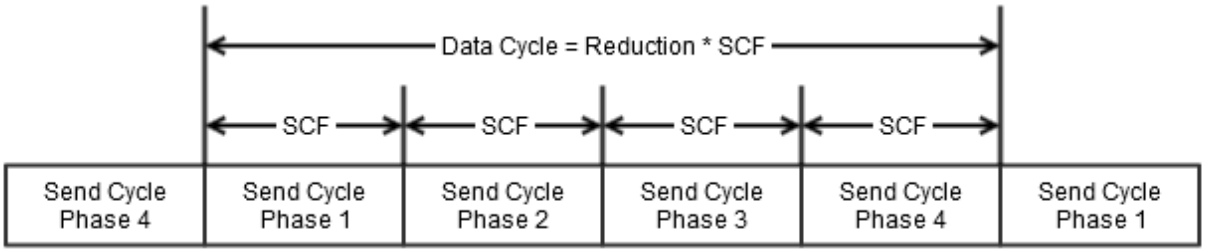

Figure 8: Definition of Send Clock Factor, Reduction Ration and Phase

The following limits apply to this parameters:

- 8 <= usSendClockFact <= 128 (allowed values for IO Controller see Technical data, for IO Devices see their GSDML)
- usSendClockFact of all RT Class 3 IOCRs must be identical
- 1 <= usReduction <= 512 for RT Class 1 IOCRs (allowed values for IO Devices see their GSDML)
- 1 <= usReduction <= 16 for RT Class 3 IOCRs (allowed values for IO Devices see their GSDML)
- 1 <= usPhase <= usReduction for one IOCR

Parameter usSequence

Currently not used. Set to zero for future compatibility.

© Hilscher, 2015–2017

Parameter `usDataHoldFact`

This parameter defines the timeout for the IOCR watchdog. If no IOCR data could be exchanged for usDataHoldFact Data Cycle repetitions, the AR will be aborted with an timeout error. The Data Hold Timeout time can be calculated as follows:

Data Hold Timeout =
(usDataHoldFact + 1) * usReductRatio * usSendclockFact * 31.25 [microseconds]

The maximum allowed value for the Data Hold Timeout is 1,92 seconds.

It is strongly recommended to set this timeout to a value greater or equal than 3. Lower values might cause problems with occasional network transmission failures due to external electromagnetic noise.

Parameter `ulFrameSendOffs`

This parameter defines the send offset of the IOCR frame in respect to send clock cycle start. It is used in RT Class 3 mode. The parameter must match the corresponding value in the IR Data Sets of the associated device. In RT Class 1 mode set this parameter to 0.

Parameter `usDpmOffset`

This parameter defines where the PROFINET IO Controller will take the data for an Output IOCR from the DPM Output Area or to put the data of an Input IOCR into the DPM Input Area. The value must be 4-byte aligned due to the internal implementation oft the PROFINET IO Controller. The following image shows the relationships between the parameter usDPMOffset, usDataLen and the DPM Area. The sum of usDPMOffset and usDataLen shall not exceed the length of DPM Area (5700 byte for Ouput Image, 5652 byte for Input Image).

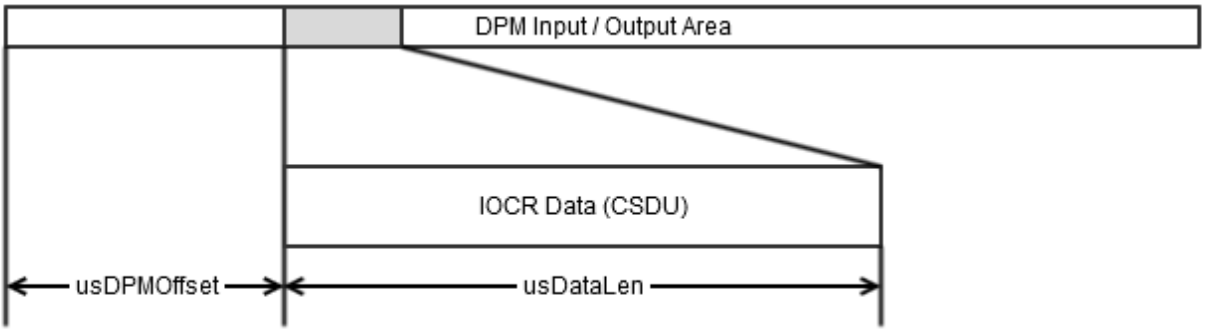

Figure 9: Memory mapping of IOCR data (CSDU) to DPM input/output area

© Hilscher, 2015–2017

3.1.6.2 PNM_AP_CFG_IOCR_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T          PNM_AP_CFG_IOCR_CNF_T;
```

Packet description

Structure PNM_AP_CFG_IOCR_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		
ulSrc	UINT32		Ignore
ulDestId	UINT32	0	
ulSrcId	UINT32	mirrored	
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x9409	PNM_AP_CMD_CFG_IOCR_CNF
ulExt	UINT32	any	
ulRoute	UINT32	any	

© Hilscher, 2015–2017

3.1.7 Configure submodule service

The configure submodule service shall be used by the application to configure a submodule of a device.

3.1.7.1 PNM_AP_CFG_SUBMODULE_REQ_T request

Packet structure reference

```
typedef uint16_t PNM_AP_SUBMODULE_HANDLE_T;


/** Submodule Properties Bitmask */
typedef enum
{
  PNM_AP_CFG_SUBMODULE_PROPERTIES_INPUT                       = 0x0001, /* Set If Input
Data Available */
  PNM_AP_CFG_SUBMODULE_PROPERTIES_OUTPUT                      = 0x0002, /* Set If Output
Daata Available*/

  PNM_AP_CFG_SUBMODULE_PROPERTIES_ACCESS_FULL                 = 0x0000, /* full access
to submodule (default)*/
  PNM_AP_CFG_SUBMODULE_PROPERTIES_ACCESS_SHARED               = 0x0004, /* Shared Input
submodule, no alarms and no parameterization */

  PNM_AP_CFG_SUBMODULE_PROPERTIES_ZERO_INPUT_DATA_LENGTH      = 0x0008, /* not
supported, for future use */
  PNM_AP_CFG_SUBMODULE_PROPERTIES_ZERO_OUTPUT_DATA_LENGTH     = 0x0010, /* not
supported, for future use */

  PNM_AP_CFG_SUBMODULE_PROPERTIES_USE_IOXS                    = 0x0000, /* IOXS enabled,
(default) */
  PNM_AP_CFG_SUBMODULE_PROPERTIES_DISCARD_IOXS                = 0x0020, /* not
supported, for future use */
} PNM_AP_CFG_SUBMODULE_PROPERTIES_E;

/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_SUBMODULE_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_SUBMODULE_DATA_Ttag PNM_AP_CFG_SUBMODULE_DATA_T;
/** request packet data */
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_SUBMODULE_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t                  ulStructVersion;
  /** unique submodule handle (freely defined by the user) */
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle;
  /** IO-Device handle the submodule belongs to */
  PNM_AP_DEVICEHANDLE_T     usDeviceHandle;
  /** consumer IOCR handle the submodule belongs to */
  PNM_AP_IOCR_HANDLE_T      usInputIocrHandle;
  /** provider IOCR handle the submodule belongs to */
  PNM_AP_IOCR_HANDLE_T      usOutputIocrHandle;
  /** module identifier according to GSDML */
  uint32_t                  ulModuleIdentNumber;
  /** submodule identifier according to GSDML */
  uint32_t                  ulSubmoduleIdentNumber;
  /** API the submodule belongs to */
  uint32_t                  ulApi;
  /** slot the submodule is expected in */
  uint16_t                  usSlot;
  /* subslot the submodule is expected in */
  uint16_t                  usSubslot;
  /** submodule flags see \ref PNM_AP_CFG_SUBMODULE_FLAGS_E */
  uint16_t                  usSubmoduleProperties;
  /** length of the data this submodule provides according to GSDML */
  uint16_t                  usDataLenInput;
```


© Hilscher, 2015–2017

```
  /** length of the data this submodule consumes according to GSDML */
  uint16_t                  usDataLenOutput;
  /** offset inside the frame where the input data of this submodule shall reside */
  uint16_t                  usFrameOffsetInput;
  /** offset inside the frame where the output data of this submodule shall reside */
  uint16_t                  usFrameOffsetOutput;
  /** offset inside the frame where IOCS of input data of this submodule shall reside.
      Note that this is an offset in the output IOCR */
  uint16_t                  usIOCSFrameOffsetInput;
  /** offset inside the frame where IOCS of output data of this submodule shall reside.
      Note that this is an offset in the input IOCR */
  uint16_t                  usIOCSFrameOffsetOutput;
};

typedef struct PNM_AP_CFG_SUBMODULE_REQ_Ttag PNM_AP_CFG_SUBMODULE_REQ_T;
/** request packet */
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_SUBMODULE_REQ_Ttag
{
  /** packet header */
  PNM_AP_PCK_HEADER_T          tHead;
  /** packet data */
  PNM_AP_CFG_SUBMODULE_DATA_T  tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_CFG_SUBMODULE_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility.
ulSrcId	UINT32	any	-
ulLen	UINT32	42	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x940A	PNM_AP_CMD_CFG_SUBMODULE_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	any	-
Structure PNM_AP_CFG_IOC_DATA_T			
ulStructVersion	UINT32	1	Structure version used for further extensions to this packet.
usSubmoduleHandle	PNM_AP_SUBMODULE_HANDLE_T	1...2048	Submodule Handle
usDeviceHandle	PNM_AP_DEVICEHANDLE_T	1...128	AR this submodule is associated to
usIocrInputHandle	PNM_AP_IOCR_HANDLE_T	0x1000 to 0x107F	Input IOCR associated with this submodule
usIocrOutputHandle	PNM_AP_IOCR_HANDLE_T	0x2000 to 0x207F	Output IOCR associated with this submodule
ulModuleIdentNumber	UINT32	any	Module identifier according to GSDML
ulSubmoduleIdentNumber	UINT32	any	Submodule identifier according to GSDML
ulApi	UINT32		Submodule API according GSDML
usSlot	UINT16		Submodule slot according device configuration
usSubslot	UINT16		Submodule subslot according device configuration
usSubmoduleProperties	UINT16		Properties of the Submodule
usDataLenInput	UINT16	0...1439	Length of the input data this submodule provides according to GSDML
usDataLenOutput	UINT16	0...1439	Length of the output data this submodule consumes according to GSDML
usFrameOffsetInput	UINT16	0...1439	Offset within the Input IOCR where the input process data and provider status shall be placed
usFrameOffsetOutput	UINT16	0...1439	Offset within the Output IOCR where the output process data and provider status shall be placed
usIOCSFrameOffsetInput	UINT16	0...1439	Offset within the Input IOCR where the output consumer status shall be placed
usIOCSFrameOffsetOutput	UINT16	0...1439	Offset within the Output IOCR where the input consumer status shall be placed

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulStructVersion`

Version of this configuration structure. Used for future extensions.

Parameter `usSubmoduleHandle`

The `usSubmoduleHandle` can be freely chosen from the user within the allowed range. It is a globally unique handle associated with a submodule. Each submodule within the whole configuration must be assigned an unique handle. This handle will be used in other services to reference to the submodule. (E.g. Record Object Read/Write)

Parameter `usDeviceHandle`

The submodule will be associated with the AR referenced by `usDeviceHandle`. This is used when establishing the connection with a PROFINET IO Device in order to verify the configuration of this particular device.

Parameters `usIocrInputHandle, usIocrOutputHandle`

Each submodule must be associated with an Input IOCR which is used to transfer the submodules input process data and provider status and output process data consumer status to the PROFINET IO Controller. Likewise each submodule must be associated with an Output IOCR which is used the transfer the submodules output process data and provider status and input process data consumer status to the PROFINET IO Device. The referenced IOCRs must be associated with the same AR as referenced by parameter usDeviceHandle.

Parameters `ulModuleIdenNumber, ulSubmoduleIdentNumber`

The identification numbers are unique IDs defined by the PROFINET IO Device vendor within the GSDML file. They are used to uniquely identify the Module and Submodule

Icon

Note: The Module ID of Submodules with same ulApi and ulSlot must be identical.

Parameters `ulApi`, `usSlot`, `usSubslot`

The parameters ulApi, usSlot, usSubslot describe where a particular Submodule has been configured within the PROFINET IO Device. These parameters depend on the actual configuration of the PROFINET IO Device. Allowed values are described in the GSDML file of the device.

Parameter `usSubmoduleProperties`

The usSubmoduleProperties is a bit field describing various properties of the submodule. The following flags are defined

Flag	Numeric value	Usage
`PNM_AP_CFG_SUBMODULE_PROPERTIES_INPUT`	0x0001	Must be set if usDataLenInput > 0 or usDataLenOutput == 0
`PNM_AP_CFG_SUBMODULE_PROPERTIES_OUTPUT`	0x0002	Must be set if usDataLenOutput > 0

Parameters `usDataLenInput`, `usDataLenOutput`

The parameters usDataLenInput and usDataLenOutput specify the lengths of the submodules input and output process data. It only contains the IO data length without IOPS. If a submodule has

© Hilscher, 2015–2017

neither input nor output data it is considered to be an input submodule with 0 byte of process data. As consequence an input provider status and input consumer status must still be allocated within the Input and Output IOCRs associated with this submodule.

Parameters `usFrameOffsetInput`, `usFrameOffsetOutput`, `usIOCSFrameOffsetInput`, `usIOCSFrameOffsetOutput`

These parameters describe the placement of the input and output process data and its associated provider and consumer states within the Input and Output IOCRs and the DPM. According to PROFINET Specification:

- usFrameOffsetInput **(1)** and usIOCSFrameOffsetOutput **(2)** must be valid for input submodules. These are submodule with input process data.
- usFrameOffsetOutput **(3)** and usIOCSFrameOffsetInput **(4)** must be valid for output submodules. These are submodules with output process data.
- usFrameOffsetInput **(1)**, usIOCSFrameOffsetOutput **(2)**, usFrameOffsetOutput **(3)** and usIOCSFrameOffsetInput **(4)** must be valid for input-output submodules. These are submodule with input and output process data.

The length of the provider and consumer data status is fixed to one byte. The mapping of the submodules data to IOCRs and DPM areas is shown in the following figure:

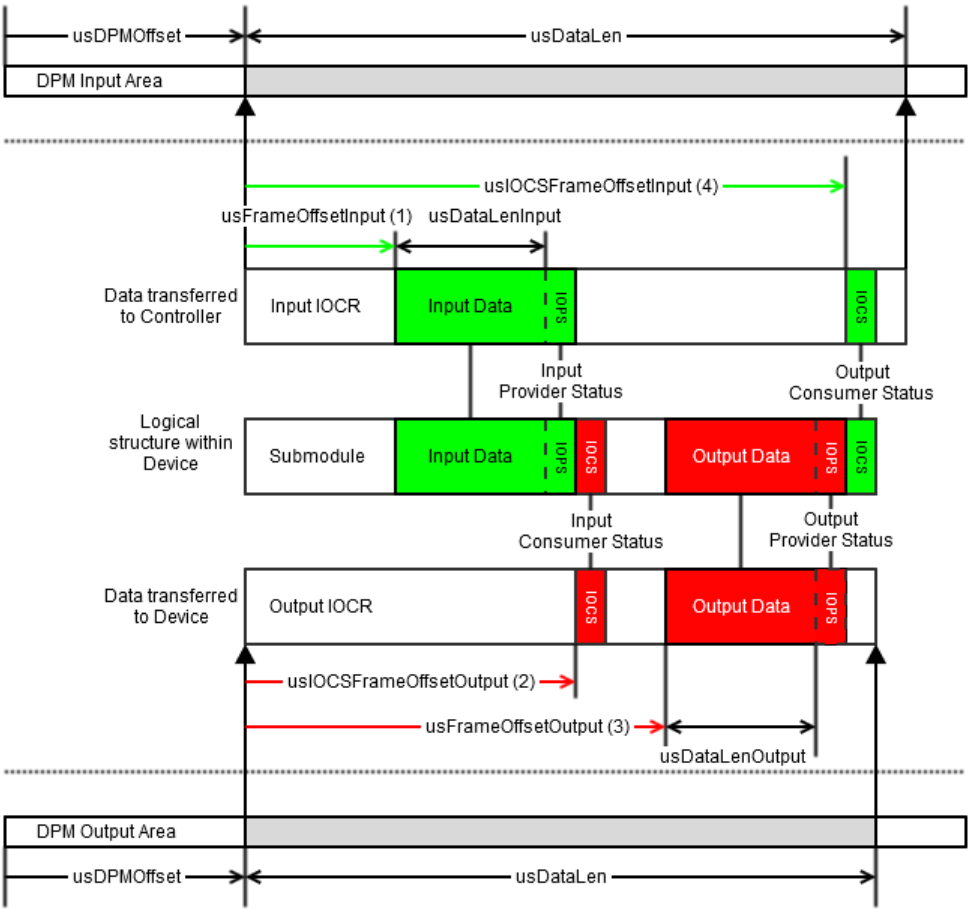

Figure 10: Mapping of submodule data to IOCRs and DPM areas

usDPMOffset and usDataLen are parameterized in Configure IOCR Service.

© Hilscher, 2015–2017

3.1.7.2 PNM_AP_CFG_SUBMODULE_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T          PNM_AP_CFG_SUBMODULE_CNF_T;
```

Packet description

Structure PNM_AP_CFG_SUBMODULE_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	
ulSrcId	UINT32	mirrored	
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x940B	PNM_AP_CMD_CFG_SUBMODULE_CNF
ulExt	UINT32	0	
ulRoute	UINT32		Ignore

© Hilscher, 2015–2017

3.1.8 Configure record service

This service shall be used to configure record parameter objects to be written by the controller to the device when establishing the AR. This service is optional.

Note: The record parameters are internally stored within a storage associated with the AR. The storage has a limited size. Please refer to section *Technical data* on page 5 for details on the available memory amount for startup record object parameterization. Additionally to the record data, each parameter consumes some byte for management information. Details are described in *PNM_AP_CFG_RECORD_REQ_T request* (page 71).

3.1.8.1 PNM_AP_CFG_RECORD_REQ_T request

Packet structure reference

```
typedef union PNM_AP_CFG_PARAMETER_Ttag PNM_AP_CFG_PARAMETER_T;

/* supported Parameter settings */
union PNM_AP_CFG_PARAMETER_Ttag
{
  PNM_AP_CFG_PRM_PDINTERFACEADJUST_T                tIntfAdjust;
  PNM_AP_CFG_PRM_PDINTERFACEFSUDATAADJUST_T         tIntfFSUDataAdjust;
  PNM_AP_CFG_PRM_SYNC_T                             tSyncData;
  PNM_AP_CFG_PRM_IRDATA_GLOBAL_T                    tIrtGlobal;
  PNM_AP_CFG_PRM_IRDATA_PHASES_T                    tIrtPhases;
  PNM_AP_CFG_PRM_IRDATA_FRAME_T                     tIrtFrames;
  PNM_AP_CFG_PRM_PDINTFMRPDATACHECK_T               tIntfMRPDataCheck;
  PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST_T              tIntfMRPDataAdjust;
  PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_T                tPortDataAdjust;
  PNM_AP_CFG_PRM_PDPORTDATA_CHECK_T                 tPortDataCheck;
  PNM_AP_CFG_PRM_PDPORTFODATAADJUST_T               tPortFODataAdjust;
  PNM_AP_CFG_PRM_PDPORTFODATACHECK_T                tPortFODataCheck;
  PNM_AP_CFG_PRM_PDNCDATACHECK_T                    tNCDataCheck;
  PNM_AP_CFG_PRM_AR_FSU_PARAMETERS_T                tArFsuData;
  PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST_T              tPortMRPDataAdjust;
  PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_T              tIsochronousModeData;
};

typedef union PNM_AP_CFG_RECORD_UNION_DATA_Ttag PNM_AP_CFG_RECORD_UNION_DATA_T;

union PNM_AP_CFG_RECORD_UNION_DATA_Ttag
{
  PNM_AP_CFG_RECORD_DATA_T    tRecord;
  PNM_AP_CFG_PARAMETER_T      tParam;
};

typedef struct PNM_AP_CFG_RECORD_Ttag PNM_AP_CFG_RECORD_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_RECORD_Ttag
{
  /** structure version of this structure */
  uint32_t                      ulStructVersion;
  /** submodule handle the record belongs to for module specific records,
   * specify use one submodule of the module, for device specific records use one
   * submodule of the whole device, e.g. DAP submodule */
  PNM_AP_SUBMODULE_HANDLE_T     usSubmoduleHandle;

  /** identifier what data is inside this dataset,
   * see \ref PNM_AP_CFG_PRM_TYPE_E */
  uint16_t                      usPrmType;

  PNM_AP_CFG_RECORD_UNION_DATA_T uData;
};

/** request packet */
```


© Hilscher, 2015–2017

```
typedef  struct PNM_AP_CFG_RECORD_REQ_Ttag PNM_AP_CFG_RECORD_REQ_T;
__PACKED_PRE struct PNM_AP_CFG_RECORD_REQ_Ttag
{
  /** packet header */
  PNM_AP_PCK_HEADER_T               tHead;
  /** packet data */
  PNM_AP_CFG_RECORD_T               tData;
};
```

Packet description

Structure PNM_AP_CFG_RECORD_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility.
ulSrcId	UINT32	any	-
ulLen	UINT32	8 …	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x940C	PNM_AP_CMD_CFG_RECORD_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	any	-
Structure PNM_AP_CFG_RECORD_T			
ulStructVersion	UINT32	1	Structure version of this structure
usSubmoduleHandle	PNM_AP_SUBMODULE_HANDLE_T	1 … 2048	The handle of the submodule this record shall be configured for
usPrmType	UINT16		The parameter type to configure Parameter type specifying what kind of data is inside this dataset (uData)
uData	PNM_AP_CFG_RECORD_UNION_DATA_T		Union of the different support record data types

Structure PNM_AP_CFG_RECORD_UNION_DATA_T		
tRecord	PNM_AP_CFG_RECORD_DATA_T	Used for specifying the manufacturers specific record data

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulStructVersion`

Version of this structure. Used for future extensions to this structure.

Parameter `usSubmoduleHandle`

This parameter specifies the submodule the record data is associated with

Parameter `usPrmType`, `uData`

The parameter usPrmType specifies the type of the record data within this request packet. This is means in particular which field within uData is to be used.

Name	Value	Union Element	Packet Length	Usage
PNM_AP_CFG_PRM_ RECORD	0	tRecord	8 + 8 + Data Length	uData.tRecord contains a manufacture specific record parameter. See *PNM_AP_CFG_RECORD_DATA_T structure* (page 83) for details.
PNM_AP_CFG_PRM_ PDPORTDATACHECK	1	tParam.tPortDataCheck	8 + 272	uData.tParam.tPortDataCheck contains a PD Port Data check configuration to be written to a Port submodule. See *PNM_AP_CFG_PRM_PDPORTDATA_CHECK_T structure* (page 84) for details.
PNM_AP_CFG_PRM_ PDPORTDATAADJUST	2	tParam.tPortDataAdjust	8 + 36	uData.tParam.tPortDataAdjust contains a PD Port Data Adjust configuration to be written to a Port submodule. See *PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_T structure* (page 86) for details.
PNM_AP_CFG_PRM_ PDIRDATA_PDIRGLOBALDATA	3	tParam.tIrtGlobal	8 + 36 +16 *Number of Ports	uData.tParam.tIrtGlobal contains a PD IR Data Adjust Global configuration to be written to the interface submodule of a PROFINET Device. See PNM_AP_CFG_PRM_IRDATA_GLOBAL_T structure (page 90) for details.
PNM_AP_CFG_PRM_ PDIRDATA_PDIRFRAMEDATA	4	tParam.tIrtFrames	8 + 36	uData.tParam.tIrtFrames contains a PD IR Data Adjust Frame configuration to be written to an Interface submodule. See *PNM_AP_CFG_PRM_IRDATA_FRAME_T structure* (page 93) for details.
PNM_AP_CFG_PRM_ PDIRDATA_PDIRBEGINENDDATA	5	tParam.tIrtPhases	4 + 16 * Number of Phases	uData.tParam.tIrtPhases contains a PD IR Data Adjust BeginEnd configuration to be written to a Port submodule. See *PNM_AP_CFG_PRM_IRDATA_PHASES_T structure* (page 95) for details. Note: For structural reasons the PNM_AP_CFG_PRM_IRDATA_PHASES_T is associated with a Port Submodule. Nevertheless will the parameter write occur on the associated interface submodule (as defined by PROFINET Specification).
PNM_AP_CFG_PRM_ PDSYNCDATA	6	tParam.tSyncData	8 + 280	uData.tParam.tSyncData contains a Sync Data for SyncID 0 Configuration to be written to an Interface Submodule. See *PNM_AP_CFG_PRM_SYNC_T structure* (page 96) for details. Due to the limitations of the PROFINET Controller Firmware this parameter must describe a sync slave configuration.

© Hilscher, 2015–2017

Name	Value	Union Element	Packet Length	Usage
PNM_AP_CFG_PRM_PDINTFMRPDATACHECK	7	tParam.tIntfMRPDataCheck	8 + 4 +28 * MRP Instances	uData.tParam.tIntfMRPDataCheck contains a PD Interface MRP Data Check Configuration to be written to a Interface Submodule. See *PNM_AP_CFG_PRM_PDINTFMRPDATACHECK_T structure* (page 99) for details
PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST	8	tParam.tIntfMRPDataAdjust	8 + 4 +276 * MRP Instances	uData.tParam.tIntfMRPDataAdjust contains a PD Interface MRP Data Adjust Configuration to be written to a Interface Submodule. See *PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST_T structure* (page 101) for details.
PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST	9	tPortMRPDataAdjust	8 + 20	uData.tParam.tPortMRPDataAdjust contains a PD Port MRP Data Adjust Configuration to be written to a Port Submodule. See *PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST_T structure* (page 104) for details.
PNM_AP_CFG_PRM_PDPORTFODATACHECK	10	tPortFODataCheck	8 + 16	uData.tParam.tPortFODataCheck contains a PD Port FO Data Check configuration to be written to a Port Submodule. See *PNM_AP_CFG_PRM_PDPORTFODATACHECK_T structure* (page 105) for details
PNM_AP_CFG_PRM_PDPORTFODATAADJUST	11	tPortFODataAdjust	8 + 12	uData.tParam.tPortFODataAdjust contains a PD Port FO Data Adjust configuration to be written to a Port Submodule. See *PNM_AP_CFG_PRM_PDPORTFODATACHECK_T structure* (page 105) for details
PNM_AP_CFG_PRM_PDNCDATACHECK	12	tParam.tNCDataCheck	8 + 20	uData.tParam.tNCDataCheck contains a PD NC Data check configuration to be written to an Interface/Port Submodule. See *PNM_AP_CFG_PRM_PDNCDATACHECK_T structure* (page 108) for details.
PNM_AP_CFG_PRM_PDINTERFACEADJUST	13	tParam.tIntfAdjust	8 + 8	uData.tParam.tInfAdjust contains a PD Interface Adjust Configuration to be written to an Interface Submodule. See *PNM_AP_CFG_PRM_PDINTERFACEADJUST_T structure* (page 110) for details.
PNM_AP_CFG_PRM_PDINTERFACEFSUDATAADJUST	14	tParam.tIntfFsuDataAdjust	8 + 20	uData.tParam.tIntfFsuDataAdjust contains a PD Interface FSU Data Adjust Configuration to be written to an Interface Submodule. See *PNM_AP_CFG_PRM_PDINTERFACEFSUDATAADJUST_T structure* (page 111) for details.
PNM_AP_CFG_PRM_ARFSUDATAADJUST	15	tParam.tArFsuData	8 + 24	uData.tParam.tArFsuData contains an AR FSU Data Configuration to be written to a Device. See *PNM_AP_CFG_PRM_AR_FSU_PARAMETERS_T structure* (page 113) for details.
PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA	16	tParam.tIsochronousModeData	8 + 24	uData.tParam.tIsochronousModeData conatins an Isochronous Mode Data configuration to be written to a submodule.

© Hilscher, 2015–2017

3.1.8.2 PNM_AP_CFG_RECORD_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T          PNM_AP_CFG_RECORD_CNF_T;
```

Packet description

Structure PNM_AP_CFG_RECORD_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		
ulSrc	UINT32		
ulDestId	UINT32	0	
ulSrcId	UINT32	mirrored	
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x940D	PNM_AP_CMD_CFG_RECORD_CNF
ulExt	UINT32	0	
ulRoute	UINT32		Ignore

© Hilscher, 2015–2017

3.1.9 Configure topology service

This packet is used to configure the topology database within the PROFINET IO Controller. This information is required in order to support automatic name assignment.

3.1.9.1 PNM_AP_CFG_TOPO_REQ_T request

Packet structure reference

```
#define PNM_AP_CFG_TOPO_STRUCT_VERSION_1    (0x0001)

/** request packet */
typedef struct PNM_AP_CFG_TOPO_DATA_Ttag PNM_AP_CFG_TOPO_DATA_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_TOPO_DATA_Ttag
{

  /** @note the following rules shall apply if this Info contains topology information
about IO-Controller local Ports:
   *  - Device Handle shall be set to  0
   *  - Submodule handle shall be set to 0xFFF1 for Port 1 and to 0xFFF2 for Port 2
   *  */

  /** structure version of this structure */
  uint32_t                  ulStructVersion;
  /** IO-Device handle of 1. submodule to connect */
  PNM_AP_DEVICEHANDLE_T     usDeviceHandle1;
  /** submodule handle of the 1. IO-Device to be connected */
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle1;
  /** IO-Device handle of 2. submodule to connect */
  PNM_AP_DEVICEHANDLE_T     usDeviceHandle2;
  /** submodule handle of the 2. IO-Device to be connected */
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle2;
};

typedef struct PNM_AP_CFG_TOPO_REQ_Ttag PNM_AP_CFG_TOPO_REQ_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_TOPO_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T         tHead;
  PNM_AP_CFG_TOPO_DATA_T tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_CFG_TOPO_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility.
ulSrcId	UINT32	any	-
ulLen	UINT32	12	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x940E	PNM_AP_CMD_CFG_TOPO_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	any	-
Structure PNM_AP_CFG_IOC_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usDeviceHandle1	PNM_AP_DEVICEHANDLE_T	0, 1 … 128	Handle to Device on this side of link
usSubmoduleHandle1	PNM_AP_SUBMODULE_HANDLE_T	1 … 2048, 0xFFF1, 0xFFF2	Handle to submodule on this side of link
usDeviceHandle2	PNM_AP_DEVICEHANDLE_T	0, 1 … 128	Handle to Device on the other side of link
usSubmoduleHandle2	PNM_AP_SUBMODULE_HANDLE_T	1 … 2048, 0xFFF1, 0xFFF2	Handle to submodule on the other side of link

Parameter descriptions

Parameter ulStructVersion

Version of this structure.

Parameters usDeviceHandle1, usDeviceHandle2

The device handle parameters are a reference to the devices attached to the both sides of the network link. The value "0" shall be used to refer to the PROFINET IO Controller.

Parameters usSubmoduleHandle1, usSubmoduleHandle2

The submodule handle parameters are references to the port submodules attached to both sides of the network link. The handles must refer to a Port Submodule of the device. The value "0xFFF1" shall be used to refer to the PROFINET IO Controllers Port 1 and "0xFFF2" to refer to the PROFINET IO Controllers Port 2.

© Hilscher, 2015–2017

3.1.9.2 PNM_AP_CFG_TOPO_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T          PNM_AP_CFG_TOPO_CNF_T;
```

Packet description

Structure PNM_AP_CFG_TOPO_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x940F	PNM_AP_CMD_CFG_TOPO_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore

© Hilscher, 2015–2017

3.1.10 Download finished service

This service shall be used by the application finish the configuration process. The PROFINET IO Controller firmware will perform several consistency checks and activate the configuration. The bus will be activate if PNM_AP_IOC_FLAG_STARTMODE_AUTOMATIC has been used.

3.1.10.1 PNM_AP_DWNL_FIN_REQ_T request

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T   PNM_AP_DWNL_FIN_REQ_T;
```

Packet description

Structure PNM_AP_DWNL_FIN_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility.
ulSrcId	UINT32	any	-
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x9410	PNM_AP_CMD_CFG_DOWNLOAD_FINISH_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	any	-

© Hilscher, 2015–2017

3.1.10.2 PNM_AP_DWNL_FIN_CNF_T confirmation

Packet structure reference

```
typedef struct PNM_AP_DWNL_FIN_CNF_DATA_Ttag   PNM_AP_DWNL_FIN_CNF_DATA_T;
struct PNM_AP_DWNL_FIN_CNF_DATA_Ttag
{
  PNM_AP_DEVICEHANDLE_T     usDeviceHandle;
  PNM_AP_IOCR_HANDLE_T      usIocrHandle;
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle;
};

typedef struct PNM_AP_DWNL_FIN_CNF_Ttag   PNM_AP_DWNL_FIN_CNF_T;
struct PNM_AP_DWNL_FIN_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T         tHead;
  PNM_AP_DWNL_FIN_CNF_DATA_T  tData;
};
```

Packet description

Structure PNM_AP_DWNL_FIN_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		
ulSrc	UINT32		
ulDestId	UINT32	0	
ulSrcId	UINT32	mirrored	
ulLen	UINT32	6	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x9411	PNM_AP_CMD_CFG_DOWNLOAD_FINISH_CN
ulExt	UINT32	any	
ulRoute	UINT32	any	Ignore
Structure PNM_AP_DWNL_FIN_CNF_DATA_T			
usDeviceHandle	PNM_AP_DEVICEHANDLE_T	0 1 to 128	No error / error related to controller configuration itself Handle to AR causing the configuration error
usIocrHandle	PNM_AP_IOCR_HANDLE_T	0 0x1000 to 0x107F 0x2000 to 0x207F	No error / not an IOCR error Handle to IOCR causing the configuration error
usSubmoduleHandle	PNM_AP_SUBMODULE_HANDLE_T	0 1 to 2048 0xFFF0 0xFFF1 0xFFF2	No error / not a submodule error Handle to submodule causing the configuration error. Handle to controller PD interface submodule Handle to controller PD port 1 submodule Handle to controller PD port 2 submodule

Parameter descriptions

Parameters usDeviceHandle, usIocrHandle, usSubmoduleHandle

These parameters provide a reference to the configuration element which caused the configuration error.

© Hilscher, 2015–2017

3.1.11 Load Remanent service

The load remanent service is used by the application to restore remanent data after power on. Such data was received before from the PROFINET IO Controller by means of Store Remanent service. As the length of the service's request packet exceeds the size of the mailbox a fragmented packet transfer must be used with this service.

Note: This service must be used after the Download finished service and before switching the controller to Bus On. This implies that the PROFINET IO Controller must be configured for application controlled startup in Configure IO Controller service.

In case of database configuration, this service shall be used after the channel has been initialized.

3.1.11.1 PNM_AP_LOAD_REMANENT_REQ_T request

Packet structure reference

```
typedef struct PNM_AP_LOAD_REMANENT_REQ_DATA_Ttag PNM_AP_LOAD_REMANENT_REQ_DATA_T;

struct PNM_AP_LOAD_REMANENT_REQ_DATA_Ttag
{
  uint16_t              usLength;
  uint8_t               abData[];
};

typedef struct PNM_AP_LOAD_REMANENT_REQ_Ttag PNM_AP_LOAD_REMANENT_REQ_T;

struct PNM_AP_LOAD_REMANENT_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T             tHead;
  PNM_AP_LOAD_REMANENT_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_LOAD_REMANENT_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility.
ulSrcId	UINT32	any	-
ulLen	UINT32	8194	Packet data length in bytes
ulId	UINT32	any	Increasing value according to fragmented transfer definition.-
ulSta	UINT32	0	-
ulCmd	UINT32	0x9442	PNM_AP_CMD_LOAD_REMANENT_REQ
ulExt	UINT32	any	Values of RCX_PACKET_SEQ_ defines to manage fragmented transfer
ulRoute	UINT32	any	-
Structure PNM_AP_LOAD_REMANENT_REQ_DATA_T			
usLength	UINT16	8192	Length of the remanent data to restore
abData[]	UINT8[]	any	Remanent data to restore

© Hilscher, 2015–2017

Parameter descriptions

Parameter `usLength`

The length of the remanent data to be restored. This field should use the same value as received in Store Remanent service.

Parameter `abData[]`

The remanent data to be restored.

3.1.11.2 PNM_AP_LOAD_REMANENT_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T          PNM_AP_LOAD_REMANENT_CNF_T;
```

Packet description

Structure `PNM_AP_LOAD_REMANENT_CNF_T`			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure `PNM_AP_PCK_HEADER_T`			
`ulDest`	UINT32		-
`ulSrc`	UINT32		-
`ulDestId`	UINT32	0	-
`ulSrcId`	UINT32	Mirrored	-
`ulLen`	UINT32	0	Packet data length in bytes
`ulId`	UINT32	Mirrored	Validate according to fragmented packet transfer sequence
`ulSta`	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
`ulCmd`	UINT32	0x9443	`PNM_AP_CMD_LOAD_REMANENT_CNF`
`ulExt`	UINT32	Mirrored	Validate according to fragmented packet transfer sequence
`ulRoute`	UINT32	Any	Ignore

© Hilscher, 2015–2017

3.1.12 Configure record structures

This section describes the structure of the various record related configuration structures. These structures are identical for configuring PROFINET IO Controller and PROFINET IO Device parameters with *Configure IO Controller parameter service* on page 41 and *Configure record service* on page 71.

3.1.12.1 PNM_AP_CFG_RECORD_DATA_T structure

Structure reference

```
#define PNM_AP_CFG_RECORD_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_RECORD_DATA_Ttag PNM_AP_CFG_RECORD_DATA_T;
struct PNM_AP_CFG_RECORD_DATA_Ttag
{
  uint32_t               ulStructVersion;
  uint16_t               usIndex;
  uint16_t               usTransferSequence;
  uint8_t                abRecordData[ARRAYS_OF_LENGTH_ZERO];
};
```

Structure description

Structure PNM_AP_CFG_RECORD_DATA_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_RECORD_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usIndex	UINT16	1 … 0x7FFF	The index of the manufacturer specific record object
usTransferSequence	UINT16	any	The transfer sequence for the record object
abRecordData[]	UINT8[]	any	Record data

Parameter descriptions

Parameter ulStructVersion

Version of this structure.

Parameter usIndex

The record object index of the record object to write. This parameter is provided in the GSDML file.

Parameter usTransferSequence

This parameter can be used to configure the order in which the record object shall be written. This value is provided in the GSDML file of the PROFINET IO Device. The default value is zero.

Parameter abRecordData[]

This parameter contains the payload (the actual record data to be written). The length of the data is determined from the packet length.

© Hilscher, 2015–2017

3.1.12.2 PNM_AP_CFG_PRM_PDPORTDATA_CHECK_T structure

Structure reference

```
enum PNM_AP_CFG_PRM_PORT_MAUTYPE_Etag
{
  PNM_AP_CFG_IOC_PRM_PORT_MAUTYPE_PORT_DISABLED  = 0,
  PNM_AP_CFG_IOC_PRM_PORT_MAUTYPE_100BASETX      = 16,
};
typedef enum PNM_AP_CFG_PRM_PORT_MAUTYPE_Etag PNM_AP_CFG_PRM_PORT_MAUTYPE_E;

typedef enum
{
  PNM_AP_CFG_PRM_PDPORTDATA_CHECK_ENABLE_FLAG_PEER_ID             = 0x0001,
  PNM_AP_CFG_PRM_PDPORTDATA_CHECK_ENABLE_FLAG_LINEDELAY           = 0x0002,
  PNM_AP_CFG_PRM_PDPORTDATA_CHECK_ENABLE_FLAG_MAUTYPE             = 0x0004,
  PNM_AP_CFG_PRM_PDPORTDATA_CHECK_ENABLE_FLAG_LINKSTATE           = 0x0008,
  PNM_AP_CFG_PRM_PDPORTDATA_CHECK_ENABLE_FLAG_SYNCMODE            = 0x0010,
  PNM_AP_CFG_PRM_PDPORTDATA_CHECK_ENABLE_FLAG_MAUTYPEMODE         = 0x0020,
} PNM_AP_CFG_PRM_PDPORTDATA_CHECK_ENABLE_FLAGS;

#define PNM_AP_CFG_PRM_PDPORTDATACHECK_VERSION_1      (0x0001)

typedef struct PNM_AP_CFG_PRM_PDPORTDATA_CHECK_Ttag
PNM_AP_CFG_PRM_PDPORTDATA_CHECK_T;struct PNM_AP_CFG_PRM_PDPORTDATA_CHECK_Ttag
{
  uint32_t ulStructversion;
  uint32_t ulEnableFlag;
  uint8_t  abPeerId[255];
  uint8_t  bReserved;
  uint32_t ulLineDelay;
  uint16_t usMauType;
  uint16_t usSyncMode;
};
```

Structure description

Structure PNM_AP_CFG_PRM_PDPORTDATA_CHECK_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_PDPORTDATA_CHECK_T			
ulStructVersion	UINT32	1	Structure version of this structure
ulEnableFlag	UINT32	0 ... 63	Bitmask of Checks to activate
abPeerId[]	UINT8[]		The alias name of the peer port expected at this port
bReserved	UINT8	0	Padding. Set to zero.
ulLineDelay	UINT32		The maximum line delay
usMauType	UINT16	16	Expected MAU Type of this port
usSyncMode	UINT16	0 … 3	Synchronization Mode check bitmask

Parameter descriptions

Parameter ulStructVersion

Version of this structure.

© Hilscher, 2015–2017

Parameter ulEnableFlag

This bitmask which of the following settings should be applied to the device. The following bits are defined

Name	Numerical Value	Usage
PNM_AP_CFG_PRM_PDPORTDATA_CHECK_ENABLE_FLAG_PEER_ID	0x0001	Validate the remote peers station name and port. Associated parameter abPeerId.
PNM_AP_CFG_PRM_PDPORTDATA_CHECK_ENABLE_FLAG_LINEDELAY	0x0002	Validate the measured line delay against reference value given by parameter ulLineDelay.
PNM_AP_CFG_PRM_PDPORTDATA_CHECK_ENABLE_FLAG_MAUTYPE	0x0004	Validate the MAU type against the reference value given by parameter usMauType.
PNM_AP_CFG_PRM_PDPORTDATA_CHECK_ENABLE_FLAG_LINKSTATE	0x0008	Check if the port detected a network link
PNM_AP_CFG_PRM_PDPORTDATA_CHECK_ENABLE_FLAG_SYNCMODE	0x0010	Check if the remote peer is using the synchronization mode given by parameter usSyncMode.
PNM_AP_CFG_PRM_PDPORTDATA_CHECK_ENABLE_FLAG_MAUTYPEMODE	0x0020	Validate the MAU type of this port against the MAU type reported by the remote port

Parameters abPeerId[255]

The expected peer port alias name of this port. The value is specified using the standard PROFINET alias name specification. E.g. if the port shall verify that it is connected to Port 1 of a device with a name of station "mydevice" the value of this parameter should be set to "port-001.mydevice"

Parameter ulLineDelay

The maximum line delay allowed on this port in nanoseconds.

Parameter usMauType

The expected MAU type of this port.

Name	Numerical Value	Meaning
PNM_AP_CFG_IOC_PRM_PORT_MAUTYPE_100BASETX	16	Expect a 100 MBit Full Duplex on this port.

Parameter usSyncMode

This parameter is a bitmask activating peer checks related to synchronization

Name	Numerical Value	Meaning
-	0x0001	Validate measured line delay of peer port against our measurement (Maximum deviation 50 ns)
-	0x0002	Validate peer's sync masters source mac address against our sync master's source mac address

© Hilscher, 2015–2017

3.1.12.3 PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_T structure

Structure reference

```
typedef enum
{
  PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_ENABLE_FLAG_DOMAIN_BOUNDARY        = 0x0001,
  PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_ENABLE_FLAG_MULTICAST_BOUNDARY     = 0x0002,
  PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_ENABLE_FLAG_PEERTOPEER_BOUNDARY    = 0x0004,
  PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_ENABLE_FLAG_DCP_BOUNDARY           = 0x0008,
  PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_ENABLE_FLAG_MAUTYPE                = 0x0010,
  PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_ENABLE_FLAG_LINKSTATE              = 0x0020,
  PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_ENABLE_FLAG_PREAMBLELENGTH         = 0x0040,
} PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_ENABLE_FLAGS;

typedef enum
{
  PNM_AP_CFG_PRM_PEER_TO_PEER_BOUNDARY_PASS                 = 0x0000,
  PNM_AP_CFG_PRM_PEER_TO_PEER_BOUNDARY_BLOCK_LLDP           = 0x0001,
  PNM_AP_CFG_PRM_PEER_TO_PEER_BOUNDARY_BLOCK_PTCP_DELAY     = 0x0002,
  PNM_AP_CFG_PRM_PEER_TO_PEER_BOUNDARY_BLOCK_PTCP_PATH      = 0x0004,
} PNM_AP_CFG_PRM_PEER_TO_PEER_BOUNDARY_E;

typedef enum
{
  PNM_AP_CFG_PRM_DCP_BOUNDARY_PASS               = 0x0000,
  PNM_AP_CFG_PRM_DCP_BOUNDARY_BLOCK_IDENTIFY     = 0x0001,
  PNM_AP_CFG_PRM_DCP_BOUNDARY_BLOCK_HELLO        = 0x0002,
} PNM_AP_CFG_PRM_DCP_BOUNDARY_E;

enum PNM_AP_CFG_PRM_PORT_MAUTYPE_Etag
{
  PNM_AP_CFG_IOC_PRM_PORT_MAUTYPE_PORT_DISABLED  = 0,
  PNM_AP_CFG_IOC_PRM_PORT_MAUTYPE_100BASETX      = 16,
};
typedef enum PNM_AP_CFG_PRM_PORT_MAUTYPE_Etag PNM_AP_CFG_PRM_PORT_MAUTYPE_E;

enum PNM_AP_CFG_PRM_PORT_PREAMBLE_LENGTH_Etag
{
  PNM_AP_CFG_IOC_PRM_PORT_PREAMBLE_LENGTH_7_OCTETS = 7,
  PNM_AP_CFG_IOC_PRM_PORT_PREAMBLE_LENGTH_1_OCTET  = 1,
};
typedef enum PNM_AP_CFG_PRM_PORT_PREAMBLE_LENGTH_Etag
PNM_AP_CFG_PRM_PORT_PREAMBLE_LENGTH_E;

#define PNM_AP_CFG_PRM_PDPORTDATAADJUST_VERSION_1      (0x0001)

typedef struct PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_Ttag PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_Ttag
{
  uint32_t ulStructversion;
  uint32_t ulEnableFlag;
  uint32_t ulDomainBoundaryIngress;
  uint32_t ulDomainBoundaryEgress;
  uint32_t ulMulticastBoundary;
  uint32_t ulPeerToPeerBoundary;
  uint32_t ulDCPBoundary;
  uint32_t ulFiberOpticCableType;
  uint32_t ulFiberOpticType;
  uint16_t usMAUType;
  uint16_t usLinkState;
  uint16_t usPreambleLength;
};
```


© Hilscher, 2015–2017

Structure description

Structure PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_T			Type: Structure
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_T			
ulStructVersion	UINT32	1	Structure version of this structure
ulEnableFlag	UINT32		Bitmask of Adjustments to active
ulDomainBoundaryIngress	UINT32	0, 1	Defines If the port should block incoming PTCP sync frames
ulDomainBoundaryEgress	UINT32	0, 1	Defines If the port should block outgoing PTCP sync frames
ulMulticastBoundary	UINT32		Bitmask defining which multicast frames shall be blocked by this port
ulPeerToPeerBoundary	UINT32	0 … 7	Bitmask defining which peer to peer multicast frames shall be blocked by this port
ulDCPBoundary	UINT32	0 … 3	Bitmask defining which DCP Frames should be blocked by this port
usMauType	UINT16		MAU Type to force on this port, supported values see GSDML of IO Device if used to configure IO Controller only value 16 is supported
usLinkState	UINT16	2	Link State to force on this port.
usPreambleLength	UINT16	7	Ethernet Preamble Length

Parameter descriptions

Parameter ulStructVersion

Version of this structure.

Parameter ulEnableFlag

This bitmask which of the following settings should be applied to the device.

Name	Numerical Value	Usage
PNM_AP_CFG_PRM_PDPORTDATA_ ADJUST_ENABLE_FLAG_ DOMAIN_BOUNDARY	0x0001	Configure Domain Boundary specified by parameters ulDomainBoundaryIngress and ulDomainBoundaryEgress for this port.
PNM_AP_CFG_PRM_PDPORTDATA_ ADJUST_ENABLE_FLAG_ MULTICAST_BOUNDARY	0x0002	Configure Multicast Boundary specified by ulMulticastBoundary for this port.
PNM_AP_CFG_PRM_PDPORTDATA_ ADJUST_ENABLE_FLAG_ PEERTOPEER_BOUNDARY	0x0004	Configure PeerToPeer Boundary specified by parameter ulPeerToPeerBoundary for this port
PNM_AP_CFG_PRM_PDPORTDATA_ ADJUST_ENABLE_FLAG_ DCP_BOUNDARY	0x0008	Configure DCP Boundary specified by parameter ulDCPBoundary for this port.
PNM_AP_CFG_PRM_PDPORTDATA_ ADJUST_ENABLE_FLAG_MAUTYPE	0x0010	Configure MAU Type specified by parameter usMauType for this port.
PNM_AP_CFG_PRM_PDPORTDATA_ ADJUST_ENABLE_FLAG_LINKSTATE	0x0020	Configure Link State as specified by parameter usLinkState for this port.
PNM_AP_CFG_PRM_PDPORTDATA_ ADJUST_ENABLE_FLAG_ PREAMBLELENGTH	0x0040	Configure the Ethernet frame preamble length as specified by parameter usPreambleLength.

© Hilscher, 2015–2017

Parameters `ulDomainBoundaryIngress`, `ulDomainBoundaryEgress`

These two parameters are used to configure a frame filter on the corresponding port. The port will block incoming (ulDomainBoundaryIngress) and/or outgoing (ulDomainBoundaryEgress) PTCP frames if the parameter is set to "1".

Parameter `ulMulticastBoundary`

This parameter configures a frame filter for the port. The value is interpreted as an bitmask where each bit 0 to 31 represents one multicast MAC address from the range 01:0E:CF:00:02:00 to 01:0E:CF:00:02:1F. The port will block any outgoing frame with the destination MAC equal to the MAC address associated with the corresponding bit if the bit is set.

Parameter `ulPeerToPeerBoundary`

This parameter configures a frame filter for Peer To Peer Multicast frames. The value is interpreted as a bitmask where each bit corresponds to a multicast MAC address. The port will block outgoing frames if the corresponding bit is set. The following values are defined.

Name	Numerical Value	Usage
PNM_AP_CFG_PRM_PEER_TO_PEER_BOUNDARY_BLOCK_LLDP	0x00000001	Block LLDP Frames
PNM_AP_CFG_PRM_PEER_TO_PEER_BOUNDARY_BLOCK_PTCP_DELAY	0x00000002	Block PTCP Delay Frames
PNM_AP_CFG_PRM_PEER_TO_PEER_BOUNDARY_BLOCK_PTCP_PATH	0x00000003	Block PATH Delay Frames

Parameter `ulDCPBoundary`

This parameter configures a frame filter for DCP frames. The value is interpreted as a bitmask where each bit corresponds to a DCP Multicast MAC address. The port will block outgoing frames if the corresponding bit is set. The following values are defined.

Name	Numerical Value	Usage
PNM_AP_CFG_PRM_DCP_BOUNDARY_BLOCK_IDENTIFY	0x0001	Block DCP Identify with Mac Address 01:0E:CF:00:00:00
PNM_AP_CFG_PRM_DCP_BOUNDARY_BLOCK_HELLO	0x0002	Block DCP Hello with Mac Address 01:0E:CF:00:00:01

© Hilscher, 2015–2017

Parameter usMauType, usLinkState

These parameters can be used to configure the connection type to use on this port. Without this setting, the port will use autonegotiation to determine the optimum connection type automatically.

Note: Only one of the parameters is used at the same time.

In order to disable the port, set usLinkState to the value 2. The following values are allowed for the MAU type and Link State.

usMauType		**usLinkState**		
Name	Numerical Value	Name	Numerical Value	Meaning
PNM_AP_CFG_IOC_PRM_PORT_ MAUTYPE_100BASETX	16	-	-	Use 100 MBit Full Duplex on this port.
-	-	-	2	Disable the Port

Parameter usPreambleLength

This parameter can be used to configure the Ethernet frame preamble length. The default value is 7.

© Hilscher, 2015–2017

3.1.12.4 PNM_AP_CFG_PRM_IRDATA_GLOBAL_T structure

Structure reference

```
#define PNM_AP_CFG_PRM_MAX_PORTS                                (4)

#define PNM_AP_CFG_IOC_PRM_IRDATA_GLOBAL_STRUCT_VERSION_1    (0x0001)

#define PNM_AP_CFG_IOD_PRM_IRDATA_GLOBAL_STRUCT_VERSION_1    (0x0001)
#define PNM_AP_CFG_IOD_PRM_IRDATA_GLOBAL_STRUCT_VERSION_2    (0x0002)

enum PNM_AP_CFG_FRAME_DATA_PROP_Etag
{
  PNM_AP_CFG_FRAME_PROP_FORWARDING_MODE_ABSOLUTE   = 0x00000000,
  PNM_AP_CFG_FRAME_PROP_FORWARDING_MODE_RELATIVE   = 0x00000001,

  PNM_AP_CFG_FRAME_PROP_FAST_FORWARDING_LEGACY     = 0x00000000,
  PNM_AP_CFG_FRAME_PROP_FAST_FORWARDING_RTC3_MCAST = 0x00000002,
  PNM_AP_CFG_FRAME_PROP_FAST_FORWARDING_MCAST      = 0x00000004,
};
typedef enum PNM_AP_CFG_FRAME_DATA_PROP_Etag PNM_AP_CFG_FRAME_DATA_PROP_E;

typedef struct PNM_AP_CFG_PRM_IRDATA_PORT_Ttag
PNM_AP_CFG_PRM_IRDATA_PORT_T;structPNM_AP_CFG_PRM_IRDATA_PORT_Ttag
{
  uint32_t ulMaxPortTxDelay;
  uint32_t ulMaxPortRxDelay;
  uint32_t ulMaxLineRxDelay;
  uint32_t ulYellowTime;
};

typedef struct PNM_AP_CFG_PRM_IRDATA_GLOBAL_Ttag
PNM_AP_CFG_PRM_IRDATA_GLOBAL_T;structPNM_AP_CFG_PRM_IRDATA_GLOBAL_Ttag
{
  uint32_t                     ulStructVersion;
  PNM_UUID_T                   tIRDataUuid;
  uint32_t                     ulMaxBridgeDelay;
  uint16_t                     usStartOfRedFrameId;
  uint16_t                     usEndOfRedFrameId;
  uint16_t                     usNumFrames;
  uint8_t                      bNumPorts;
  uint8_t                      bNumPhases;
  uint32_t                     ulFrameDataProperties;
  PNM_AP_CFG_PRM_IRDATA_PORT_T atPorts[ARRAYS_OF_LENGTH_ZERO];
} ;
```


© Hilscher, 2015–2017

Structure description

Structure PNM_AP_CFG_PRM_IRDATA_GLOBAL_T			Type: Structure
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_IRDATA_GLOBAL_T			
ulStructVersion	UINT32	1, 2	Structure version of this structure
tIRDataUuid	UUID		UUID of the IRT Parameter set
ulMaxBridgeDelay	UINT32		Maximum Bridge Delay
usStartOfRedFrameId	UINT16	0x0100 to 0x013F	The first frame id used for IRT
usEndOfRedFrameId	UINT16	0x0200 to 0x023F	The last frame id used for IRT
usNumFrames	UINT16	1 … 128	Number of Frames in IR Data set
bNumPorts	UINT8		Number of ports of the device as specified in GSDML
bNumPhases	UINT8	1 … 16	Number of IRT Phases
ulFrameDataProperties	UINT32	0 … 7	Common IRT Frame Properties
atPorts[]			Array of Port Description structures (one for each port)
Structure PNM_AP_CFG_PRM_IRDATA_PORT_T			
ulMaxPortTxDelay	UINT32		Maximum Port TX Delay as specified in GSDML
ulMaxPortRxDelay	UINT32		Maximum Port RX Delay as specified in GSDML
ulMaxLineRxDelay	UINT32		Maximum Line RX Delay
ulYellowTime	UINT32	125000	Length of Yellow Interval

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used to specify the encoding of the PD IR Data Record. The following values are defined:

Name	Numerical Value	Meaning
PNM_AP_CFG_IOC_PRM_IRDATA_GLOBAL_STRUCT_VERSION_1	0x0001	To be used when the parameter is associated with the PROFINET Controller Firmware itself
PNM_AP_CFG_IOD_PRM_IRDATA_GLOBAL_STRUCT_VERSION_1	0x0001	To be used when the parameter is associated with a PROFINET Specification 2.2 Device
PNM_AP_CFG_IOD_PRM_IRDATA_GLOBAL_STRUCT_VERSION_2	0x0002	To be used when the parameter is associated with a PROFINET Specification 2.3 Device

Parameter tIRDataUuid

This parameter defines the UUID associated with the IRT domain. All devices of an IRT domain have are parameterized with the same UUID.

Parameter ulMaxBridgeDelay

The maximum bridge delay of the IRT Switch. This value shall be derived from the PROFINET Device's GSDML file. An additional delay auf 100 ns must be added to the GSDML value for "Relative Forwarding Mode".

© Hilscher, 2015–2017

Parameters usStartOfRedFrameId and usEndOfRedFrameId

These parameters define the range of frame ids to be forwarded synchronously. Due to the limitations of the PROFINET IO Controller firmware the parameter usStartOfRedFrameId should be in the range 0x0100 to 0x013F and the parameter usEndOfRedFrameIdvalues should be in the range of 0x0200 and 0x023F. See *Configuration of PROFINET IO Controller* on page 13 for details.

Parameter usNumFrames

This parameter defines the number of frames configured for the associated IRT switch.

Parameter bNumPorts

This parameter configures the number of ports of the associated IRT switch.

Parameter bNumPhases

This parameter specifies the number of phases used in the IRT domain. A maximum of 16 phases is supported.

Parameter ulFrameDataProperties

This parameter is a bitmask defining additional properties of IR frame set configuration

Name	Numerical Value	Meaning
Bit 0 - Forwarding Mode	-	-
PNM_AP_CFG_FRAME_PROP_FORWARDING_MODE_ABSOLUTE	0x00000000	The device is an absolute forwarder.
PNM_AP_CFG_FRAME_PROP_FORWARDING_MODE_RELATIVE	0x00000001	The device is a relative forwarder
Bit 1 to 2 - Fast Forwarding Mode	-	-
PNM_AP_CFG_FRAME_PROP_FAST_FORWARDING_LEGACY	0x00000000	The switch shall forward according PROFINET Specification V2.2 using Interface Destination Mac Address
PNM_AP_CFG_FRAME_PROP_FAST_FORWARDING_RTC3_MCAST	0x00000002	The switch shall forward according PROFINET Specification V2.3 using RTC3 Multicast Destination Mac Addresses
PNM_AP_CFG_FRAME_PROP_FAST_FORWARDING_MCAST	0x00000004	The switch shall forward according PROFINET Specification V2.3 using Fast Forwarding Multicast Destination Mac Addresses
Bit 3 to 4 - Fragmentation Mode (not supported)	-	-

Parameters ulMaxPortTxDelay, ulMaxPortRxDelay, ulMaxLineRxDelay, ulYellowTime

These parameters describe device specific and engineering related port timings. The Port Delay values are specified in the GSDML file of the device and used as input to the IRT planning algorithm. The Line Delay and Yellow Time Parameters are output values from the planning algorithm. For IRT/PTCP SendClocks greater or equal to 8 (250 µs) the Yellow Time is fixed to 125000 ns. All parameters use a time base of one nanosecond.

© Hilscher, 2015–2017

3.1.12.5 PNM_AP_CFG_PRM_IRDATA_FRAME_T structure

Structure reference

```
enum PNM_AP_CFG_FRAME_DETAILS_Etag
{
  PNM_AP_CFG_FRAME_NO_SYSNC_FRAME        = 0x00,
  PNM_AP_CFG_FRAME_PRIMARY_SYNC_FRAME    = 0x01,
  PNM_AP_CFG_FRAME_SECONDARY_SYNC_FRAME  = 0x02,
};
typedef enum PNM_AP_CFG_FRAME_DETAILS_Etag PNM_AP_CFG_FRAME_DETAILS_E;

#define PNM_AP_CFG_IOC_PRM_IRDATA_FRAME_STRUCT_VERSION_1    (0x0001)
#define PNM_AP_CFG_IOD_PRM_IRDATA_FRAME_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_PRM_IRDATA_FRAME_Ttag
PNM_AP_CFG_PRM_IRDATA_FRAME_T;structPNM_AP_CFG_PRM_IRDATA_FRAME_Ttag
{
  uint32_t               ulStructVersion;
  uint16_t               usEntryId;
  uint16_t               usFrameId;
  uint16_t               usCSDULength;
  uint16_t               usPhase;
  uint16_t               usReduction;
  uint8_t                bFrameDetails;
  uint8_t                bRxPort;
  uint32_t               ulFrameSendOffset;
  uint8_t                atTxPortGroups[16];
};
```

Structure description

Structure PNM_AP_CFG_PRM_IRDATA_FRAME_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_IRDATA_FRAME_T			
ulStructVersion	UINT32	1	Structure version of this structure
usEntryId	UINT16	0 … 127	Index of the frame with PD IR Data Record
usFrameId	UINT16		Frame ID of the associated frame
usCSDULength	UINT16	40 …1440	Length of the C_SDU of the associated frame
usPhase	UINT16	1 … 16	Phase in which the frame shall be forwarded
usReduction	UINT16	1 … 16	Reduction of the frame
bFrameDetails	UINT8	0 ... 3	Some details about the frame (Bitmask)
bRxPort	UINT8		The port where the frame shall be received from
ulFrameSendOffset	UINT32	1 to 3999999	The frame send offset related to phase start
atTxPortGroups[]	UINT8		Bitmask defining to which ports the frame shall be forwarded

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used for future extensions.

Parameter usEntryId

A reference to the frame within the PD IR Data Set. Each frame of a PD IR Data set must be associated an unique entry id. The value must be smaller than usNumFrames parameter from *PNM_AP_CFG_PRM_IRDATA_GLOBAL_T structure* on page 90. This value is used internally in the PROFINET Controller Firmware to organize the PD IR Data set.

© Hilscher, 2015–2017

Parameter `usFrameId`

The PROFINET Frame ID of the frame to forward / inject.

Parameter `usCSDULength`

The length of the frames C_SDU part. That is the length of the cyclically exchanged process data and states padded to at least 40 byte.

Parameters `usPhase`, `usReduction` and `ulFrameSendOffset`

The parameters describe the timing properties of the frame. The reduction defines the interval between two subsequent frames - the cyclic data exchange interval. The time base is the time defined by `usSendClockFactor` from *PNM_AP_CFG_PRM_SYNC_T structure* on page 96. If a reduction greater than one is used, it is required to define in which phase the frame shall be sent. The parameter usPhase defines the phase of a frame. It must be lower or equal than the reduction. The parameter ulFrameSendOffset defines the time offset relative to the start of a phase when the frame shall be sent/forwarded. Time base of this parameter is one nanosecond.

Parameter `bFrameDetails`

This parameter provides additional information about the frame. It is a bitmask defined as follows:

Name	Numerical Value	Meaning
Bits 0 to 1 - Syncframe		
`PNM_AP_CFG_FRAME_NO_SYSNC_FRAME`	0x00	Frame is no PTCP Sync Frame
`PNM_AP_CFG_FRAME_PRIMARY_SYNC_FRAME`	0x01	Frame is a primary Sync Frame
`PNM_AP_CFG_FRAME_SECONDARY_SYNC_FRAME`	0x02	Frame is a secondary Sync Frame

Parameters `bRxPort`, `atTxPortGroups[]`

This parameters describe the routing of the frame. Parameter bRxPort is the source of frame. A value of zero corresponds to the local port, a frame which is injected by the device itself. The parameter atTxPortGroups is a bitmask describing the ports the frame shall be forwarded to. The 0th bit corresponds to the device's local receive port.

© Hilscher, 2015–2017

3.1.12.6 PNM_AP_CFG_PRM_IRDATA_PHASES_T structure

Structure reference

```
#define PNM_AP_CFG_PRM_MAX_BEGIN_END_ASSIGNMENTS        (16)

typedef struct PNM_AP_CFG_PRM_BEGIN_END_ASSIGNMENT_Ttag
PNM_AP_CFG_PRM_BEGIN_END_ASSIGNMENT_T;struct PNM_AP_CFG_PRM_BEGIN_END_ASSIGNMENT_Ttag
{
  uint32_t     ulTxRedBegin;
  uint32_t     ulTxGreenBegin;
  uint32_t     ulRxRedBegin;
  uint32_t     ulRxGreenBegin;
};

#define PNM_AP_CFG_IOC_PRM_IRDATA_PHASES_STRUCT_VERSION_1    (0x0001)
#define PNM_AP_CFG_IOD_PRM_IRDATA_PHASES_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_PRM_IRDATA_PHASES_Ttag PNM_AP_CFG_PRM_IRDATA_PHASES_T;struct
PNM_AP_CFG_PRM_IRDATA_PHASES_Ttag
{
  uint32_t                                ulStructVersion;
  PNM_AP_CFG_PRM_BEGIN_END_ASSIGNMENT_T     atBeginEnd[ARRAYS_OF_LENGTH_ZERO];
} ;
```

Structure description

Structure PNM_AP_CFG_PRM_IRDATA_PHASES_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_IRDATA_PHASES_T			
ulStructVersion	UINT32	1	Structure version of this structure
atBeginEnd[]			Array of Begin End Definitions define the Phase configuration of one port.
Structure PNM_AP_CFG_PRM_BEGIN_END_ASSIGNMENT_T			
ulTxRedBegin	UINT32	0 to 3999999	Transmit Begin of Red Interval
ulTxGreenBegin	UINT32	0 to 3999999	Transmit End of Red Interval
ulRxRedBegin	UINT32	0 to 4000000	Receive Begin of Red Interval
ulRxGreenBegin	UINT32	0 to 3999999	Receive End of Red Interval

Parameter descriptions

Parameter `ulStructVersion`

Version of this structure. Used for future extensions.

Parameter `atBeginEnd[]`

This parameter contains an array of phase descriptions for one port. The array must contain bNumPhases entries. (See *PNM_AP_CFG_PRM_IRDATA_GLOBAL_T structure* on page 90)

Parameters `ulTxRedBegin`, `ulTxGreenBegin`, `ulRxRedBegin`, `ulRxGreenBegin`

These parameters describe the timing of one phase of the port. The values are the output of the planning algorithm. The time base is one nanosecond.

© Hilscher, 2015–2017

3.1.12.7 PNM_AP_CFG_PRM_SYNC_T structure

Structure reference

```
#define PNM_AP_CFG_IOC_PRM_SYNC_STRUCT_VERSION_1    (0x0001)
#define PNM_AP_CFG_IOD_PRM_PDSYNCDATA_VERSION_1     (0x0001)

enum PNM_AP_CFG_PRM_SYNC_SYNCPROP_FLAGS_Etag
{
  PNM_AP_CFG_PRM_SYNC_SYNCPROP_ROLE_SLAVE  = 0x00000001,
  PNM_AP_CFG_PRM_SYNC_SYNCPROP_ROLE_MASTER = 0x00000002,
};
typedef enum PNM_AP_CFG_PRM_SYNC_SYNCPROP_FLAGS_Etag
PNM_AP_CFG_PRM_SYNC_SYNCPROP_FLAGS_E;

typedef struct PNM_AP_CFG_PRM_SYNC_Ttag PNM_AP_CFG_PRM_SYNC_T;
struct PNM_AP_CFG_PRM_SYNC_Ttag
{
  uint32_t               ulStructVersion;
  PNM_UUID_T             tSyncDomainUuid;
  uint16_t               usSyncProperties;
  uint16_t               usSendClockFactor;
  uint32_t               ulPllWindow;
  uint32_t               ulSyncSendFactor;
  uint16_t               usPtcpTimeoutFactor;
  uint16_t               usPtcpTakeoverTimeoutFactor;
  uint16_t               usPtcpMasterStartupTime;
  uint8_t                bPtcpMasterPrio1;
  uint8_t                bPtcpMasterPrio2;
  uint8_t                abSubdomainName[240];
} ;
```

Structure description

Structure PNM_AP_CFG_PRM_SYNC_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_SYNC_T			
ulStructVersion	UINT32	1	Structure version of this structure
tSyncDomainUuid	UUID		UUID of the synchronization domain
usSyncProperties	UINT16	1, 2	Synchronization Properties
usSendClockFactor	UINT16	8 … 128	Sync Domain Send Clock
ulPllWindow	UINT32	50 … 10000	Max PLL Sync Deviation
ulSyncSendFactor	UINT32	1 … 2764800000	Reduction of the Sync Frame
usPtcpTimeoutFactor	UINT16	0 … 511	Sync Frame Timeout Factor
usPtcpTakeoverTimeoutFactor	UINT16	0 … 511	Sync Master Takeover Timeout Factor
usPtcpMasterStartupTime	UINT16	0 … 300	Sync Master Startup Time
bPtcpMasterPrio1	UINT8	0 ... 63	Sync Master Priority 1
bPtcpMasterPrio2	UINT8	255	Sync Master Priority 2
abSubdomainName[]	UINT8[]		Synchronization Domain Name

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used for future extensions.

© Hilscher, 2015–2017

Parameter `tSyncDomainUuid`

This parameter defines the UUID associated with the synchronization domain. All devices in one domain must be parameterized with the same synchronization domain.

Parameter `usSyncProperties`

This parameter is a bitmask defining the synchronization properties of the associated device. The following values are defined:

Name	Numerical Value	Meaning
PNM_AP_CFG_PRM_SYNC_SYNCPROP_ROLE_SLAVE	0x0001	The associated device is configured as synchronization slave. To be used for a PROFINET Device.
PNM_AP_CFG_PRM_SYNC_SYNCPROP_ROLE_MASTER	0x0002	The associated device is configured as synchronization master. To be used for the PROFINET Controller

Parameter `usSendClockFactor`

This parameter defines the Base Clock of the Synchronization Domain. This is the length of a single IRT phase and thus defines the minimum cycle time for process data exchange. The value is encoded in units of 31.25 µs. (1ms corresponds to a factor of 32)

Parameter `ulPllWindow`

This parameter defines the maximum allowed deviation of a synchronization slave's local PLL from the synchronization master. If the deviation exceeds the window the synchronization will be regarded as lost and restarted. The value is encoded in units of 1ns. The proper value is described within the GSDML of a PROFINET Device. The value has no meaning for a Synchronization Master.

Parameter `ulSyncSendFactor`

This parameter parameterizes the frequency of the synchronization frames exchanged on the bus. The time base for this parameter is the length of one phase configured by usSendClockFactor The synchronization master will issue a synchronization frame every `ulSyncSendFactor` phases while the synchronization slave expects a synchronization frame every `ulSyncSendFactor` phases. The default value for this parameter is 30.

Parameter `usPtcpTimeoutFactor`

This parameter defines the synchronization timeout. A synchronization slave will regard the synchronization master as lost if no synchronization frame has been received within the specified interval. The time base of this parameter is defined by `ulSyncSendFactor`. The default value is 6.

Parameter `usPtcpTakeoverTimeoutFactor`

This parameter defines the synchronization takeover timeout. If the device does not receive a synchronization frame within the specified time, the synchronization slave tries to switch to another synchronization master of the same domain and a secondary synchronization master tries to become the primary synchronization master. The time base if this parameter is defined by `ulSyncSendFactor`. The default value is 3 for a synchronization master and 2 for a synchronization slave.

© Hilscher, 2015–2017

Parameter `usPtcpMasterStartupTime`

This parameter defines a timeout for a synchronization master to become the primary synchronization master. If the synchronization master does not receive a PTCP Announce frame within the specified time it will become the primary synchronization master. The time base for the parameter is one second. The default value is 60.

Parameter `bPtcpMasterPrio1`

This parameter is a bitmask used for the Best Master Clock Algorithm. The following values are defined:

Name	Numerical Value	Meaning
Bit 0 to 2 - Priority	0x00	To be used for a synchronization slave
	0x01	The device is the primary synchronization master
	0x02	The device is a secondary synchronization master
Bit 3 to 5 - Level	0x00	Level 0 (highest priority)
	0x08	Level 1
	0x10	Level 2
	0x18	Level 3
	0x20	Level 4
	0x28	Level 5
	0x30	Level 6
	0x38	Level 7 (lowest priority)

The default value is 0x00 for a synchronization slave and 0x01 for the synchronization master.

Parameter `bPtcpMasterPrio2`

The parameter is used for the Best Master Clock algorithm. Only the value 0xFF shall be used.

Parameter `abSubdomainName[]`

This parameter defines a human readable identifier of the synchronization domain.

© Hilscher, 2015–2017

3.1.12.8 PNM_AP_CFG_PRM_PDINTFMRPDATACHECK_T structure

Structure reference

```
#define PNM_AP_CFG_PRM_MRP_INSTANCE_DATA_VERSION_1            (0x0001)

typedef enum
{
  PNM_AP_CFG_PRM_MRP_CHECK_OFF           = 0,
  PNM_AP_CFG_PRM_MRP_CHECK_MANGER_ON     = 1,
  PNM_AP_CFG_PRM_MRP_CHECK_UUID_ON       = 2,
  PNM_AP_CFG_PRM_MRP_CHECK_MAX           = 4,
} PNM_AP_CFG_PRM_MRP_CHECK_FLAGS_E;

typedef struct PNM_AP_CFG_PRM_MRP_INSTANCE_DATACHECK_Ttag
PNM_AP_CFG_PRM_MRP_INSTANCE_DATACHECK_T;
struct PNM_AP_CFG_PRM_MRP_INSTANCE_DATACHECK_Ttag
{
  uint32_t   ulStructVersion;
  uint8_t    bMRP_Instance;
  uint8_t    bReserved1;
  uint16_t   usReserved1;
  uint32_t   ulMRP_Check;
  PNM_UUID_T tMRP_DomainUUID;
};

#define PNM_AP_CFG_PRM_PDINFMRPDATACHECK_VERSION_1           (0x0001)
#define PNM_AP_CFG_PRM_PDINFMRPDATACHECK_VERSION_2           (0x0002)

typedef struct PNM_AP_CFG_PRM_PDINTFMRPDATACHECK_Ttag
PNM_AP_CFG_PRM_PDINTFMRPDATACHECK_T;
#define PNM_AP_CFG_PRM_MAX_MRP_INSTANCES                      (32)

struct PNM_AP_CFG_PRM_PDINTFMRPDATACHECK_Ttag
{
  uint32_t ulStructVersion;
  PNM_AP_CFG_PRM_MRP_INSTANCE_DATACHECK_T  atCheck[ARRAYS_OF_LENGTH_ZERO];
};
```

Structure description

Structure PNM_AP_CFG_PRM_PDINTFMRPDATACHECK_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_PDINTFMRPDATACHECK_T			
ulStructVersion	UINT32	1,2	Structure version of this structure / Encoding to be used when transferring the record
atCheck[]			Array of Structures describing one MRP Instance

Structure PNM_AP_CFG_PRM_MRP_INSTANCE_DATACHECK_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_MRP_INSTANCE_DATACHECK_T			
ulStructVersion	UINT32	1	Structure version of this structure
bMRP_Instance	UINT8	0 … 126	Reference of this MRP Instance
bReserved1	UINT8	0	Padding. Set to zero for future extensions
usReserved1	UINT16	0	Padding. Set to zero for future extensions
tMRP_DomainUUID zulMRP_Check	UINT32	0 to 3	Bitmask specifying what to check
tMRP_DomainUUID	UUID	any	The MRP domain uuid to validate against

© Hilscher, 2015–2017

Parameter descriptions

Parameter ulStructVersion

Version of this structure. This field is used for future extensions. In case of PNM_AP_CFG_PRM_PDINTFMRPDATACHECK_T the structure version specifies the encoding to use when writing the associated record data object:

Name	Numerical Value	Usage
PNM_AP_CFG_PRM_PDINFMRPDATACHECK_VERSION_1	1	Use encoding according PROFINET Specification V2.2. Only one MRP Instance per device is possible (field bMRP_Instance is ignored)
PNM_AP_CFG_PRM_PDINFMRPDATACHECK_VERSION_2	2	Use encoding according PROFINET Specification V2.31. Multiple MRP Instances can be defined (depends on capabilities of device).

Parameter bMRP_Instance

A reference to the MRP instance. Only relevant for Multiple Interface Devices. Default value is Zero.

Parameter ulMRP_Check

A bitmask that defines which parameters should be checked. The following values are defined

Name	Numerical Value	Usage
PNM_AP_CFG_PRM_MRP_CHECK_MANGER_ON	0x00000001	Enable MRP Manager Diagnosis
PNM_AP_CFG_PRM_MRP_CHECK_UUID_ON	0x00000002	Enable Validation of Peer MRP Domain UUID

Parameter tMRP_DomainUUID

This parameter defines the domain uuid to check against if enabled by parameter ulMRP_Check.

© Hilscher, 2015–2017

3.1.12.9 PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST_T structure

Structure reference

```
#define PNM_AP_CFG_MRP_TOPO_CHANGE_IN     TERVAL_MAX         (100)

#define PNM_AP_CFG_MRP_TOPO_CHANGE_REPEAT_COUNT_MIN     (1)
#define PNM_AP_CFG_MRP_TOPO_CHANGE_REPEAT_COUNT_MAX     (5)

#define PNM_AP_CFG_MRP_SHORT_TEST_INTERVAL_MIN          (1)
#define PNM_AP_CFG_MRP_SHORT_TEST_INTERVAL_MAX          (500)

#define PNM_AP_CFG_MRP_DEFAULT_TEST_INTERVAL_MIN        (1)
#define PNM_AP_CFG_MRP_DEFAULT_TEST_INTERVAL_MAX        (1000)

#define PNM_AP_CFG_MRP_TEST_MONITOR_COUNT_MIN           (2)
#define PNM_AP_CFG_MRP_TEST_MONIROT_COUNT_MAX           (10)

typedef struct PNM_AP_CFG_PRM_MRP_MANAGER_PARAMS_Ttag PNM_AP_CFG_MRP_MANAGER_PARAMS_T;
struct PNM_AP_CFG_PRM_MRP_MANAGER_PARAMS_Ttag
{
  uint16_t usMRP_Prio;
  uint16_t usMRP_TOPchgT;
  uint16_t usMRP_TOPNRmax;
  uint16_t usMRP_TSTshortT;
  uint16_t usMRP_TSTdefaultT;
  uint16_t usMRP_TSTNRmax;
};

#define PNM_AP_CFG_MRP_LINK_DOWN_INTERVAL_MIN          (1)
#define PNM_AP_CFG_MRP_LINK_DOWN_INTERVAL_MAX          (1000)

#define PNM_AP_CFG_MRP_LINK_UP_INTERVAL_MIN            (1)
#define PNM_AP_CFG_MRP_LINK_UP_INTERVAL_MAX            (1000)

#define PNM_AP_CFG_MRP_LINK_CHANGE_COUNT_MIN           (1)
#define PNM_AP_CFG_MRP_LINK_CHANGE_COUNT_MAX           (5)

typedef struct PNM_AP_CFG_PRM_MRP_CLIENT_PARAMS_Ttag PNM_AP_CFG_MRP_CLIENT_PARAMS_T;
struct PNM_AP_CFG_PRM_MRP_CLIENT_PARAMS_Ttag
{
  uint16_t usMRP_LNKdownT;
  uint16_t usMRP_LNKupT;
  uint16_t usMRP_LNKNRmax;
};

typedef union
{
  PNM_AP_CFG_MRP_MANAGER_PARAMS_T  tManager;
  PNM_AP_CFG_MRP_CLIENT_PARAMS_T   tClient;
} PNM_AP_CFG_MRP_PARAMS_T;

enum PNM_AP_CFG_MRP_ROLE_Etag
{
  PNM_AP_CFG_MRP_ROLE_DISABLE  = 0,
  PNM_AP_CFG_MRP_ROLE_CLIENT   = 1,
  PNM_AP_CFG_MRP_ROLE_MANAGER  = 2,
};

#define PNM_AP_CFG_PRAM_MRP_VERSION_1      (0x0001)

typedef struct PNM_AP_CFG_PRM_MRP_Ttag PNM_AP_CFG_PRM_MRP_T;
struct PNM_AP_CFG_PRM_MRP_Ttag
{
  uint32_t               ulStructVersion;
  uint8_t                bMRP_Instance;
  uint8_t                bReserved;
  uint16_t               usMrpRole;
  PNM_UUID_T             tMrpDomainUuid;
  uint8_t                abMrpDomainName[240];
```


© Hilscher, 2015–2017

```
  PNM_AP_CFG_MRP_PARAMS_T uParam;
};

#define PNM_AP_CFG_PRM_MAX_MRP_INSTANCES                          (32)

#define PNM_AP_CFG_PRM_PDINFMRPDATAADJUST_VERSION_1           (0x0001)
#define PNM_AP_CFG_PRM_PDINFMRPDATAADJUST_VERSION_2           (0x0002)

typedef struct PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST_Ttag
PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST_T;
struct PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST_Ttag
{
  uint32_t ulStructVersion;
  PNM_AP_CFG_PRM_MRP_T  atAdjust[ARRAYS_OF_LENGTH_ZERO];
};
```

Structure description

Structure PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST_T			Type: Structure
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST_T			
ulStructVersion	UINT32	1, 2	Structure version of this structure / Encoding to be used when transferring the record
atAdjust[]			Array of Structures describing one MRP Instance

Structure PNM_AP_CFG_PRM_MRP_T			Type: Structure
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_MRP_T			
ulStructVersion	UINT32	1	Structure version of this structure
bMRP_Instance	UINT8	0 … 126	Reference of this MRP Instance
bReserved1	UINT8	0	Padding. Set to zero for future extensions
usMrpRole	UINT16	0 … 2	Role of the Device. Defines which field of uParam is valid.
abMrpDomainName[240]	UINT8[]		MRP Domain Name, minimum one character
uParam			MRP Slave or Master Parameters

Structure PNM_AP_CFG_MRP_CLIENT_PARAMS_T			Type: Structure
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_MRP_CLIENT_PARAMS_T			
usMRP_Prio	UINT16		Priority of MRP Master
usMRP_TOPchgT	UINT16	1 … 100	Interval between two subsequent topology change frames
usMRP_TOPNRmax	UINT16	1 … 5	How often the topology change frame should be repeated
usMRP_TSTshortT	UINT16	1… 500	Interval for ring test frames after a link change
usMRP_TSTdefaultT	UINT16	1 … 1000	Interval for ring test frames in operation
usMRP_TSTNRmax	UINT16	2 … 10	Number of ring test frames until a change of the ring is recognized

© Hilscher, 2015–2017

Structure PNM_AP_CFG_MRP_CLIENT_PARAMS_T			Type: Structure
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_MRP_CLIENT_PARAMS_T			
usMRP_LNKdownT	UINT16	1 … 1000	Interval between two subsequent link down frames
usMRP_LNKupT	UINT16	1 … 1000	Interval between two subsequent link up frames
usMRP_usMRP_LNKNRmax	UINT16	1 … 5	Repeat count for link down/up frames

Parameter descriptions

Parameter ulStructVersion

Version of this structure. This field is used for future extensions. In case of PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST_T the structure version specifies the encoding to use when writing the associated record data object:

Name	Numerical Value	Usage
PNM_AP_CFG_PRM_PDINFMRPDATAADJUST_VERSION_1	1	Use encoding according PROFINET Specification V2.2. Only one MRP Instance per device is possible (field bMRP_Instance is ignored)
PNM_AP_CFG_PRM_PDINFMRPDATAADJUST_VERSION_2	2	Use encoding according PROFINET Specification V2.31. Multiple MRP Instances can be defined. (Depends on capabilities of device)

Parameter bMRP_Instance

A reference to the MRP instance. Only relevant for Multiple Interface Devices. Default value is zero.

Parameter usMrpRole

The role of the device within the MRP ring. The following values are defined

Name	Numerical Value	Usage
PNM_AP_CFG_MRP_ROLE_DISABLE	0x00000000	MRP Disabled for this device
PNM_AP_CFG_MRP_ROLE_CLIENT	0x00000001	Device shall operate as MRP Client. uParam.tClient contains the related parameters
PNM_AP_CFG_MRP_ROLE_MANAGER	0x00000002	Device shall operate as MRP Manager. uParam.tManager contains the related parameters

Parameter abMrpDomainName[240]

The domain name of the MRP domain. Must be a name with minimum length 1 according PROFINET Name Of Station syntax. Even if MRP is disabled a valid name must be specified.

© Hilscher, 2015–2017

3.1.12.10 PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST_T structure

Structure reference

```
#define PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST_VERSION_1          (0x0001)

typedef struct PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST_Ttag
PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST_T;
struct PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST_Ttag
{
  uint32_t   ulStructVersion;
  PNM_UUID_T tMRP_DomainUUID;
};
```

Structure description

Structure PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST_T			
ulStructVersion	UINT32	1	Structure version of this structure
tMRP_DomainUUID	UUID		A UUID associated with the MRP Domain

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used for future extensions.

Parameter tMRP_DomainUUID

This parameter defines the UUID associated with a particular MRP domain. All Ports in one MRP Domain must be configured with the same UUID.

© Hilscher, 2015–2017

3.1.12.11 PNM_AP_CFG_PRM_PDPORTFODATACHECK_T structure

This record structure is used to configure the fiber optic check parameters of a fiber optic port. If the corresponding limits are exceeded the device will generate a diagnosis.

Structure reference

```
#define PNM_AP_CFG_PRM_PDPORTFIBEROPTICDATACHECK_VERSION_1   (0x0001)

typedef struct PNM_AP_CFG_PRM_PDPORTFODATACHECK_Ttag PNM_AP_CFG_PRM_PDPORTFODATACHECK_T;
struct PNM_AP_CFG_PRM_PDPORTFODATACHECK_Ttag
{
  uint32_t ulStructVersion;
  uint32_t ulMaintenanceRequiredPowerBudget;
  uint32_t ulMaintenanceDemandedPowerBudget;
  uint32_t ulErrorPowerBudget;
};
```

Structure description

Structure PNM_AP_CFG_PRM_PDPORTFODATACHECK_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_PDPORTFODATACHECK_T			
ulStructVersion	UINT32	1	Structure version of this structure
ulMaintenanceRequiredPowerBudget	UINT32	0, (1 to 999) | 0x80000000	Maintenance Required Power Budget
ulMaintenanceDemandedPowerBudget	UINT32	0, (1 to 999) | 0x80000000	Maintenance Demanded Power Budget
ulErrorPowerBudget	UINT32	0, (1 to 999) | 0x80000000	Diagnosis Power Budget

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used for future extensions.

Parameters ulMaintenanceRequiredPowerBudget, ulMaintenanceDemandedPowerBudget and ulErrorPowerBudget

These parameters define the minimum power budget of the transmitter (the power reserve of the transmitter) for a maintenance required, demanded or error diagnosis entry shall be generated. If the power budget falls below the corresponding value a diagnosis entry will be generated by the device. The bit 0 to 30 of the value express the power budget in 0.1 db steps from 0 to 99,9. Bit 31 is a flag which enables the particular check.

© Hilscher, 2015–2017

3.1.12.12 PNM_AP_CFG_PRM_PDPORTFODATAADJUST_T structure

This record structure is used to configure the fiber optic parameters of a fiber optic port.

Structure reference

```
typedef enum
{
  PNM_AP_CFG_FIBEROPTIC_CABLE_UNKNOWN                        = 0,
  PNM_AP_CFG_FIBEROPTIC_CABLE_FIXED_INSTALLATION             = 1,
  PNM_AP_CFG_FIBEROPTIC_CABLE_FLIXIBLE_INSTALLATION          = 2,
  PNM_AP_CFG_FIBEROPTIC_CABLE_OUTDOOR_FIXED_INSTALLATION     = 3,
} PNM_AP_CFG_FIBEROPTIC_CABLE_TYPE_E;

typedef struct PNM_AP_CFG_PRM_PDPORTFODATAADJUST_Ttag
PNM_AP_CFG_PRM_PDPORTFODATAADJUST_T;
struct PNM_AP_CFG_PRM_PDPORTFODATAADJUST_Ttag
{
  uint32_t ulStructVersion;
  uint32_t ulFiberOpticType;
  uint32_t ulFiberOpticCableType;
};
```

Structure description

Structure PNM_AP_CFG_PRM_PDPORTFODATAADJUST_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_PDPORTFODATAADJUST_T			
ulStructVersion	UINT32	1	Structure version of this structure
ulFiberOpticType	UINT32	0 … 3	Type of the fiber optic module
ulFiberOpticCableType	UINT32	0 … 255	Type of the fiber optic cable

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used for future extensions.

Parameters ulFiberOpticCableType and ulFiberOpticType

These parameters define the fiber optic connection on this port. The values are typically described within the GSDML of the Device. The following settings are valid.

Name	Numerical value	Meaning
PNM_AP_CFG_FIBEROPTIC_CABLE_UNKNOWN	0	No cable specified
PNM_AP_CFG_FIBEROPTIC_CABLE_FIXED_ INSTALLATION	1	Inside/Outside cable, fixed installation
PNM_AP_CFG_FIBEROPTIC_CABLE_FLIXIBLE_ INSTALLATION	2	Inside/Outside cable, flexible installation
PNM_AP_CFG_FIBEROPTIC_CABLE_OUTDOOR_ FIXED_INSTALLATION	3	Outdoor cable, fixed installation

Table 11: Known values for fiber optic cable type

© Hilscher, 2015–2017

Numerical value	Meaning
0	Not Adjusted
1	9 µm single mode
2	50 µm multi mode
3	62,5 µm multi mode
4	SI-POF, NA = 0.5
5	SI-PCF, NA = 0.36
6	LowNA-POF, NA = 0.3
7	GI-POF
8	GI-PCF
0x80 to 0xFF	Vendor specific

Table 12: Known values for fiber optic type

© Hilscher, 2015–2017

3.1.12.13 PNM_AP_CFG_PRM_PDNCDATACHECK_T structure

Structure reference

```
typedef enum
{
  PNM_AP_CFG_PRM_NC_DROP_BUDGET_CHECK_VALID_FLAG_MAINTENANCE_REQUIRED = 0x00000001,
  PNM_AP_CFG_PRM_NC_DROP_BUDGET_CHECK_VALID_FLAG_MAINTENANCE_DEMANDED = 0x00000002,
  PNM_AP_CFG_PRM_NC_DROP_BUDGET_CHECK_VALID_FLAG_ERROR                = 0x00000004,
} PNM_AP_CFG_PRM_NC_DROP_BUDGET_VALID_FLAGS;

#define PNM_AP_CFG_NC_DROPPED_FRAMES_MAX         (0x03E7)

#define PNM_AP_CFG_PRM_PDNCDATACHECK_VERSION_1   (0x0001)

typedef struct PNM_AP_CFG_PRM_PDNCDATACHECK_Ttag PNM_AP_CFG_PRM_PDNCDATACHECK_T;
struct PNM_AP_CFG_PRM_PDNCDATACHECK_Ttag
{
  uint32_t ulStructVersion;
  uint32_t ulEnableFlag;
  uint32_t ulMaintenanceRequiredDropBudget;
  uint32_t ulMaintenanceDemandedDropBudget;
  uint32_t ulErrorDropBudget;
};
```

Structure description

Structure PNM_AP_CFG_PRM_PDNCDATACHECK_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_PDNCDATACHECK_T			
ulStructVersion	UINT32	1	Structure version of this structure
ulEnableFlag	UINT32	0 … 7	Bitmask specifying which checks to enable
ulMaintenanceRequiredDropBudget	UINT32	1 … 999	Maximum number of frames dropped within 1 second until a Maintenance Required shall be generated
ulMaintenanceDemandedDropBudget	UINT32	1 … 999	Maximum number of frames dropped within 1 second until a Maintenance Demanded shall be generated
ulErrorDropBudget	UINT32	1 … 999	Maximum number of frames dropped within 1 second until a Diagnosis shall be generated

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulStructVersion`

Version of this structure. Used for future extensions.

Parameter `ulEnableFlag`

This bitmask defines which of the following Ethernet frame drop limits are active and will generate a diagnosis.

Name	Numerical value	Meaning
`PNM_AP_CFG_PRM_NC_DROP_BUDGET_CHECK_VALID_FLAG_MAINTENANCE_REQUIRED`	0x00000001	Enable Maintenance Required Limit
`PNM_AP_CFG_PRM_NC_DROP_BUDGET_CHECK_VALID_FLAG_MAINTENANCE_DEMANDED`	0x00000002	Enable Maintenance Demanded Limit
`PNM_AP_CFG_PRM_NC_DROP_BUDGET_CHECK_VALID_FLAG_ERROR`	0x00000004	Enable Diagnosis (Error) Limit

Parameters `ulMaintenanceRequiredDropBudget`, `ulMaintenanceDemandedDropBudget`, `ulErrorDropBudget`

These parameters define the maximum number of frames being dropped within one second until the corresponding diagnosis will be activated.

© Hilscher, 2015–2017

3.1.12.14 PNM_AP_CFG_PRM_PDINTERFACEADJUST_T structure

Structure reference

```
#define PNM_AP_CFG_PRM_PDINTFDATAADJUST_VERSION_1   (0x0001)

typedef struct PNM_AP_CFG_PRM_PDINTERFACEADJUST_Ttag PNM_AP_CFG_PRM_PDINTERFACEADJUST_T;
struct PNM_AP_CFG_PRM_PDINTERFACEADJUST_Ttag
{
  uint32_t ulStructVersion;
  uint32_t ulMultipleInterfaceMode;
};
```

Structure description

Structure PNM_AP_CFG_PRM_PDINTERFACEADJUST_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_PDINTERFACEADJUST_T			
ulStructVersion	UINT32	1	Structure version of this structure
ulMultipleInterfaceMode	UINT32	0, 1The interface mode to use	

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used for future extensions.

Parameter ulMultipleInterfaceMpde

This parameter shall be set to "1" to enable LLDP Multiple Interface Mode according PROFINET Specification V2.31 or to "0" to use standard LLDP Mode as defined in PROFINET specification V2.2.

© Hilscher, 2015–2017

3.1.12.15 PNM_AP_CFG_PRM_PDINTERFACEFSUDATAADJUST_T structure

This record structure is used to configure the DCP Hello Frames sent by the device in Fast Startup Mode. It is associated with the PD Interface Submodule of the corresponding device.

Structure reference

```
enum PNM_AP_FS_HELLO_MODE_Etag
{
  PNM_AP_FS_HELLO_MODE_OFF          = 0,
  PNM_AP_FS_HELLO_MODE_LINKUP       = 1,
  PNM_AP_FS_HELLO_MODE_LINKUP_DELAY = 2,
};

enum PNM_AP_FS_HELLO_INTERVAL_Ttag
{
  PNM_AP_FS_HELLO_INTERVAL_30MS     = 30,
  PNM_AP_FS_HELLO_INTERVAL_50MS     = 50,
  PNM_AP_FS_HELLO_INTERVAL_100MS    = 100,
  PNM_AP_FS_HELLO_INTERVAL_300MS    = 300,
  PNM_AP_FS_HELLO_INTERVAL_500MS    = 500,
  PNM_AP_FS_HELLO_INTERVAL_1000MS   = 1000,
};

enum PNM_AP_FS_HELLO_RETRY_Ttag
{
  PNM_AP_FS_HELLO_RETRY_MIN         = 1,
  PNM_AP_FS_HELLO_RETRY_MAX         = 15,
};

enum PNM_AP_FS_HELLO_DELAY_Ttag
{
  PNM_AP_FS_HELLO_DELAY_OFF         = 0,
  PNM_AP_FS_HELLO_DELAY_50MS        = 50,
  PNM_AP_FS_HELLO_DELAY_100MS       = 100,
  PNM_AP_FS_HELLO_DELAY_500MS       = 500,
  PNM_AP_FS_HELLO_DELAY_1000MS      = 1000,
};

#define PNM_AP_CFG_PRM_PDINTFFSUDATAADJUST_VERSION_1   (0x0001)

typedef struct PNM_AP_CFG_PRM_PDINTERFACEFSUDATAADJUST_Ttag
PNM_AP_CFG_PRM_PDINTERFACEFSUDATAADJUST_T;
struct PNM_AP_CFG_PRM_PDINTERFACEFSUDATAADJUST_Ttag
{
  uint32_t ulStructVersion;
  uint32_t ulFSHelloMode;
  uint32_t ulFSHelloInterval;
  uint32_t ulFSHelloRetry;
  uint32_t ulFSHelloDelay;
};
```

Structure description

Structure PNM_AP_CFG_PRM_PDINTERFACEFSUDATAADJUST_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_PDINTERFACEFSUDATAADJUST_T			
ulStructVersion	UINT32	1	Structure version of this structure
ulFSHelloMode	UINT32	0 … 2	
ulFSHelloInterval	UINT32	30, 50, 100, 300, 500, 1000	The interval between two DCP Hello Frames in ms
ulFSHelloRetry	UINT32	1 … 15	Count of repetitions of the DCP Hello Frames
ulFSHelloDelay	UINT32	0, 50, 100, 500, 1000	The initial delay in ms before the first DCP Hello shall be issued

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulStructVersion`

Version of this structure. Used for future extensions.

Parameter `ulFSHelloMode`

The "Hello" mode the device shall use. The following values are defined:

Name	Numerical value	Meaning
PNM_AP_FS_HELLO_MODE_OFF	0	The device shall not issue hello mode.
PNM_AP_FS_HELLO_MODE_LINKUP	1	The device shall issue DCP Hello Frames after LinkUp
PNM_AP_FS_HELLO_MODE_LINKUP_DELAY	2	The device shall issue DCP Hello Frames with a delay of ulFSHelloDelay milliseconds after Linkup.

Parameter `ulFSHelloInterval`

This parameter defines the time interval in milliseconds between two DCP Hello Frames sent by the device. The following values are valid:

Name	Numerical value	Meaning
PNM_AP_FS_HELLO_INTERVAL_30MS	30	30 ms interval
PNM_AP_FS_HELLO_INTERVAL_50MS	50	50 ms interval
PNM_AP_FS_HELLO_INTERVAL_100MS	100	100 ms interval
PNM_AP_FS_HELLO_INTERVAL_300MS	300	300 ms interval
PNM_AP_FS_HELLO_INTERVAL_500MS	500	500 ms interval
PNM_AP_FS_HELLO_INTERVAL_1000MS	1000	1000 ms interval

Parameter `ulFSHelloRetry`

The device will resend the DCP Hello Frame up to `ulFSHelloRetry` times if no AR is established.

Parameter `ulFSHelloDelay`

This parameter defines the delay between LinkUp and the first DCP Hello Frame if mode PNM_AP_FS_HELLO_MODE_LINKUP_DELAY is used. The following values are defined:

Name	Numerical value	Meaning
PNM_AP_FS_HELLO_DELAY_OFF	0	No Delay. Use for PNM_AP_FS_HELLO_MODE_LINKUP
PNM_AP_FS_HELLO_DELAY_50MS	50	50 ms delay
PNM_AP_FS_HELLO_DELAY_100MS	100	100 ms delay
PNM_AP_FS_HELLO_DELAY_500MS	500	500 ms delay
PNM_AP_FS_HELLO_DELAY_1000MS	1000	1000 ms delay

© Hilscher, 2015–2017

3.1.12.16 PNM_AP_CFG_PRM_AR_FSU_PARAMETERS_T structure

This record structure is used to configure the Parameter Set UUID in Fast Startup Mode. The UUID is used by the application of the device in fast startup mode when applying remanent stored application parameter records to the submodules. If the same UUID is used in subsequent AR connects, the application will apply the stored parameters immediately. This improves the startup speed of an AR.

Note: Depending on the AR Startup Mode, this record object is either written as first record (Legacy Startup) or parameterized already in AR Connect (Advanced Startup). This is done automatically within the PROFINET IO Controller protocol stack.

Structure reference

```
enum PNM_AP_CFG_AR_PRM_FSPARAM_MODE_Etag
{
  PNM_AP_CFG_AR_PRM_FSPARAM_MODE_DISABLE = 1,
  PNM_AP_CFG_AR_PRM_FSPARAM_MODE_ENABLE  = 2,
};
typedef enum PNM_AP_CFG_AR_PRM_FSPARAM_MODE_Etag PNM_AP_CFG_AR_PRM_FSPARAM_MODE_E;

#define PNM_AP_CFG_AR_PRM_FSU_PARAMETERS_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_PRM_AR_FSU_PARAMETERS_Ttag PNM_AP_CFG_PRM_AR_FSU_PARAMETERS_T;
struct PNM_AP_CFG_PRM_AR_FSU_PARAMETERS_Ttag
{
  uint32_t              ulStructVersion;
  uint32_t              ulFSParamMode;
  PNM_UUID_T            tFSParamUuid;
};
```

Structure description

Structure PNM_AP_CFG_PRM_AR_FSU_PARAMETERS_T			Type: Structure
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_AR_FSU_PARAMETERS_T			
ulStructVersion	UINT32	1	Structure version of this structure
ulFSParamMode	UINT32	1, 2	Mode of Fast Startup Parameters
tFSParamUuid	UUID		A UUID associated with the parameters set

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used for future extensions.

Parameter ulFSParamMode

The mode hello mode the device shall use. The following values are defined

Name	Numerical value	Meaning
PNM_AP_CFG_AR_PRM_FSPARAM_MODE_DISABLE	1	The device shall not use fast parameterization
PNM_AP_CFG_AR_PRM_FSPARAM_MODE_ENABLE	2	The device shall use fast parameterization

Parameter tFSParamUuid

A UUID associated the with complete parameter set associated with the device. This UUID is generated by the engineering software.

© Hilscher, 2015–2017

3.1.12.17 PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_T structure

This record structure is used to configure the isochronous mode data record of a submodule. The record data will be written when establishing an IRT AR and contains information about input and output timing of the submodule. This parameter must only be used if the submodule support isochronous operation (GSDML file), the submodule shall operate in isochronous mode (Application dependent) and the submodule is part of an IRT application relation. (Engineering)

Structure reference

```
#define PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_TIMEIOINPUT_MAX       (32000000)
#define PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_TIMEIOOUTPUT_MAX      (32000000)
#define PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_TIMEIOINPUTVALID_MAX  (32000000)
#define PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_TIMEIOOUTPUTVALID_MAX (32000000)

#define PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_CONTROLLERAPPLICATIONCYCLEFACTOR_MIN (0x0001)
#define PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_CONTROLLERAPPLICATIONCYCLEFACTOR_MAX (0x0400)

#define PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_TIMEDATACYCLE_MIN     (0x0001)
#define PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_TIMEDATACYCLE_MAX     (0x0400)

#define PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_VERSION_1             (0x0001)

typedef struct PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_Ttag
PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_Ttag
{
  uint32_t ulStructVersion;
  uint16_t usControllerApplicationCycleFactor;
  uint16_t usTimeDataCycle;
  uint32_t ulTimeIOInput;
  uint32_t ulTimeIOOutput;
  uint32_t ulTimeIOInputValid;
  uint32_t ulTimeIOOutputValid;
};
```

Structure description

Structure PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usControllerApplicationCycleFactor	UINT16	0x0001 to 0x0400	Reduction of the controller application relative to time data cycle
usTimeDataCycle	UINT16	0x0001 to 0x0400	Length of one data cycle in units of 31.25 µs.
ulTimeIOInput	UINT32	0 to 32000000	Sample point of input data relative to start of next network cycle in nanoseconds.
ulTimeIOOutput	UINT32	0 to 32000000	Apply point of output data relative to start of current network cycle in nanoseconds
ulTimeIOInputValid	UINT32	0 to 32000000	Set to zero
ulTimeIOOutputValid	UINT32	0 to 32000000	Time relative to start of current network cycle when the process data is available at the device.

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used for future extensions.

© Hilscher, 2015–2017

Parameter `usControllerApplicationCycleFactor`

This parameter describes the reduction of the controller application relative to time data cycle of the submodule.

Note: For simplicity it is strongly recommended to use the value 0x0001.

Parameter `usTimeDataCycle`

Length of one network cycle. This is typically the same value as the sendclock of the IRT AR associated with this submodule

Parameter `ulTimeIOInput`

The time when the submodule shall sample the input value. The time is relative to the next network cycle start. This time is calculated by the engineering software based on values from the GSDML and controller application requirements.

Parameter `ulTimeIOOutput`

The time when the submodule shall apply the output values. The time is relative to the current network cycle start. This time is calculated by the engineering software bases on values from the GSDML, controller application requirements and the network topology. This value is always greater then ulTimeIOOutputValid.

Parameter `ulTimeIOInputValid`

This parameter is defined in PROFINET specification symmetrically to ulTimeIOOutputValid, but this time is always zero. (There is no propagation delay of the input data to the next bus cycle).

Parameter `ulTimeIOOutputValid`

This time indicates when the new output process data is available at the device for processing. It is relative to the current network cycle start and depends on the network topology.

© Hilscher, 2015–2017

3.1.12.18 PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_T structure

This record structure is used to configure the isochronous controller data record of the profinet controller. This is an artifical record which is not related to any record object of the PROFINET specification. Purpose of this record is to provide timing information from the engineering software to the host application of the PROFINET IRT controller firmware

Structure reference

```
#define PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_VERSION_1             (0x0001)

typedef struct PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_Ttag
PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_Ttag
{
  uint32_t ulStructVersion;
  uint32_t ulTimeControllerApplicationInputMin;
  uint32_t ulTimeControllerApplicationOutputMin;
  uint32_t ulTimeControllerApplicationValid;
  uint32_t ulTimeControllerApplicationStart;
  uint32_t ulTimeControllerApplicationEnd;
};
```

Structure description

Structure PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
ulTimeController ApplicationInputMin	UINT32		Time required by the firmware to update the Input DPM area from receive buffers in nanoseconds
ulTimeController applicationOutputMin	UINT32		Time required by the firmware to update the transmit buffers from Output DPM area in nanoseconds
ulTimeController ApplicationValid	UINT32		Time when the process data has been received at the profinet controller in nanoseconds
ulTimeController ApplicationStart	UINT32		Time when the host application can start handling the process data.
ulTimeController ApplicationEnd	UINT32		Time when the host application is expected to finish handling the process data

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used for future extensions.

Parameter ulTimeControllerApplicationInputMin

This is the time required by the PROFINET IRT Controller firmware to copy process data from receive buffers into the DPM input area. This time depends on the configuration and other is estimated by the engineering software.

Parameter usTimeControllerApplicationOutputMin

This is the time required by the PROFINET IRT Controller firmware to copy process data from the DPM output area to the transmit buffers. This time depends on the configuration and other is estimated by the engineering software.

© Hilscher, 2015–2017

Parameter `ulTimeControllerApplicationValid`

This time offset is relative to the start of the current network cycle and defines at which time the IRT process data exchange on the network is finished. The time depends on the network topology and is calculated by the engineering software.

Parameter `ulTimeControllerApplicationStart`

This time offset is relative to the start of the current network cycle and provides an estimation when the controller application can start handling the process data. The value depends on network topology and includes the time required to copy the input data into the DPM inpuit area.

Note: It is not expected that the host application acts upon this time point. The DPM input area handshake mechanism will ensure that the input process data will be available to the application around this time point.

Parameter `ulTimeControllerApplicationEnd`

This time offset is relative to the start of the current network cycle and provides an estimation when the controller application should be finished handling the process data. The value depends mainly on the copy time from DPM output area into the transmit buffer. Using `ulTimeControllerApplicationStart` the application can calculate how much time is available for data processing.

Note: In order for the output data to be transmitted in the next bus cycle, the application shall handshake DPM output area before this time point.

© Hilscher, 2015–2017

3.2 Acyclic requests

This section contains the description of packets the application can use for requesting acyclic services.

3.2.1 Read submodule record service

This service can be used by the application to read a record object of a previously configured submodule. The submodule must be in data exchange for that purpose.

Note: This service can also be used to read record objects provided by the PROFINET IRT Controller itself. For that purpose the fix defined submodule handles of the controllers PDEV submodules can be used.

Note: When the confirmation packet of this service might exceed the mailbox size a fragmented transfer will be used. This is either the case if a fragmented transfer is explicitly initiated by setting ulExt to RCX_PACKET_SEQ_LAST or when the ulLenToRead is set to a value greater than 1540. Only one such fragmented service might be active at the same time.

The following record objects are currently implemented by the PROFINET controller (when using submodule handles 0xFFF0, 0xFFF1 or 0xFFF2):

Name	Numerical value of record index	Meaning
Diagnosis in channel coding for one subslot	0x800A	Return all channel diagnosis for the specified subslot with severity “diagnosis”
Diagnosis in all codings for one subslot	0x800B	Return all diagnosis for the specified subslot with severity “diagnosis”
Diagnosis, Maintenance, Qualified and Status for one subslot	0x800C	Return all diagnosis for the specified subslot
Maintenance required in channel coding for one subslot	0x8010	Return all channel diagnosis for the specified subslot with severity “maintenance required”
Maintenance demanded in channel coding for one subslot	0x8011	Return all channel diagnosis for the specified subslot with severity “maintenance demanded”
Maintenance required in all codings for one subslot	0x8012	Return all diagnosis for the specified subslot with severity “maintenance required”
Maintenance demanded in all codings for one subslot	0x8013	Return all diagnosis for the specified subslot with severity “maintenance demanded”
PD Port Data Real	0x802A	Return current Profinet Physical Device Port state. E.g. Information about detected network peers, Mau Type, Cable Delay. (Available for Handle 0xFFF1 and 0xFF2 only)
PD Port Statistics	0x8072	Return ethernet switch port or interface statistics, E.g. Bytes received/Sent.
PD Interface Data Real	0x8080	Return current Profinet Physical Device Interface state. E.g. Name Of Station, IP Address, Interface Mac Address. (Available for Handle 0xFFF0 only)
Diagnosis in channel coding for one slot	0xC00A	Return all channel diagnosis for the specified slot with severity “diagnosis”
Diagnosis in all codings for one slot	0xC00B	Return all diagnosis for the specified slot with severity “diagnosis”

© Hilscher, 2015–2017

Name	Numerical value of record index	Meaning
Diagnosis, Maintenance, Qualified and Status for one slot	0xC00C	Return all diagnosis for the specified slot
Maintenance required in channel coding for one slot	0xC010	Return all channel diagnosis for the specified slot with severity “maintenance required”
Maintenance demanded in channel coding for one slot	0xC011	Return all channel diagnosis for the specified slot with severity “maintenance demanded”
Maintenance required in all codings for one slot	0xC012	Return all diagnosis for the specified slot with severity “maintenance required”
Maintenance demanded in all codings for one slot	0xC013	Return all diagnosis for the specified slot with severity “maintenance demanded”
Diagnosis in channel coding for one AR	0xE00A	Return all channel diagnosis for the specified AR with severity “diagnosis”
PD Real Data	0xF841	Return combined information from PD Interface Data Real, PD Port Data Real and PD Port Statistics. (MRP Information not implemented)

3.2.1.1 PNM_AP_READ_RECORD_SUBM_REQ_T request

Packet structure reference

```
typedef struct PNM_AP_READ_RECORD_SUBM_CNF_DATA_Ttag PNM_AP_READ_RECORD_SUBM_CNF_DATA_T;
struct PNM_AP_READ_RECORD_SUBM_CNF_DATA_Ttag
{
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle;
  uint16_t                  usIndex;
  uint32_t                  ulDataLen;
  uint32_t                  ulPnio;
  uint16_t                  usAddVal1;
  uint16_t                  usAddVal2;
  uint8_t                   abRecordData[ARRAYS_OF_LENGTH_ZERO];
};
typedef struct PNM_AP_READ_RECORD_SUBM_CNF_Ttag PNM_AP_READ_RECORD_SUBM_CNF_T;
struct PNM_AP_READ_RECORD_SUBM_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T                tHead;
  PNM_AP_READ_RECORD_SUBM_CNF_DATA_T tData;
};
```

Packet description

Structure PNM_AP_CFG_IOC_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Unused by Stack. Set to zero for future compatibility
ulSrcId	UINT32	any	-
ulLen	UINT32	8	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x9422	PNM_AP_CMD_READ_RECORD_SUBM_REQ
ulExt	UINT32	0	Set to Zero

© Hilscher, 2015–2017

Structure PNM_AP_CFG_IOC_REQ_T				**Type: Request**
ulRoute	UINT32	any	Set to Zero	
Structure PNM_AP_CFG_IOC_DATA_T				
usSubmoduleHandle	UINT16	1 to 2048, 0xFFF, 0xFFF1, 0xFFF2	Handle of submodule to read data from	
usIndex	UINT16	0 to 0xFFFF	Index of record object to read	
ulMaxReadLen	UINT32	1 to 65536	Amount of data to read from object	

© Hilscher, 2015–2017

Parameter descriptions

Parameter usSubmoduleHandle

The handle of the (previously configured) submodule to read the record object from. The following values are defined:

Submodule handle	Meaning
1 to 2048	Read a record object from a configured device's submodule.
0xFFF0	Read a record object from the controllers interface submodule
0xFFF1	Read a record object from the controllers first port submodule
0xFFF2	Read a record object from the controllers second port submodule

Parameter usIndex

The index of the record object to read.

Parameter ulMaxReadLen

The maximum number of bytes to read from the object.

© Hilscher, 2015–2017

3.2.1.2 PNM_AP_READ_RECORD_SUBM_CNF_T confirmation

Packet structure reference

```
typedef struct PNM_AP_READ_RECORD_SUBM_CNF_DATA_Ttag PNM_AP_READ_RECORD_SUBM_CNF_DATA_T;
struct PNM_AP_READ_RECORD_SUBM_CNF_DATA_Ttag
{
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle;
  uint16_t                  usIndex;
  uint32_t                  ulDataLen;
  uint32_t                  ulPnio;
  uint16_t                  usAddVal1;
  uint16_t                  usAddVal2;
  uint8_t                   abRecordData[ARRAYS_OF_LENGTH_ZERO];
};
typedef struct PNM_AP_READ_RECORD_SUBM_CNF_Ttag PNM_AP_READ_RECORD_SUBM_CNF_T;
struct PNM_AP_READ_RECORD_SUBM_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T                tHead;
  PNM_AP_READ_RECORD_SUBM_CNF_DATA_T tData;
};
```

Packet description

Structure PNM_AP_READ_RECORD_SUBM_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	Ignore for future compatibility.
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	16 to 65552	Packet data length in bytes. Depends on amount of record data returned by the device
ulId	UINT32	mirrored	-
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x9423	PNM_AP_CMD_READ_RECORD_SUBM_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure PNM_AP_READ_RECORD_SUBM_CNF_DATA_T			
usSubmoduleHandle	UINT16	1 to 2048	Handle of submodule the data was read from
usIndex	UINT16	0 to 0xFFFF	Index of record object read
ulReadLen	UINT32	0 to 65536	Amount of data read from the object
ulPnio	UINT32	any	The PROFINET Status Code from the read operation
usAddVal1	UINT16	any	Additional value returned by PROFINET Device when reading the record object
usAddVal2	UINT16	any	Additional value returned by the PROFINET Device when reading the record object
abRecordData[]	UINT8		The data returned by the record object

© Hilscher, 2015–2017

Parameter descriptions

Parameter `usSubmoduleHandle`

The handle of the (previously configured) submodule the record data was read from

Parameter `usIndex`

The index of the record object read.

Parameter `ulReadLen`

The number of bytes returned from the record object

Parameter `ulPnio`

The PROFINET status code of the acyclic operation. This field is non-zero if an PROFINET error occurred during processing the read service. The error code is either generated by the PROFINET IO Controller or by the associated PROFINET IO Device. Section *PROFINET Status Code* on page 279 describes the meanings of the PROFINET status code.

Parameters `usAddVal1`, `usAddVal2`

The PROFINET specification defines additional values to be passed from the device in read record object service responses. The content of these values can be found in this parameters.

Parameter `abRecordData`[]

This parameter contains the record data returned by the device.

© Hilscher, 2015–2017

3.2.2 Write submodule record service

This service can be used by the application to write a record object while the submodule is in cyclic data exchange.

Note: When the request packet of this service will exceed the mailbox size a fragmented transfer will be used. This is the case when the ulDataLen is set to a value greater than 1544. Only one such fragmented service might be active at the same time.

3.2.2.1 PNM_AP_WRITE_RECORD_SUBM_REQ_T request

```
typedef struct PNM_AP_WRITE_RECORD_SUBM_REQ_DATA_Ttag
PNM_AP_WRITE_RECORD_SUBM_REQ_DATA_T;
struct PNM_AP_WRITE_RECORD_SUBM_REQ_DATA_Ttag
{
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle;
  uint16_t                  usIndex;
  uint32_t                  ulDataLen;
  uint8_t                   abRecordData[ARRAYS_OF_LENGTH_ZERO];
};

typedef struct PNM_AP_WRITE_RECORD_SUBM_REQ_Ttag PNM_AP_WRITE_RECORD_SUBM_REQ_T;
struct PNM_AP_WRITE_RECORD_SUBM_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T                tHead;
  PNM_AP_WRITE_RECORD_SUBM_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_WRITE_RECORD_SUBM_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Unused by stack Set to zero for future compatibility
ulSrcId	UINT32	any	-
ulLen	UINT32	13 to 65548	Packet Data Length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x9424	PNM_AP_CMD_WRITE_RECORD_SUBM_REQ
ulExt	UINT32	0	Set to zero
ulRoute	UINT32	0	Set to zero
Structure PNM_AP_WRITE_RECORD_SUBM_REQ_DATA_T			
usSubmoduleHandle	UINT16	1 to 2048	Handle of submodule to write the data to
usIndex	UINT16	0 to 0xFFFF	Index of record object to write
ulDataLen	UINT32	1 to 65536	Amount of data to write to the object
abRecordData[]	UINT8		The record data to write

© Hilscher, 2015–2017

Parameter descriptions

Parameter `usSubmoduleHandle`

The handle of the (previously configured) submodule to write the record object for.

Parameter `usIndex`

The index of the record object to write.

Parameter `ulDataLen`

The number of bytes to write to the object

Parameter `abRecordData[]`

The record data to write.

© Hilscher, 2015–2017

3.2.2.2 PNM_AP_WRITE_RECORD_SUBM_CNF_T confirmation

```
typedef struct PNM_AP_WRITE_RECORD_SUBM_CNF_DATA_Ttag
PNM_AP_WRITE_RECORD_SUBM_CNF_DATA_T;
struct PNM_AP_WRITE_RECORD_SUBM_CNF_DATA_Ttag
{
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle;
  uint16_t                  usIndex;
  uint32_t                  ulPnio;
  uint16_t                  usAddVal1;
  uint16_t                  usAddVal2;
};

typedef struct PNM_AP_WRITE_RECORD_SUBM_CNF_Ttag PNM_AP_WRITE_RECORD_SUBM_CNF_T;
struct PNM_AP_WRITE_RECORD_SUBM_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T                tHead;
  PNM_AP_WRITE_RECORD_SUBM_CNF_DATA_T tData;
};
```

Packet description

Structure PNM_AP_WRITE_RECORD_SUBM_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	Ignore for future compatibility.
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	12	Packet data length in bytes
ulId	UINT32	mirrored	-
ulSta	UINT32		=0: no error <> 0: see section Status codes / Error codes on page 276.
ulCmd	UINT32	0x9425	PNM_AP_CMD_WRITE_RECORD_SUBM_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure PNM_AP_WRITE_RECORD_SUBM_CNF_DATA_T			
usSubmoduleHandle	UINT16	1…2048	Handle of submodule the data was written to
usIndex	UINT16	0…0xFFFF	Index of record object written
ulPnio	UINT32	any	The PROFINET Status Code of the write operation
usAddVal1	UINT16	any	Additional value returned by PROFINET Device when writing the record object
usAddVal2	UINT16	any	Additional value returned by the PROFINET Device when writing the record object

© Hilscher, 2015–2017

Parameter descriptions

Parameter usSubmoduleHandle

The handle of the (previously configured) submodule the record data was written to.

Parameter `usIndex`

The index of the record object written.

Parameter `ulPnio`

The PROFINET status code of the acyclic operation. This field is non-zero if a PROFINET error occurred during processing the write service. The error code is either generated by the PROFINET IO Controller or by the associated PROFINET IO Device. Section *PROFINET Status Code* on page 279 describes the meanings of the PROFINET status code.

Parameters usAddVal1, usAddVal2

The PROFINET specification defines additional values to be passed from the device in write record object service responses. The content of these values can be found in this parameters.

© Hilscher, 2015–2017

3.2.3 Read implicit record service

This service can be used by the application to implicitly read an record object of an device. This service can be used to read PROFINET record objects from any PROFINET IO Device with an assigned IP Address in the network.

Note: The service involves RPC Transactions which might timeout due to network problems or unreachable devices. Due to these timeouts the confirmation packet might be delayed by several seconds. This must be considered for the application implementation.

Note: Only one Read implicit record service may be active at the same time. The application must wait for the confirmation before the next read can be performed.

Note: When the confirmation packet of this service might exceed the mailbox size a fragmented transfer will be used. This is either the case if a fragmented transfer is explicitly initiated by setting ulExt to RCX_PACKET_SEQ_LAST or when the ulMaxLenToRead field is set to a value greater than 1530.

3.2.3.1 PNM_AP_READ_IMPLICIT_RECORD_REQ_T request

Packet structure reference

```
typedef struct PNM_AP_READ_IMPLICIT_RECORD_REQ_DATA_Ttag
PNM_AP_READ_IMPLICIT_RECORD_REQ_DATA_T;
struct PNM_AP_READ_IMPLICIT_RECORD_REQ_DATA_Ttag
{
  uint32_t      ulIPAddr;
  uint16_t      usVendorId;
  uint16_t      usDeviceId;
  uint16_t      usInstanceId;
  uint16_t      usReserved1;
  uint32_t      ulApi;
  uint16_t      usSlot;
  uint16_t      usSubslot;
  uint16_t      usIndex;
  uint16_t      usReserved2;
  uint32_t      ulMaxLenToRead;
};

typedef struct PNM_AP_READ_IMPLICIT_RECORD_REQ_Ttag PNM_AP_READ_IMPLICIT_RECORD_REQ_T;
struct PNM_AP_READ_IMPLICIT_RECORD_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T                    tHead;
  PNM_AP_READ_IMPLICIT_RECORD_REQ_DATA_T tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_READ_IMPLICIT_RECORD_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Unused by Stack Set to zero for future compatibility
ulSrcId	UINT32	any	-
ulLen	UINT32	26	Packet Data Length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x9426	PNM_AP_CMD_READ_IMPLICIT_RECORD_REQ
ulExt	UINT32	0	Set to zero
ulRoute	UINT32	0	Set to zero
Structure PNM_AP_READ_IMPLICIT_RECORD_REQ_DATA_T			
ulIpAddr	UINT32	any	IP address of device to read the record from
usVendorID	UINT16	Any	PNO Vendor ID of the device to read the record from
usDeviceID	UINT16	any	Device ID of the device to read the record from
usInstanceID	UINT16	any	Instance ID of the device to read the record from
usReserved1	UINT16	0	Reserved for future use. Set to zero.
ulApi	UINT32	Any	API to use for reading the object
usSlot	UINT16	Any	Slot to use for reading the object
usSubslot	UINT16	Any	Subslot to use for reading the object
usIndex	UINT16	any	Index of the record object to read
usReserved2	UINT16	0	Reserved for future use. Set to zero.
ulMaxLenToRead	UINT32	1 to 65536	Maximum length to read

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulIPAddr`

The IP Address of the device to read the record object from.

Parameters `usVendorId`, `usDeviceId`, `usInstanceId`

PROFINET Device identification according GSDML file of the device to read the record object from.

Parameters `ulApi`, `usSlot`, `usSubslot`

The addressing parameters of the submodule to read the record object from.

Parameter `usIndex`

The index of the record object to read.

Parameter `ulMaxLenToRead`

The maximum number of bytes to read from the object.

© Hilscher, 2015–2017

3.2.3.2 PNM_AP_READ_IMPLICIT_RECORD_CNF_T confirmation

Packet structure reference

```
typedef struct PNM_AP_READ_IMPLICIT_RECORD_CNF_DATA_Ttag
PNM_AP_READ_IMPLICIT_RECORD_CNF_DATA_T;
struct PNM_AP_READ_IMPLICIT_RECORD_CNF_DATA_Ttag
{
  uint32_t      ulIPAddr;
  uint16_t      usVendorId;
  uint16_t      usDeviceId;
  uint16_t      usInstanceId;
  uint16_t      usReserved1;
  uint32_t      ulApi;
  uint16_t      usSlot;
  uint16_t      usSubslot;
  uint16_t      usIndex;
  uint16_t      usReserved2;
  uint32_t      ulLenRead;
  uint32_t      ulPnio;
  uint16_t      usAddVal1;
  uint16_t      usAddVal2;
  uint8_t       abRecordData[ARRAYS_OF_LENGTH_ZERO];
};

typedef struct PNM_AP_READ_IMPLICIT_RECORD_CNF_Ttag PNM_AP_READ_IMPLICIT_RECORD_CNF_T;
struct PNM_AP_READ_IMPLICIT_RECORD_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T                   tHead;
  PNM_AP_READ_IMPLICIT_RECORD_CNF_DATA_T tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_READ_IMPLICIT_RECORD_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		
ulSrc	UINT32		
ulDestId	UINT32	0	Ignore for future compatibility
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	36 - 65572	Packet Data Length in bytes. Depends on amount of record data returned by the device
ulId	UINT32	mirrored	-
ulSta	UINT32	0	=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x9427	PNM_AP_CMD_READ_IMPLICIT_RECORD_CNF
ulExt	UINT32	0	Ignore
ulRoute	UINT32	0	Ignore
Structure PNM_AP_READ_IMPLICIT_RECORD_CNF_DATA_T			
ulIPAddr	UINT32	mirrored	IP Address of the device the record was read from
usVendorId	UINT16	mirrored	PROFINET Vendor Id of the device the record was read from
usDeviceId	UINT16	mirrored	PROFINET Device Id of the device the record was read from
usInstanceId	UINT16	mirrored	Instance Id of the device the record was read from
usReserved1	UINT16	0	Ignore for future compatibility
ulApi	UINT32	mirrored	API of the submodule the record was read from
usSlot	UINT16	mirrored	Slot of the submodule the record was read from
usSubslot	UINT16	mirrored	Subslot of the submodule the record was read from
usIndex	UINT16	mirrored	Index of record object read
usReserved2	UINT16	0	Ignore for future compatibility
ulLenRead	UINT32	0 to 65536	Amount of bytes read from the record object
ulPnio	UINT32	any	PROFINET Status code of the read implicit operation
usAddVal1	UINT16	any	Additional value returned by the PROFINET Device when reading the record object
usAddVal2	UINT16	any	Additional value returned by the PROFINET Device when reading the record object
abRecordData[]	UINT8		The data read from the record object
usAddVal2	UINT16	any	Additional value returned by the PROFINET Device when reading the record object
abRecordData[]	UINT8		The data read from the record object

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulIPAddr`

The IP Address of the device the record object was read from.

Parameters usVendorId, usDeviceId, usInstanceId

PROFINET Device identification according GSDML file of the device the record object was read from.

Parameters `ulApi`, `usSlot`, usSubslot

Addressing parameters of the submodule the record was read from

Parameter `usIndex`

The index of the record object to read.

Parameter `ulLenRead`

The number of bytes returned from the record object

Parameter `ulPnio`

The PROFINET status code of the acyclic operation. This field is non-zero if an PROFINET error occurred during processing the read implicit service. The error code is either generated by the PROFINET IO Controller or by the associated PROFINET IO Device. Section *PROFINET Status Code* on page 279 describes the meanings of the PROFINET status code.

Parameters usAddVal1, usAddVal2

The PROFINET specification defines additional values to be passed from the device in read record object service responses. The content of these values can be found in this parameters.

Parameter abRecordData[]

This parameter contains the record data returned by the device.

© Hilscher, 2015–2017

3.2.4 Acknowledge Alarm service

This service shall be used by the host application to acknowledge a previously received *Receive Alarm service* (page 176) or *Receive Diagnosis service* (page 181). This service initiates the application alarm acknowledge as defined by the PROFINET specification.

Note: According to PROFINET Specification it is expected that this service is used after application processing of the Diagnosis/Alarm. The intention behind this service is to confirm to the device that the Diagnosis/Alarm had been processed by the application.

3.2.4.1 PNM_AP_ACK_ALARM_REQ_T request

Packet structure reference

```
typedef struct PNM_AP_ACK_ALARM_REQ_DATA_Ttag PNM_AP_ACK_ALARM_REQ_DATA_T;
struct PNM_AP_ACK_ALARM_REQ_DATA_Ttag
{
  PNM_AP_DEVICEHANDLE_T     usDeviceHandle;
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle;
  uint16_t                  usAlarmType;
  uint16_t                  usAlarmPriority;
  uint32_t                  ulPnio;
};

typedef struct PNM_AP_ACK_ALARM_REQ_Ttag PNM_AP_ACK_ALARM_REQ_T;
struct PNM_AP_ACK_ALARM_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T         tHead;
  PNM_AP_ACK_ALARM_REQ_DATA_T tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_ACK_ALARM_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility
ulSrcId	UINT32	any	-
ulLen	UINT32	12	Packet Data Length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x942C	PNM_AP_CMD_ACK_ALARM_REQ
ulExt	UINT32	0	
ulRoute	UINT32	0	
Structure PNM_AP_ACK_ALARM_REQ_DATA_T			
usDeviceHandle	UINT16	0, 1 to 128	Handle of the device causing the alarm
usSubmoduleHandle	UINT16	1 to 2048, 0xFFF0, 0xFFF1, 0xFFF2	Handle of the submodule causing the alarm
usAlarmType	UINT16		PROFINET Type of the Alarm
usAlarmPriority	UINT16	0, 1	Priority of the Alarm
ulPnio	UINT32		Status Code of Alarm Processing

Parameter descriptions

Parameters usDeviceHandle, usSubmoduleHandle

These parameter specify the source of the PROFINET alarm. They correspond to the handles used when configuring the PROFINET IO Controller and must be set to the same value as in the indication packet.

Parameter usAlarmType

This parameter defines the type of the alarm received. It must be set to the same value as in the indication packet.

Parameter usAlarmPriority

This parameter defines the alarm priority. It must be set to the same value as in the indication packet.

Parameter ulPnio

This parameter shall be set to the application status code of the alarm/diagnosis processing. If this value is set to a non-zero value the associated application relation might be aborted depending on the device application implementation.

© Hilscher, 2015–2017

3.2.4.2 PNM_AP_ACK_ALARM_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T PNM_AP_ACK_ALARM_CNF_T;
```

Packet description

Structure PNM_AP_ACK_ALARM_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	mirrored	-
ulSta	UINT32		=0: no error <> 0: see section Status codes / Error codes on page 276.
ulCmd	UINT32	0x942D	PNM_AP_CMD_ACK_ALARM_CNF
ulExt	UINT32	0	
ulRoute	UINT32		Ignore

© Hilscher, 2015–2017

3.2.5 DCP Set Name service

The DCP Set Name service can be used to set the name of an IO Device.

Note: Only one DCP Service can be active at a time. It is not possible to use different DCP Services at the same time to the same or different devices. The application must wait for the confirmation of this service before requesting the next DCP Service.

3.2.5.1 PNM_AP_DCP_SET_NAME_REQ_T request

Packet structure reference

```
#define PNIO_APCTL_DCP_QUALIFIER_STORE_TEMPORARY                0x0
#define PNIO_APCTL_DCP_QUALIFIER_STORE_PERMANENT                0x1

typedef struct PNM_AP_DCP_SET_NAME_REQ_DATA_Ttag PNM_AP_DCP_SET_NAME_REQ_DATA_T;
 struct PNM_AP_DCP_SET_NAME_REQ_DATA_Ttag
{
  uint8_t        abMacAddr[6];
  uint16_t       usQualifier;
  uint8_t        abName[240];
};

typedef struct PNM_AP_DCP_SET_NAME_REQ_Ttag PNM_AP_DCP_SET_NAME_REQ_T;
 struct PNM_AP_DCP_SET_NAME_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T            tHead;
  PNM_AP_DCP_SET_NAME_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_DCP_SET_NAME_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility
ulSrcId	UINT32	any	-
ulLen	UINT32	8 to 248	Packet Data Length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x9430	PNM_AP_CMD_DCP_SET_NAME_REQ
ulExt	UINT32	0	
ulRoute	UINT32	0	
Structure PNM_AP_DCP_SET_NAME_REQ_DATA_T			
abMac[6]	UINT8[]	Valid MAC address	MAC address of IO Device
usQualifier	UINT16	0,1	Additional qualification of the request
tName[240]	UINT8[]		The new Name of Station

© Hilscher, 2015–2017

Parameter descriptions

Parameter `abMac[6]`

MAC address of IO Device whose name shall be set.

Parameter `usQualifier`

This parameter defines additional qualification of the request. The following table explains the available flags and their meaning:

Name	Numerical Value	Meaning
PNIO_APCTL_DCP_QUALIFIER_STORE_TEMPORARY	0x0000	The name should be assigned temporary. The device will startup without a Name Of Station after a Power Cycle
PNIO_APCTL_DCP_QUALIFIER_STORE_PERMANENT	0x0001	The name should be assigned permanently.

Parameter `abName[240]`

The new NameOfStation to set for the specified IO Device. The length of the name will be derived from the packet length.

© Hilscher, 2015–2017

3.2.5.2 PNM_AP_DCP_SET_CNF_T confirmation

Packet structure reference

```
typedef struct PNM_AP_DCP_SET_CNF_DATA_Ttag PNM_AP_DCP_SET_CNF_DATA_T;
struct PNM_AP_DCP_SET_CNF_DATA_Ttag
{
  uint8_t bDcpError;
};

typedef struct PNM_AP_DCP_SET_CNF_Ttag PNM_AP_DCP_SET_CNF_T;
struct PNM_AP_DCP_SET_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T       tHead;
  PNM_AP_DCP_SET_CNF_DATA_T tData;
};
```

Packet description

Structure PNM_AP_DCP_SET_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	Ignore for future compatibility
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	1	Packet Data Length in bytes
ulId	UINT32	mirrored	-
ulSta	UINT32		=0: no error <> 0: see section Status codes / Error codes on page 276.
ulCmd	UINT32	0x9431	PNM_AP_CMD_DCP_SET_NAME_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure PNM_AP_DCP_SET_CNF_DATA_T			
bDcpError	UINT8		Result status of DCP Operation. Non-zero in case of an Error.

Parameter descriptions

Parameter bDcpError

contains the result status of the DCP operation. The following values are defined according PROFINET Specification:

Name	Numerical Value	Meaning
-	0x00	No error occurred
-	0x01	Unsupported DCP Option - The device does not support setting this parameter
-	0x02	Unsupported DCP Suboption - The device does not support setting this parameter
-	0x03	Setting suboption failed - The device was not able to apply the requested parameter setting for unknown reason
-	0x04	Resource error - The device encountered a resource error while applying the parameter setting
-	0x05	Set not possible - The device cannot apply the parameter at the moment
-	0x06	Set not possible In operation - The device cannot apply the parameter as it is in operation

© Hilscher, 2015–2017

3.2.6 DCP Set IP service

The DCP Set IP service can be used to set the IP address of an IO Device.

Note: Only one DCP Service can be active at a time. It is not possible to use different DCP Services at the same time to the same or different devices. The application must wait for the confirmation of this service before requesting the next DCP Service.

3.2.6.1 PNM_AP_DCP_SET_IP_REQ_T request

Packet structure reference

```
#define PNIO_APCTL_DCP_QUALIFIER_STORE_TEMPORARY                0x0
#define PNIO_APCTL_DCP_QUALIFIER_STORE_PERMANENT                0x1

typedef struct PNM_AP_DCP_SET_IP_REQ_DATA_Ttag PNM_AP_DCP_SET_IP_REQ_DATA_T;
struct PNM_AP_DCP_SET_IP_REQ_DATA_Ttag
{
  uint8_t         abMacAddr[6];
  uint16_t        usQualifier;
  uint32_t        ulIpAddr;
  uint32_t        ulNetmask;
  uint32_t        ulGateway;
};

typedef struct PNM_AP_DCP_SET_IP_REQ_Ttag PNM_AP_DCP_SET_IP_REQ_T;
struct PNM_AP_DCP_SET_IP_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T            tHead;
  PNM_AP_DCP_SET_IP_REQ_DATA_T   tData;
};
```

Packet description

Structure PNM_AP_DCP_SET_IP_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility
ulSrcId	UINT32	any	-
ulLen	UINT32	20	Packet Data Length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x9432	PNM_AP_CMD_DCP_SET_IP_REQ
ulExt	UINT32	0	
ulRoute	UINT32	0	
Structure PNM_AP_DCP_SET_IP_REQ_DATA_T			
abMac[6]	UINT8[]	Valid MAC address	MAC address of IO Device
usQualifier	UINT16	0,1	
ulIpAddr	UINT32		The IP Address to be assigned to the device
ulNetmask	UINT32		The Network Mask to be assigned to the device

© Hilscher, 2015–2017

Structure PNM_AP_DCP_SET_IP_REQ_T			Type: Request
ulGateway	UINT32		The IPv4 Gateway's Address to be assigned to the device.

Parameter descriptions

Parameter abMac[6]

MAC address of IO Device whose name shall be set.

Parameter usQualifier

This parameter defines additional qualification of the request

The following table explains the available flags and their meaning:

Name	Numerical Value	Meaning
PNIO_APCTL_DCP_QUALIFIER_STORE_TEMPORARY	0x0000	The name should be assigned temporary. The device will startup without a Name Of Station after a Power Cycle
PNIO_APCTL_DCP_QUALIFIER_STORE_PERMANENT	0x0001	The name should be assigned permanently.

Parameters ulIpAddr, ulNetmask, ulGateway

The parameters define the new IP settings for the device. If no gateway is required, set the gateway address to the same value as the IP address.

© Hilscher, 2015–2017

3.2.6.2 PNM_AP_DCP_SET_CNF_T confirmation

Packet structure reference

```
typedef struct PNM_AP_DCP_SET_CNF_DATA_Ttag PNM_AP_DCP_SET_CNF_DATA_T;
struct PNM_AP_DCP_SET_CNF_DATA_Ttag
{
  uint8_t bDcpError;
};

typedef struct PNM_AP_DCP_SET_CNF_Ttag PNM_AP_DCP_SET_CNF_T;
struct PNM_AP_DCP_SET_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T       tHead;
  PNM_AP_DCP_SET_CNF_DATA_T tData;
};
```

Packet description

Structure PNM_AP_DCP_SET_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	Ignore for future compatibility
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	1	Packet Data Length in bytes
ulId	UINT32	mirrored	-
ulSta	UINT32		=0: no error <> 0: see section Status codes / Error codes on page 276.
ulCmd	UINT32	0x9431	PNM_AP_CMD_DCP_SET_NAME_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure PNM_AP_DCP_SET_CNF_DATA_T			
bDcpError	UINT8		Result status of DCP Operation. Non-zero in case of an Error.

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulSta`

Contains a non-zero error code if the request was rejected by the PROFINET IO Controller. For details on error codes refer to Status/Error codes.

Parameter `bDcpError`

contains the result status of the DCP operation. The following values are defined according PROFINET Specification:

Name	Numerical Value	Meaning
-	0x00	No error occurred
-	0x01	Unsupported DCP Option - The device does not support setting this parameter
-	0x02	Unsupported DCP Suboption - The device does not support setting this parameter
-	0x03	Setting suboption failed - The device was not able to apply the requested parameter setting for unknown reason
-	0x04	Resource error - The device encountered a resource error while applying the parameter setting
-	0x05	Set not possible - The device cannot apply the parameter at the moment
-	0x06	Set not possible In operation - The device cannot apply the parameter as it is in operation

© Hilscher, 2015–2017

3.2.7 DCP Set Signal Service

The DCP Set Signal Service can be used to activate the Signal LED on a device. The Signal LED can be used to identify the device on wall mount.

Note: Only one DCP Service can be active at a time. It is not possible to use different DCP Services at the same time to the same or different devices. The application must wait for the confirmation of this service before requesting the next DCP Service.

3.2.7.1 PNM_AP_DCP_SET_SIGNAL_REQ_T request

Packet structure reference

```
typedef struct PNM_AP_DCP_SET_SIGNAL_REQ_DATA_Ttag PNM_AP_DCP_SET_SIGNAL_REQ_DATA_T;
struct PNM_AP_DCP_SET_SIGNAL_REQ_DATA_Ttag
{
  uint8_t        abMacAddr[6];
  uint16_t       usQualifier;
};

typedef struct PNM_AP_DCP_SET_SIGNAL_REQ_Ttag PNM_AP_DCP_SET_SIGNAL_REQ_T;
struct PNM_AP_DCP_SET_SIGNAL_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T              tHead;
  PNM_AP_DCP_SET_SIGNAL_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_DCP_SET_SIGNAL_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility
ulSrcId	UINT32	any	-
ulLen	UINT32	8	Packet Data Length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x9434	PNM_AP_CMD_DCP_SET_SIGNAL_REQ
ulExt	UINT32	0	
ulRoute	UINT32	0	
Structure PNM_AP_DCP_SET_SIGNAL_REQ_DATA_T			
abMac[6]	UINT8[]	Valid MAC address	MAC address of IO Device
usQualifier	UINT16	0x0100	DCP Block Qualifier

© Hilscher, 2015–2017

Parameter descriptions

Parameter abMac[6]

MAC address of IO Device whose name shall be set.

Parameter usQualifier

This parameter defines additional qualification of the request. The following values are defined:

Name	Numerical Value	Meaning
PNIO_APCTL_DCP_QUALIFIER_SIGNAL	0x0100	The only valid value in case of DCP Set Signal.

3.2.7.2 PNM_AP_DCP_SET_CNF_T confirmation

Packet structure reference

```
typedef struct PNM_AP_DCP_SET_CNF_DATA_Ttag PNM_AP_DCP_SET_CNF_DATA_T;
struct PNM_AP_DCP_SET_CNF_DATA_Ttag
{
  uint8_t bDcpError;
};

typedef struct PNM_AP_DCP_SET_CNF_Ttag PNM_AP_DCP_SET_CNF_T;
struct PNM_AP_DCP_SET_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T       tHead;
  PNM_AP_DCP_SET_CNF_DATA_T tData;
};
```

Packet description

Structure PNM_AP_DCP_SET_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	Ignore for future compatibility
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	1	Packet Data Length in bytes
ulId	UINT32	mirrored	-
ulSta	UINT32		=0: no error <> 0: see section Status codes / Error codes on page 276.
ulCmd	UINT32	0x9435	PNM_AP_CMD_DCP_SET_SIGNAL_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure PNM_AP_DCP_SET_CNF_DATA_T			
bDcpError	UINT8		Result status of DCP Operation. Non-zero in case of an error.

© Hilscher, 2015–2017

Parameter descriptions

Parameter `bDcpError`

Contains the result status of the DCP operation. The following values are defined according to the PROFINET Specification:

Name	Numerical Value	Meaning
-	0x00	No error occurred
-	0x01	Unsupported DCP Option - The device does not support setting this parameter
-	0x02	Unsupported DCP Suboption - The device does not support setting this parameter
-	0x03	Setting suboption failed - The device was not able to apply the requested parameter setting for unknown reason
-	0x04	Resource error - The device encountered a resource error while applying the parameter setting
-	0x05	Set not possible - The device cannot apply the parameter at the moment
-	0x06	Set not possible In operation - The device cannot apply the parameter as it is in operation

© Hilscher, 2015–2017

3.2.8 DCP Reset Factory Settings service

The DCP Set Reset Factory Settings service can be used to reset parameters of an IO Device.

Note: Only one DCP Service can be active at a time. It is not possible to use different DCP Services at the same time to the same or different devices. The application must wait for the confirmation of this service before requesting the next DCP Service.

3.2.8.1 PNM_AP_DCP_RESET_FACTORY_SETTINGS_REQ_T request

Packet structure reference

```
#define PNIO_APCTL_DCP_QUALIFIER_RESETTOFACTORY_APPLICATION    (1 << 1)
#define PNIO_APCTL_DCP_QUALIFIER_RESETTOFACTORY_COMMUNICATION  (2 << 1)
#define PNIO_APCTL_DCP_QUALIFIER_RESETTOFACTORY_ENGINEERING    (3 << 1)
#define PNIO_APCTL_DCP_QUALIFIER_RESETTOFACTORY_ALL            (4 << 1)
#define PNIO_APCTL_DCP_QUALIFIER_RESETTOFACTORY_DEVICE         (8 << 1)
#define PNIO_APCTL_DCP_QUALIFIER_RESETTOFACTORY_RESTORE        (9 << 1)

typedef struct PNM_AP_DCP_RESET_FACTORY_SETTINGS_REQ_DATA_Ttag
PNM_AP_DCP_RESET_FACTORY_SETTINGS_REQ_DATA_T;
struct PNM_AP_DCP_RESET_FACTORY_SETTINGS_REQ_DATA_Ttag
{
  uint8_t        abMacAddr[6];
  uint16_t       usQualifier;
};

typedef struct PNM_AP_DCP_RESET_FACTORY_SETTINGS_REQ_Ttag
PNM_AP_DCP_RESET_FACTORY_SETTINGS_REQ_T;
struct PNM_AP_DCP_RESET_FACTORY_SETTINGS_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T                          tHead;
  PNM_AP_DCP_RESET_FACTORY_SETTINGS_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_DCP_RESET_FACTORY_SETTINGS_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility
ulSrcId	UINT32	any	-
ulLen	UINT32	8	Packet Data Length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x9436	PNM_AP_CMD_DCP_RESET_FACTORY_SETTINGS_REQ
ulExt	UINT32	0	
ulRoute	UINT32	0	
Structure PNM_AP_DCP_RESET_FACTORY_SETTINGS_REQ_DATA_T			
abMac[6]	UINT8[]	Valid MAC address	MAC address of IO Device
usQualifier	UINT16	0,2,4,6,8,16,18	DCP Block Qualifier

© Hilscher, 2015–2017

Parameter descriptions

Parameter `abMac[6]`

MAC address of PROFINET IO Device whose name shall be set.

Parameter `usQualifier`

This parameter defines additional qualification of the request. The following values are defined for DCP Reset Factory Settings:

Name	Numerical Value	Reset Mode	Meaning
	0x0000	-	Use DCP FactoryReset according to PROFINET Specification 2.2
PNIO_APCTL_DCP_QUALIFIER_RESETTOFACTORY_APPLICATION	0x0002	1	The device shall reset the application data in submodules and modules.
PNIO_APCTL_DCP_QUALIFIER_RESETTOFACTORY_COMMUNICATION	0x0004	2	The device shall reset all PDev Parameters, AR Parameters and Communication Parameters.
PNIO_APCTL_DCP_QUALIFIER_RESETTOFACTORY_ENGINEERING	0x0006	3	The device shall reset all Parameters stored by the engineering system
PNIO_APCTL_DCP_QUALIFIER_RESETTOFACTORY_ALL	0x0008	4	The device shall reset all stored data to factory default values
PNIO_APCTL_DCP_QUALIFIER_RESETTOFACTORY_DEVICE	0x0010	8	The device shall reset all stored data and communication parameters of the device to factory default values. (Out of the Box State)
PNIO_APCTL_DCP_QUALIFIER_RESETTOFACTORY_RESTORE	0x0012	9	The device shall reset its software to the factory images.

Note: PROFINET Devices according to Specification V2.2 support only the DCP FactoryReset variant of this service. Therefore the usQualifier must be set to zero for these devices. For PROFINET Devices according to Specification V2.3 the supported Reset Modes are specified in the GSDML file.

© Hilscher, 2015–2017

3.2.8.2 PNM_AP_DCP_SET_CNF_DATA_T confirmation

Packet structure reference

```
typedef struct PNM_AP_DCP_SET_CNF_DATA_Ttag PNM_AP_DCP_SET_CNF_DATA_T;
struct PNM_AP_DCP_SET_CNF_DATA_Ttag
{
  uint8_t bDcpError;
};

typedef struct PNM_AP_DCP_SET_CNF_Ttag PNM_AP_DCP_SET_CNF_T;
struct PNM_AP_DCP_SET_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T       tHead;
  PNM_AP_DCP_SET_CNF_DATA_T tData;
};
```

Packet description

Structure PNM_AP_DCP_SET_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	Ignore for future compatibility
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	1	Packet Data Length in bytes
ulId	UINT32	mirrored	-
ulSta	UINT32		=0: no error <> 0: see section Status codes / Error codes on page 276.
ulCmd	UINT32	0x9437	PNM_AP_CMD_DCP_RESET_FACTORY_SETTINGS_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure PNM_AP_DCP_SET_CNF_DATA_T			
bDcpError	UINT8		Result status of DCP Operation. Non-zero in case of an error.

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulSta`

Contains a non-zero error code if the request was rejected by the PROFINET IO Controller. For details on error codes refer to Status/Error codes.

Parameter `bDcpError`

contains the result status of the DCP operation. The following values are defined according to the PROFINET Specification:

Name	Numerical Value	Meaning
-	0x00	No error occurred
-	0x01	Unsupported DCP Option - The device does not support setting this parameter
-	0x02	Unsupported DCP Suboption - The device does not support setting this parameter
-	0x03	Setting suboption failed - The device was not able to apply the requested parameter setting for unknown reason
-	0x04	Resource error - The device encountered a resource error while applying the parameter setting
-	0x05	Set not possible - The device cannot apply the parameter at the moment
-	0x06	Set not possible In operation - The device cannot apply the parameter as it is in operation

© Hilscher, 2015–2017

3.2.9 DCP Get service

The DCP Get service can be used to read parameters from a device (IO Controller or IO Device) that is connected to the network via the DCP protocol.

Note: Only one DCP Service can be active at a time. It is not possible to use different DCP Services at the same time to the same or different devices. The application must wait for the confirmation of this service before requesting the next DCP Service.

3.2.9.1 PNM_AP_DCP_GET_REQ_T request

Packet structure reference

```
#define PNM_AP_DCP_GET_OPTION_MACADDR         (0x0001)
#define PNM_AP_DCP_GET_OPTION_IP_PARAM        (0x0002)
#define PNM_AP_DCP_GET_OPTION_TYPEOFSTATION   (0x0004)
#define PNM_AP_DCP_GET_OPTION_NAMEOFSTATION   (0x0008)
#define PNM_AP_DCP_GET_OPTION_DEVICE_IDENT    (0x0010)
#define PNM_AP_DCP_GET_OPTION_DEVICE_ROLE     (0x0020)
#define PNM_AP_DCP_GET_OPTION_DEVICE_INSTANCE (0x0040)

typedef struct PNM_AP_DCP_GET_REQ_DATA_Ttag PNM_AP_DCP_GET_REQ_DATA_T;
struct PNM_AP_DCP_GET_REQ_DATA_Ttag
{
  uint8_t        abMacAddr[6];
  uint16_t       usRequestFlags;
};

typedef struct PNM_AP_DCP_GET_REQ_Ttag PNM_AP_DCP_GET_REQ_T;
struct PNM_AP_DCP_GET_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T                   tHead;
  PNM_AP_DCP_GET_REQ_DATA_T             tData;
};
```

Packet description

Structure PNM_AP_DCP_GET_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility
ulSrcId	UINT32	any	-
ulLen	UINT32	8	Packet Data Length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x9438	PNM_AP_CMD_DCP_GET_REQ
ulExt	UINT32	0	
ulRoute	UINT32	0	
Structure PNM_AP_DCP_SET_SIGNAL_REQ_DATA_T			
abMac[6]	UINT8[]	Valid MAC address	MAC address of IO Device
usRequestFlags	UINT16		Bitmask of values to retrieve

© Hilscher, 2015–2017

Parameter descriptions

Parameter `abMac[6]`

MAC address of IO Device whose name shall be set.

Parameter `usRequestFlags`

This parameter is a bit field defining which parameters shall be retrieved from the target device.

Name	Numerical Value	Meaning
PNM_AP_DCP_GET_OPTION_MACADDR	0x0001	Return the MAC Address of the device (internally forced to be always true)
PNM_AP_DCP_GET_OPTION_IP_PARAM	0x0002	Retrieve the IP Settings of the device
PNM_AP_DCP_GET_OPTION_TYPEOFSTATION	0x0004	Retrieve the Type Of Station of the Device (Equal to Device Type since PROFINET Specification V2.3)
PNM_AP_DCP_GET_OPTION_NAMEOFSTATION	0x0008	Retrieve the Name of Station of the device
PNM_AP_DCP_GET_OPTION_DEVICE_IDENT	0x0010	Retrieve the Vendor and Device ID of the device
PNM_AP_DCP_GET_OPTION_DEVICE_ROLE	0x0020	Retrieve the role of the device
PNM_AP_DCP_GET_OPTION_DEVICE_INSTANCE	0x0040	Retrieve the instance of the device. This Parameter is supported by PROFINET Specification V2.3 devices and above.

© Hilscher, 2015–2017

3.2.9.2 PNM_AP_DCP_GET_CNF_T confirmation

Packet structure reference

```
#define PNM_AP_DCP_GET_OPTION_MACADDR         (0x0001)
#define PNM_AP_DCP_GET_OPTION_IP_PARAM        (0x0002)
#define PNM_AP_DCP_GET_OPTION_TYPEOFSTATION   (0x0004)
#define PNM_AP_DCP_GET_OPTION_NAMEOFSTATION   (0x0008)
#define PNM_AP_DCP_GET_OPTION_DEVICE_IDENT    (0x0010)
#define PNM_AP_DCP_GET_OPTION_DEVICE_ROLE     (0x0020)
#define PNM_AP_DCP_GET_OPTION_DEVICE_INSTANCE (0x0040)

typedef struct PNM_AP_DCP_GET_CNF_DATA_Ttag PNM_AP_DCP_GET_CNF_DATA_T;
struct PNM_AP_DCP_GET_CNF_DATA_Ttag
{
  uint8_t  abMacAddr[6];
  uint16_t usValidFlags;
  uint32_t ulIpAddr;
  uint32_t ulNetmask;
  uint32_t ulGateway;
  uint16_t usVendorId;
  uint16_t usDeviceId;
  uint16_t usInstance;
  uint16_t usRole;
  uint8_t  bLenNameOfStation;
  uint8_t  bLenTypeOfStation;
  uint8_t  abNameOfStation[240];
  uint8_t  abTypeOfStation[240];
  uint8_t  bErrorCount;
  uint8_t  bBlockError;
};

typedef struct PNM_AP_DCP_GET_CNF_Ttag PNM_AP_DCP_GET_CNF_T;
struct PNM_AP_DCP_GET_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T                   tHead;
  PNM_AP_DCP_GET_CNF_DATA_T              tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_DCP_GET_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	Ignore for future compatibility
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	512	Packet Data Length in bytes
ulId	UINT32	mirrored	-
ulSta	UINT32		=0: no error <> 0: see section Status codes / Error codes on page 276.
ulCmd	UINT32	0x9439	PNM_AP_CMD_DCP_GET_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure PNM_AP_DCP_GET_CNF_DATA_T			
abMacAddr[]	UINT8[]		The MAC Address of the target device
usValidFlags	UINT16		Bitmask defining the valid values
ulIpAddr	UINT32		IP Address returned by device
ulNetmask	UINT32		Network mask returned by the device
ulGateway	UINT32		IPv4 Gateway address returned by the device
usVendorId	UINT16		PROFINET Vendor ID returned by the device
usDeviceId	UINT16		PROFINET Device ID returned by the device
usInstance	UINT16		PROFINET Device Instance
usRole	UINT16	1 to 3	PROFINET Role returned by the device
bLenNameOfStation	UINT8	0 to 240	Length of the Name Of Station returned by the device
bLenTypeOfStation	UINT8	0 to 240	Length of the Type Of Station returned by the device
abNameOfStation[]	UINT8[]		The Name Of Station of the Device
abTypeOfStation[]	UINT8		The Type Of Station of the Device
bErrorCount	UINT8		The number of DCP Get Option Errors returned by the device
bBlockError	UINT8		The last DCP Option Block error returned by the device

© Hilscher, 2015–2017

Parameter descriptions

Parameter `usValidFlags`

This parameter is a bitmask which defines which parameters have been retrieved from the target device. The following values are defined:

Name	Numerical Value	Meaning
PNM_AP_DCP_GET_OPTION_MACADDR	0x0001	Return the MAC address of the device (taken from request packet, internally forced to be always true)
PNM_AP_DCP_GET_OPTION_IP_PARAM	0x0002	Retrieve the IP Settings of the device
PNM_AP_DCP_GET_OPTION_TYPEOFSTATION	0x0004	Retrieve the Type Of Station of the Device (Equal to Device Type since PROFINET Specification V2.3)
PNM_AP_DCP_GET_OPTION_NAMEOFSTATION	0x0008	Retrieve the Name of Station of the device
PNM_AP_DCP_GET_OPTION_DEVICE_IDENT	0x0010	Retrieve the Vendor and Device ID of the device
PNM_AP_DCP_GET_OPTION_DEVICE_ROLE	0x0020	Retrieve the role of the device
PNM_AP_DCP_GET_OPTION_DEVICE_INSTANCE	0x0040	Retrieve the instance of the device. This Parameter is supported by PROFINET Specification V2.3 devices and above.

© Hilscher, 2015–2017

3.2.10 Get Logbook Service

The controller maintains an internal logbook for each device and one for itself. The logbook contains an ordered list of events occured since last Download finished service.

3.2.10.1 PNM_AP_GET_LOGBOOK_REQ_T request

Packet structure reference

```
typedef struct PNM_AP_GET_LOGBOOK_REQ_DATA_Ttag PNM_AP_GET_LOGBOOK_REQ_DATA_T;
struct PNM_AP_GET_LOGBOOK_REQ_DATA_Ttag
{
  PNM_AP_DEVICEHANDLE_T     usDeviceHandle;
  uint16_t                  usMaxEntries;
};
typedef struct PNM_AP_GET_LOGBOOK_REQ_Ttag PNM_AP_GET_LOGBOOK_REQ_T;
struct PNM_AP_GET_LOGBOOK_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T           tHead;
  PNM_AP_GET_LOGBOOK_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_GET_LOGBOOK_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	
ulSrc	UINT32	Handle (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility
ulSrcId	UINT32	any	Ignored by protocol stack
ulLen	UINT32	4	Packet data length in bytes
ulId	UINT32	any	Ignored by protocol stack
ulSta	UINT32	0	-
ulCmd	UINT32	0x94F0	PNM_AP_CMD_GET_LOGBOOK_REQ
ulExt	UINT32	0	
ulRoute	UINT32	0	
Structure PNM_AP_GET_LOGBOOK_REQ_DATA_T			
usDeviceHandle	PNM_AP_DEVICEHANDLE_T	0 1 to 128	PROFINET IO Controller logbook Handle of AR for which to retrieve the logbook
usMaxEntries	UINT16	1 to 64	Number of entries to return in response

Parameter descriptions

Parameter usDeviceHandle

Reference to the AR for which to retrieve the logbook. Set to zero to retrieve the PROFINET IO Controller logbook.

Parameter usMaxEntries

Maximum number of entries to return in confirmation packet

© Hilscher, 2015–2017

3.2.10.2 PNM_AP_GET_LOGBOOK_CNF_T confirmation

Packet structure reference

```
enum PNM_AP_LOGBOOK_EVENT_Etag
{
  PNM_AP_LOGBOOK_CONFIGSTATE                  = 0x0001,
  PNM_AP_LOGBOOK_CONFIGLOCKED                 = 0x0002,

  PNM_AP_LOGBOOK_HIF_WDGERROR                 = 0x0100,

  PNM_AP_LOGBOOK_EVENT_AR_NOTESTABLISHED      = 0x8000,
  PNM_AP_LOGBOOK_EVENT_AR_ESTABLISHED         = 0x8001,
  PNM_AP_LOGBOOK_EVENT_AR_ABORT               = 0x8002,
  PNM_AP_LOGBOOK_EVENT_AR_LOW_PRIO_ALARM      = 0x8003,
  PNM_AP_LOGBOOK_EVENT_AR_HIGH_PRIO_ALARM     = 0x8004,
  PNM_AP_LOGBOOK_EVENT_AR_RPC_READ_FAILED     = 0x8005,
  PNM_AP_LOGBOOK_EVENT_AR_RPC_WRITE_FAILED    = 0x8006,

  PNM_AP_LOGBOOK_EVENT_STACK_BUSSTATE         = 0x8100,
};
typedef enum PNM_AP_LOGBOOK_EVENT_Etag PNM_AP_LOGBOOK_EVENT_E;

typedef struct PNM_AP_LOGBOOK_RECORD_Ttag PNM_AP_LOGBOOK_RECORD_T;
struct PNM_AP_LOGBOOK_RECORD_Ttag
{
  uint64_t ulCycleCounter;
  uint16_t usEvent;
  uint16_t usPadding;
  uint32_t ulAdditionalValue;
};

typedef struct PNM_AP_GET_LOGBOOK_CNF_DATA_Ttag PNM_AP_GET_LOGBOOK_CNF_DATA_T;
struct PNM_AP_GET_LOGBOOK_CNF_DATA_Ttag
{
  PNM_AP_DEVICEHANDLE_T     usDeviceHandle;
  uint16_t                  usNumEntries;
  PNM_AP_LOGBOOK_RECORD_T   atEntries[64];
};

typedef struct PNM_AP_GET_LOGBOOK_CNF_Ttag PNM_AP_GET_LOGBOOK_CNF_T;
struct PNM_AP_GET_LOGBOOK_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T           tHead;
  PNM_AP_GET_LOGBOOK_CNF_DATA_T tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_GET_LOGBOOK_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		
ulSrc	UINT32		
ulDestId	UINT32	0	Ignore for future compatibility.
ulSrcId	UINT32	mirrored	
ulLen	UINT32	4 to 1028	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section Status codes / Error codes on page 276.
ulCmd	UINT32	0x94F1	PNM_AP_CMD_GET_LOGBOOK_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure PNM_AP_DWNL_FIN_CNF_DATA_T			
usDeviceHandle	PNM_AP_DEVICEHANDLE_T	0 1 to 128	PROFINET IO Controller logbook Handle of AR to retrieve the logbook for
usNumEntries	UINT16	0 to 64	Number of entries in logbook
atEntries[]	PNM_AP_LOGBOOK_RECORD_T		The newest entries in the logbook in ascending time order.
Structure PNM_AP_LOGBOOK_RECORD_T			
ulCycleCounter	UINT64		Elapsed time since power-up of the PROFINET IO Controller in Units of 31.25 µs
usEvent	UINT16		Id of the Event
usPadding	UINT16	0	Ignore for future compatibility.
ulAdditionalValue	UINT32		Event parameter

Parameter descriptions

Parameter usDeviceHandle

A reference to the AR the logbook was retreived for. Zero if the PROFINET IO Controllers logbook was received

Parameter usNumEntries

The actual number of entries returned in atEntries.

© Hilscher, 2015–2017

Parameter `atEntries[]`, `ulCyclecounter`, `usEvent`, `ulAdditionalValue`

The entries of the logbook ordered in ascending time. The parameter `ulCycleCounter` contains the value of the PROFINET cycle when the event occurred. The following values are defined for the event id and the event parameter:

Name	Numerical Value	Meaning	Event Parameter
AR related events			
PNM_AP_LOGBOOK_EVENT_AR_NOTESTABLISHED	0x8000	The AR was not established	PROFINET Error Status
PNM_AP_LOGBOOK_EVENT_AR_ESTABLISHED	0x8001	The AR was established	0: No Module Diff Block was reported by the device 1: A Module Diff Block was reported by the device
PNM_AP_LOGBOOK_EVENT_AR_ABORT	0x8002	The AR was aborted	PROFINET Error Status
PNM_AP_LOGBOOK_EVENT_AR_LOW_PRIO_ALARM	0x8003	A low priority PROFINET alarm was received	Lower 16 Bit: Handle of Submodule Upper 16 Bit: Alarm Type
PNM_AP_LOGBOOK_EVENT_AR_HIGH_PRIO_ALARM	0x8004	A high priority PROFINET alarm was received	Lower 16 Bit: Handle of Submodule Upper 16 Bit: Alarm Type
PNM_AP_LOGBOOK_EVENT_AR_RPC_READ_FAILED	0x8005	A Record Read Request failed	PROFINET Error Status
PNM_AP_LOGBOOK_EVENT_AR_RPC_WRITE_FAILED	0x8006	A Record Write Request failed	PROFINET Error Status
PROFINET IO Controller related events			
PNM_AP_LOGBOOK_CONFIGSTATE	0x0001	Change of configuration state	0: Not configured 1: Packet configuration pending (waiting for *Download finished service*) 2: Configured by Packet
PNM_AP_LOGBOOK_CONFIGLOCKED	0x0002	Configuration Lock State Changed	0: Not locked 1: Locked
PNM_AP_LOGBOOK_HIF_WDGERROR	0x0100	Application DPM Watchdog Timeout	Ignore for future compatibility.
PNM_AP_LOGBOOK_EVENT_STACK_BUSSTATE	0x8100	The bus state changed	0: Bus Off 1: Start communication was requested 2: Bus On 3: Stop communication was requested 4: Shutting down: waiting for pending RPC calls

© Hilscher, 2015–2017

3.2.11 Get AR Vendor Block Response Service

This service can be used by the application to retrieve the response of a PROFINET IO Device to an AR Vendor Block Request. The AR Vendor Block Request must have been configured previously using Configure AR parameters service.

3.2.11.1 PNM_AP_GET_ARVENDORBLOCK_RESPONSE_REQ_T request

Packet structure reference

```
typedef struct PNM_AP_GET_ARVENDORBLOCK_RESPONSE_REQ_DATA_Ttag
PNM_AP_GET_ARVENDORBLOCK_RESPONSE_REQ_DATA_T;
struct PNM_AP_GET_ARVENDORBLOCK_RESPONSE_REQ_DATA_Ttag
{
  PNM_AP_ARVENDORBLOCKHANDLE_T usArVendorBlockHandle;
};

typedef struct PNM_AP_GET_ARVENDORBLOCK_RESPONSE_REQ_Ttag
PNM_AP_GET_ARVENDORBLOCK_RESPONSE_REQ_T;
struct PNM_AP_GET_ARVENDORBLOCK_RESPONSE_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T                          tHead;
  PNM_AP_GET_ARVENDORBLOCK_RESPONSE_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_GET_ARVENDORBLOCK_RESPONSE_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	Handle (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility
ulSrcId	UINT32	any	-
ulLen	UINT32	2	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x9420	PNM_AP_CMD_GET_ARVENDORBLOCK_RESPONSE_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	0	-
Structure PNM_AP_CFG_AR_PRM_DATA_T			
usArVendorBlockHandle	UINT16	1 to 256	Handle of the previously configured AR Vendor Block Request associated with the desired AR Vendor Block Response

Parameter descriptions

Parameter usArVendorBlockHandle

The handle of the previously configured AR Vendor Block. The confirmation packet will contain the associated AR Vendor Block Response Value.

© Hilscher, 2015–2017

3.2.11.2 PNM_AP_GET_ARVENDORBLOCK_RESPONSE_CNF_DATA_T confirmation

Packet structure reference

```
typedef struct PNM_AP_GET_ARVENDORBLOCK_RESPONSE_CNF_DATA_Ttag
PNM_AP_GET_ARVENDORBLOCK_RESPONSE_CNF_DATA_T;
struct PNM_AP_GET_ARVENDORBLOCK_RESPONSE_CNF_DATA_Ttag
{
  PNM_AP_ARVENDORBLOCKHANDLE_T usArVendorBlockHandle;
  uint16_t                     usAPStructureIdentifier;
  uint32_t                     ulApi;
  uint8_t                      abData[ARRAYS_OF_LENGTH_ZERO];
};
typedef struct PNM_AP_GET_ARVENDORBLOCK_RESPONSE_CNF_Ttag
PNM_AP_GET_ARVENDORBLOCK_RESPONSE_CNF_T;
struct PNM_AP_GET_ARVENDORBLOCK_RESPONSE_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T                          tHead;
  PNM_AP_GET_ARVENDORBLOCK_RESPONSE_CNF_DATA_T tData;
};
```

Packet description

Structure PNM_AP_CFG_AR_PRM_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	-
ulSrcId	UINT32	mirrored	This field will contain the same value as the request packet
ulLen	UINT32	8 + AR Vendor Block Response Data Length	Packet data length in bytes
ulId	UINT32	mirrored	This field will contain the same value as the request packet
ulSta	UINT32		=0: no error <> 0: see section Status codes / Error codes on page 276.
ulCmd	UINT32	0x9421	PNM_AP_CMD_GET_ARVENDORBLOCK_RESPONSE_CNF
ulExt	UINT32	any	Ignore.
ulRoute	UINT32	any	Ignore.
Structure PNM_AP_GET_ARVENDORBLOCK_RESPONSE_CNF_DATA_T			
usArVendorBlockHandle	UINT16	1 to 256	Handle to the associated is AR Vendor Block.
usAPStructureIdentifier	UINT16	0 to 0x7FFF, 0x8000	AP Structure Identifier as configured previously
ulApi	UINT32	any	API as configured previously
abData[]	UINT8[]		Byte array containing the AR Vendor Block Response Data.

© Hilscher, 2015–2017

Parameter descriptions

Parameter `usArVendorBlockHandle`

The handle of the requested AR Vendor Block.

Parameters `usAPStructureIdentifier`, `ulApi`

These parameters contain the values configured previously using Configure AR parameters service.

Parameter `abData[]`

The AR Vendor Block Response Data returned by the device in its connect response. The actual length of the data can be calculated from tHead.ulLen.

© Hilscher, 2015–2017

3.2.12 Set AR Status service

This service can be used to manipulate the status of an AR at runtime.

3.2.12.1 PNM_AP_SET_ARSTATUS_REQ request

Packet structure reference

```
enum PNM_AP_SET_ARSTATUS_FLAGS_ETag
{
  PNM_AP_SET_ARSTATUS_DISABLED = 0x04,
};
typedef enum PNM_AP_SET_ARSTATUS_FLAGS_ETag PNM_AP_SET_ARSTATUS_FLAGS_E;

typedef struct PNM_AP_SET_ARSTATUS_REQ_DATA_Ttag PNM_AP_SET_ARSTATUS_REQ_DATA_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_SET_ARSTATUS_REQ_DATA_Ttag
{
  PNM_AP_DEVICEHANDLE_T usDeviceHandle;
  uint16_t              usPadding;
  uint32_t              ulStatus;
};

typedef struct PNM_AP_SET_ARSTATUS_REQ_Ttag PNM_AP_SET_ARSTATUS_REQ_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_SET_ARSTATUS_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T             tHead;
  PNM_AP_SET_ARSTATUS_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_SET_ARSTATUS_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Unused by stack. Set to zero for future compatibility.
ulSrcId	UINT32	any	-
ulLen	UINT32	8	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x942E	PNM_AP_CMD_SET_AR_STATUS_REQ
ulExt	UINT32	0	Set to zero
ulRoute	UINT32	0	Set to zero
Structure PNM_AP_SET_ARSTATUS_REQ_DATA_T			
usDeviceHandle	UINT16	1 to 128	Handle of AR to configure
usPadding	UINT16	0	Reserved for future use. Set to zero.
ulStatus	UINT32		Bitmask to configure the AR

© Hilscher, 2015–2017

Parameter descriptions

Parameter `usDeviceHandle`

The handle of the AR to configure.

Parameter `ulStatus`

A bitmask defining the new properties of an AR.

Name	Value	Meaning
PNM_AP_SET_ARSTATUS_ENABLED	0x00000000	If no bit is set, this particular AR will be enabled.
PNM_AP_SET_ARSTATUS_DISABLED	0x00000004	If this bit is set, this particular AR will be disabled. If the AR is connected, the connection will be aborted.

3.2.12.2 PNM_AP_SET_ARSTATUS_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T          PNM_AP_SET_ARSTATUS_CNF_T;
```

Packet description

Structure PNM_AP_SET_ARSTATUS_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	mirrored	-
ulSta	UINT32		=0: no error <> 0: see section Status codes / Error codes on page 276.
ulCmd	UINT32	0x942F	PNM_AP_CMD_SET_AR_STATUS_CNF
ulExt	UINT32		-
ulRoute	UINT32		Ignore

© Hilscher, 2015–2017

3.2.13 Establish Device Access AR

This service can be used to establish a Device Access AR.

Note: The confirmation packet will be sent after the DA-AR has been established. If it was not possible to establish a DA-AR the confirmation packet will contain an error.

. This process may take several seconds.

3.2.13.1 PNM_AP_ESTABLISH_DAAR_REQ_T request

Packet structure reference

```
enum PNM_AP_CFG_IOD_AR_TYPE_Etag
{
  PNM_AP_CFG_IOD_AR_TYPE_SUPERVISOR       = 0x06,
};
typedef enum PNM_AP_CFG_IOD_AR_TYPE_Etag PNM_AP_CFG_IOD_AR_TYPE_E;

enum PNM_AP_CFG_IOD_AR_PROP_FLAGS_Etag
{
  PNM_AP_CFG_IOD_AR_PROP_FLAG_DEVICEACCESS = 0x00000100,
};

typedef uint16_t PNM_AP_DEVICEHANDLE_T;

#define PNM_AP_ESTABLISH_DAAR_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_ESTABLISH_DAAR_DATA_Ttag PNM_AP_ESTABLISH_DAAR_DATA_T;
struct PNM_AP_ESTABLISH_DAAR_DATA_Ttag
{
  uint32_t              ulStructVersion;
  PNM_AP_DEVICEHANDLE_T usDeviceHandle;
  uint8_t               bARType;
  uint8_t               bPadding;
  uint32_t              ulFlags;
  PNM_UUID_T            tArUuid;
  uint32_t              ulArProperties;
  uint16_t              usVendorID;
  uint16_t              usDeviceID;
  uint16_t              usInstanceID;
  uint16_t              ausReserved[3];
  uint8_t               abNameOfStation[240];
  uint32_t              ulIPAddr;
  uint32_t              ulNetworkMask;
  uint32_t              ulGatewayAddr;
};

typedef struct PNM_AP_ESTABLISH_DAAR_REQ_Ttag PNM_AP_ESTABLISH_DAAR_REQ_T;
struct PNM_AP_ESTABLISH_DAAR_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T               tHead;
  PNM_AP_ESTABLISH_DAAR_DATA_T      tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_ESTABLISH_DAAR_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	Handle (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility
ulSrcId	UINT32	any	Ignored by protocol stack
ulLen	UINT32	296	Packet data length in bytes
ulId	UINT32	any	Ignored by protocol stack
ulSta	UINT32	0	-
ulCmd	UINT32	0x941C	PNM_AP_CMD_ESTABLISH_DAAR_REQ command
ulExt	UINT32	0	-
ulRoute	UINT32	0	-
Structure PNM_AP_ESTABLISH_DAAR_REQ_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usDeviceHandle	PNM_AP_DEVICEHANDLE_T	0xF000	Handle of device acccess AR
bARType	UINT8	0x06	AR Type
bPadding	UINT8	0	Set to zero for future compatibility
ulFlags	UINT32	0	Flags controlling AR behavior
tArUuid	PNM_UUID_T		UUID associated with this AR.
ulArProperties	UINT32	0x00000100	Properties of this AR.
usVendorID	UINT16	GSDML	PROFINET Vendor ID of the Device
usDeviceID	UINT16	GSDML	PROFINET Device ID of the Device
usInstanceID	UINT16	GSDML	PROFINET Instance ID of the Device
ausReserved[3]	UINT16[]	0	Reserved. Set to Rero
abNameOfStation[240]	UINT8[]		NameOfStation of the IO Device
ulIPAddr	UINT32		IP address
ulNetworkMask	UINT32		Network mask
ulGatewayAddr	UINT32		Gateway address (i.e. IP address of gateway)

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulStructVersion`

Structure version of the configuration structure. Used for future extensions.

Parameter `usDeviceHandle`

This parameter specifies the handle of the AR to be configured. For device access AR this handle is fixed to the value PNM_AP_DEVICEHANDLE_DAAR_FIRST (0xF000).

Parameter `bARType`

This parameter describes the type of the AR

Name	Numeric value	Meaning
PNM_AP_CFG_IOD_AR_TYPE_SUPERVISOR	0x06	AR Type for supervisor communication

Parameter `ulFlags`

The parameter ulFlags is a bitmask of flag bits specifying additional properties of the AR. For device access AR this field shall be set to zero.

Parameter `tArUUID`

Unique identifier of the AR. The AR UUID shall be generated by the engineering software or application. It is a unique id of the AR and must be non-zero. Each new device access AR must use a new UUID.

Parameter `ulArProperties`

The parameter ulARProperties is a bitmask describing PROFINET Properties of this AR. For device access AR the value PNM_AP_CFG_IOD_AR_PROP_FLAG_DEVICEACCESS must be used.

Name	Numeric value	Meaning
PNM_AP_CFG_IOD_AR_PROP_FLAG_DEVICEACCESS	0x00000100	A device access ar shall be established

Parameter `usVendorID`

VendorID to be used by IO Controller for addressing the IO Device. (Extract from GSDML file)

Parameter `usDeviceID`

DeviceID to be used by IO Controller for addressing the IO Device (Extract from GSDML file)

Parameter `usInstanceID`

The instance ID of the device. (Extract from GSDML file, Attribute "ObjectUUID Instance")

Parameter `abNameOfStation[240]`

The parameter abNameOfStation is the PROFINET Name Of Station to be used for this AR. The PROFINET IO Controller will establish the AR with a PROFINET IO Device with that NameOfStation. If multiple devices with same NameOfStation exist, no connection will be made. The field should be zero padded to full field length.

© Hilscher, 2015–2017

Parameter `ulIPAddr`

IP address to be assigned to the PROFINET IO Device. This IP Address Setting is to be specified by the application and will be assigned to the PROFINET IO Device by the PROFINET IO Controller on Startup.

Parameter `ulNetmask`

Network Mask to be assigned to the PROFINET IO Device. This IP Address Setting is to be specified by the application and will be assigned to the PROFINET IO Device by the PROFINET IO Controller on Startup.

Parameter `ulGatewayAddr`

Gateway Address to be assigned to the PROFINET IO Device. This IP Address Setting is to be specified by the application and will be assigned to the PROFINET IO Device by the PROFINET IO Controller on Startup.

3.2.13.2 PNM_AP_ESTABLISH_DAAR_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T          PNM_AP_ESTABLISH_DAAR_CNF_T;
```

Packet description

Structure `PNM_AP_ESTABLISH_DAAR_CNF_T`			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure `PNM_AP_PCK_HEADER_T`			
`ulDest`	UINT32		-
`ulSrc`	UINT32		-
`ulDestId`	UINT32	0	-
`ulSrcId`	UINT32	mirrored	-
`ulLen`	UINT32	0	Packet data length in bytes
`ulId`	UINT32	mirrored	-
`ulSta`	UINT32		=0: no error <> 0: see section Status codes / Error codes on page 276.
`ulCmd`	UINT32	0x941D	`PNM_AP_CMD_ESTABLISH_DAAR_CNF`
`ulExt`	UINT32	0	-
`ulRoute`	UINT32		Ignore

© Hilscher, 2015–2017

3.2.14 Release Device Access AR

This service can be used to release a previously established device access AR.

3.2.14.1 PNM_AP_RELEASE_DAAR_REQ_T request

Packet structure reference

```
enum PNM_AP_RELEASE_DAAR_FLAGS_Etag
{
  PNM_AP_RELEASE_DAAR_RELEASE = 0x04,
};
typedef enum PNM_AP_RELEASE_DAAR_FLAGS_Etag PNM_AP_RELEASE_DAAR_FLAGS_E;

typedef struct PNM_AP_RELEASE_DAAR_REQ_DATA_Ttag PNM_AP_RELEASE_DAAR_REQ_DATA_T;
struct PNM_AP_RELEASE_DAAR_REQ_DATA_Ttag
{
  PNM_AP_DEVICEHANDLE_T usDeviceHandle;
  uint16_t              usPadding;
  uint32_t              ulFlags;
};

typedef struct PNM_AP_RELEASE_DAAR_REQ_Ttag PNM_AP_RELEASE_DAAR_REQ_T;
struct PNM_AP_RELEASE_DAAR_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T            tHead;
  PNM_AP_RELEASE_DAAR_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_RELEASE_DAAR_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Unused by stack. Set to zero for future compatibility.
ulSrcId	UINT32	any	-
ulLen	UINT32	8	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x941E	PNM_AP_CMD_RELEASE_DAAR_REQ
ulExt	UINT32	0	Set to zero
ulRoute	UINT32	0	Set to zero
Structure PNM_AP_RELEASE_DAAR_REQ_DATA_T			
usDeviceHandle	UINT16	0xF000	Handle of AR to release
usPadding	UINT16	0	Reserved for future use. Set to zero.
ulFlags	UINT32	0x04	Flags associated with the service

© Hilscher, 2015–2017

Parameter descriptions

Parameter `usDeviceHandle`

The handle of the AR to release. For device access AR this handle is fixed to the value PNM_AP_DEVICEHANDLE_DAAR_FIRST (0xF000).

Parameter `ulFlags`

A bitmask defining the flags associated with the service properties of an AR. For device access AR the value PNM_AP_RELEASE_DAAR_RELEASE must be used (0x00000004).

Name	Value	Meaning
PNM_AP_RELEASE_DAAR_RELEASE	0x00000004	Release the AR associated with this service

3.2.14.2 PNM_AP_RELEASE_DAAR_CNF_T confirmation

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T          PNM_AP_RELEASE_DAAR_CNF_T;
```

Packet description

Structure PNM_AP_RELEASE_DAAR_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32		-
ulSrc	UINT32		-
ulDestId	UINT32	0	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	mirrored	-
ulSta	UINT32		=0: no error <> 0: see section Status codes / Error codes on page 276.
ulCmd	UINT32	0x941F	PNM_AP_CMD_RELEASE_DAAR_CNF
ulExt	UINT32		-
ulRoute	UINT32		Ignore

© Hilscher, 2015–2017

3.3 Network scan

The PROFINET IO Controller V3 supports scanning the network for PROFINET devices. This scan is non intrusive and can be used to discover the PROFINET devices connected to the network ports of the controller. The scan will not deliver any information about the module configuration of a particular device. This might require intrusive changes to the PROFINET network like assigning IP addresses and/or name of station. Nevertheless the application can discover the module configuration of the devices after scanning the network by using additional services provided by the controller.

The network scan is implemented according the "Network scan" Application note using the rcX Bus scan service and rcX Get Device Info service. The controller will perform the scan in two stages to support scanning large networks. The following figure shows the sequence of a bus scan.

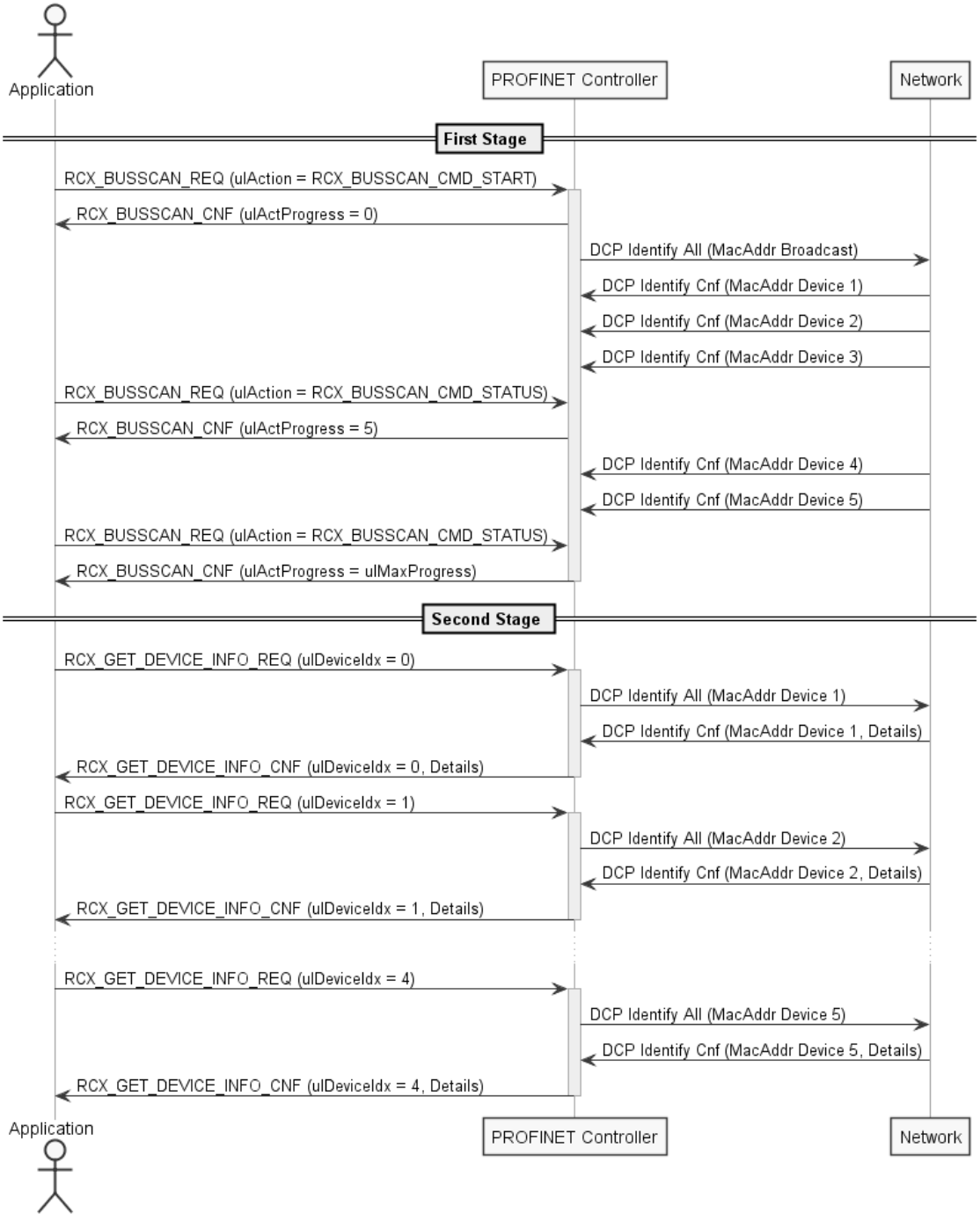

Figure 11: Network scan sequence

© Hilscher, 2015–2017

The rcX Get Device Info service is a generic service used in all supporting master implementations. Nevertheless the confirmation packets contain protocol specific information after the data part. The following definitions apply for the PROFINET Controller V3.

Packet structure reference

```
typedef struct PNM_EXTDIA_DEVICEINFO_Ttag PNM_EXTDIA_DEVICEINFO_T;
__PACKED_PRE struct __PACKED_POST PNM_EXTDIA_DEVICEINFO_Ttag
{
  uint16_t usVendorId;
  uint16_t usDeviceId;
  uint32_t ulIpAddress;
  uint32_t ulNetmask;
  uint32_t ulGateway;
  uint16_t usLenName;
  uint16_t usLenType;
  uint16_t usDeviceRole;
  uint8_t abMacAddress[6];
  uint8_t abNameOfStation[240];
  uint8_t abTypeOfStation[240];
};

typedef __TLR_PACKED_PRE struct RCX_GET_DEVICE_INFO_CNF_DATA_Ttag
{
  TLR_UINT32 ulDeviceIdx;
  TLR_UINT32 ulStructId;
} __TLR_PACKED_POST RCX_GET_DEVICE_INFO_CNF_DATA_T;

typedef struct RCX_GET_DEVICE_INFO_CNF_Ttag
{
  TLR_PACKET_HEADER_T            tHead;
  RCX_GET_DEVICE_INFO_CNF_DATA_T tData;
  PNM_EXTDIA_DEVICEINFO_T        tInfo;
} __TLR_PACKED_POST RCX_GET_DEVICE_INFO_CNF_T;
```


© Hilscher, 2015–2017

Packet description

Structure RCX_GET_DEVICE_INFO_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure TLR_PACKET_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	Destination Queue-Handle
ulSrc	UINT32	0 ... 2^{32}-1	Source Queue-Handle
ulDestId	UINT32	0	Destination End Point Identifier, specifying the final receiver of the packet within the Destination Process. Set to 0 for the Initialization Packet
ulSrcId	UINT32	0 ... 2^{32}-1	Source end point identifier, specifying the origin of the packet inside the source process
ulLen	UINT32	516	Packet data length in bytes
ulId	UINT32		Packet Identification as unique number generated by the source process of the packet
ulSta	UINT32		See section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x2f25	RCX_GET_DEVICE_INFO_CNF_T - Command
ulExt	UINT32	0	Extension not in use, set to zero for compatibility reasons
ulRoute	UINT32		Routing, do not touch
Structure RCX_GET_DEVICE_INFO_CNF_DATA_T			
ulDeviceIdx	UINT32		Value taken from request
ulStructId	UINT32	5389	Struct identifier for PROFINET Device Info
usVendorId	UINT16		PROFINET Vendor Id of the device
usDeviceId	UINT16		PROFINET Device Id of the device
ulIpAddress	UINT32		Current IP address of the device
ulNetmask	UINT32		Current IP network mask of the device
ulGateway	UINT32		Current IP gateway address of the device
usLenName	UINT16	0 to 240	Length of name of station of the device
usLenType	UINT16	0 to 240	Length of type of station of the device
usDeviceRole	UINT16	0 ... 3	Role of the device
abNameOfStation [240]	UINT8		Name of station of the device
abTypeOfStation [240]	UINT8		Type of station of the device

Table 13: RCX_GET_DEVICE_INFO_CNF_T - Get User Parameter Data Request

Parameter descriptions

Parameter ulDeviceIdx

The identifier of the device associated with the device info.

Parameter ulStructId

The identifier of the structure type. Always 5389 for the PROFINET Controller

© Hilscher, 2015–2017

3.4 Device Access AR (DA-AR)

3.4.1 Overview

Beside the standard IO-AR the PROFINET IO Controller implements Device Access AR. The main use of this AR type is writing Identification & Maintenance record objects from the engineering software. The Device Access AR is internally handled in a similar way to standard IO ARs. Its identified by a device handle. The following device handle is used for an DA-AR:

Name	Value
PNM_AP_DEVICEHANDLE_DAAR_FIRST	0xF000

Table 14: Device Handle for Device Access AR

In contrast to an IO-AR, the DA-AR is configured and established at runtime. A DA-AR has no IOCRs, no submodules and no records. Record objects are read and written by the application using acyclic services. The following services are used in conjunction with an Device Access AR:

Name	Usage	Page
Establish Device Access AR	Connect a Device Access AR to an Device	165
Release Device Access AR	Terminate Device Access AR	169
Read Record service (legacy)	Read record object	241
Write Record service (legacy)	Write record object	245

Table 15: Device Access AR Services

© Hilscher, 2015–2017

3.4.2 Usage

The general application sequence for using a DA-AR is shown in the following figure.

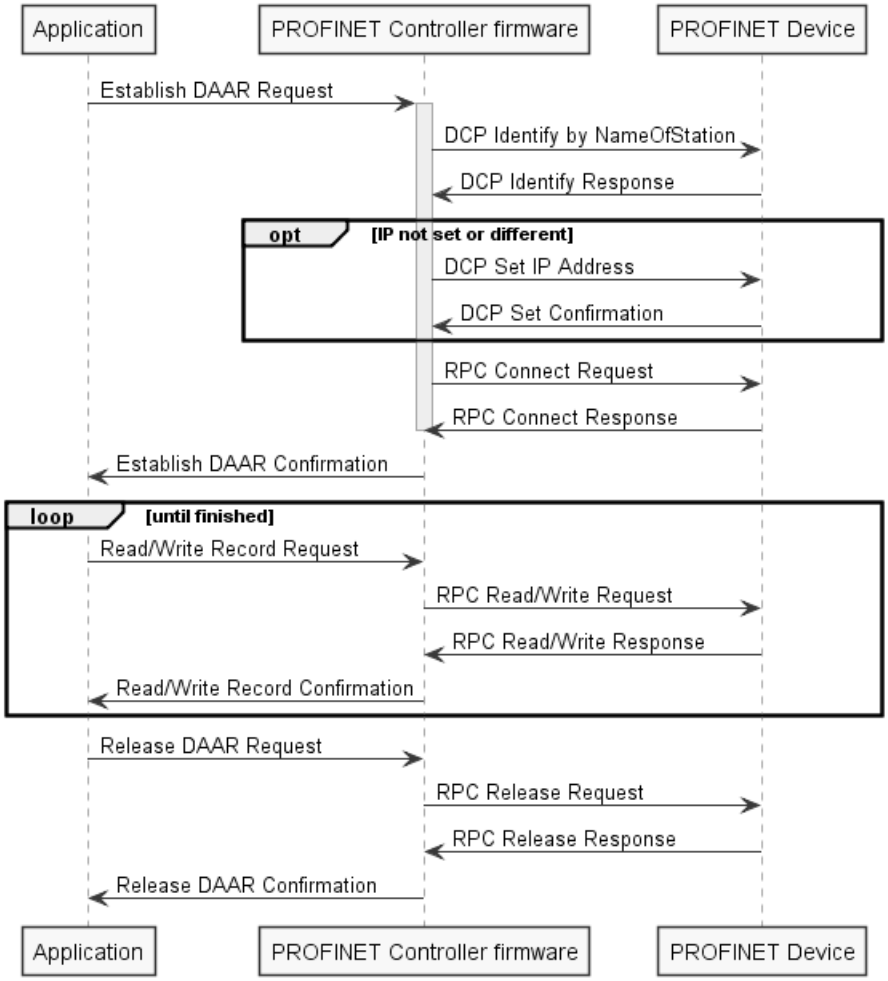

Figure 12: Device Access AR Sequence

According to the PROFINET specification the PROFINET Device will internally abort the DA-AR after the CMI timeout. The timeout starts initially with the RPC Connect Response and is restarted on each RPC Read/Write response. The Controller must sent the next RPC Request before the CMI timeout has passed, otherwise the Device will reject the RPC request. The PROFINET specification defines no direct indication from Device to Controller in case of a CMI timeout for an Device Access AR.

© Hilscher, 2015–2017

3.5 Acyclic indications

This section contains the description of packets used for acyclic servies the protocol stack sends to the application for processing.

3.5.1 Receive Alarm service

The receive alarm service is used by the PROFINET Controller firmware to indicate the reception of a PROFINET Alarm to the host application. The indication will be used for all alarm types which do not use diagnosis encoding for their payload. After reception of this service and processing of the alarm data the alarm must be confirmed by the host application using *Acknowledge Alarm service* on page 134. Otherwise the associated device cannot send other alarms of the same priority.

Note: This service will not be used for alarm types configured for automatic alarm handling in *Configure IO Controller service* on page 33. These alarms will be handled internally by the PROFINET controller firmware.

3.5.1.1 PNM_AP_RECV_ALARM_IND_T indication

Packet structure reference

```
typedef struct PNM_AP_RECV_ALARM_IND_DATA_Ttag PNM_AP_RECV_ALARM_IND_DATA_T;
struct PNM_AP_RECV_ALARM_IND_DATA_Ttag
{
  PNM_AP_DEVICEHANDLE_T     usDeviceHandle;
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle;
  uint16_t                  usAlarmType;
  uint16_t                  usAlarmPriority;
  uint32_t                  ulApi;
  uint16_t                  usSlot;
  uint16_t                  usSubslot;
  uint16_t                  usAlarmSpecifier;
  uint16_t                  usAlarmDataLen;
  uint8_t                   abAlarmData[ARRAYS_OF_LENGTH_ZERO];
};

typedef struct PNM_AP_RECV_ALARM_IND_Ttag PNM_AP_RECV_ALARM_IND_T;
struct PNM_AP_RECV_ALARM_IND_Ttag
{
  PNM_AP_PCK_HEADER_T          tHead;
  PNM_AP_RECV_ALARM_IND_DATA_T tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_RECV_ALARM_IND_T			Type: Indication
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	Handle	Ignore
ulSrc	UINT32	Handle	Ignore
ulDestId	UINT32	any	Will contain value from ulSrcId from RCX_REGISTER_APPLICATION_REQ request packet
ulSrcId	UINT32	0	Ignore for future compatibility
ulLen	UINT32	20 to 280	Packet data length in bytes
ulId	UINT32	any	Ignore
ulSta	UINT32	0	-
ulCmd	UINT32	0x9428	PNM_AP_CMD_RECV_ALARM_IND
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure PNM_AP_RECV_ALARM_IND_DATA_T			
usDeviceHandle	UINT16	0, 1 to 128	Handle of the device causing the alarm
usSubmoduleHandle	UINT16	1 to 2048, 0xFFF0, 0xFFF1, 0xFFF2	Handle of the submodule causing the alarm
usAlarmType	UINT16		PROFINET Type of the Alarm
usAlarmPriority	UINT16	0, 1	Priority of the Alarm
ulApi	UINT32	any	Api of the submodule associated with the alarm
usSlot	UINT16	any	Slot of the submodule associated with the alarm
usSubslot	UINT16	any	Subslot of the submodule associated with the alarm
usAlarmSpecifier	UINT16		PROFINET Alarm Specifier
usAlarmDataLen	UINT16	0 to 240	Length of alarm data payload
abAlarmData[]	UINT8[]		Alarm data payload

Parameter descriptions

Parameters usDeviceHandle, usSubmoduleHandle

These parameters specify the source of the PROFINET alarm. They correspond to the handles used when configuring the PROFINET IO controller.

Parameter usAlarmType

This parameter defines the type of the alarm received. The following values are defined:

Name	Numerical value	Meaning
PN_ALARM_TYPE_PROCESS	0x0002	The submodules application issued a process alarm.
PN_ALARM_TYPE_PULL	0x0003	The submodule was pulled from the device
PN_ALARM_TYPE_PLUG	0x0004	The submodule was plugged into the device
PN_ALARM_TYPE_STATUS	0x0005	PROFINET Status Alarm was received
PN_ALARM_TYPE_UPDATE	0x0006	PROFINET Update Alarm was received
PN_ALARM_TYPE_MEDIA_REDUNDANCY	0x0007	PROFINET Media Redundancy Alarm was received
PN_ALARM_TYPE_SUPERVISOR	0x0008	The submodule was taken over by an IO Supervisor
PN_ALARM_TYPE_RELEASED	0x0009	The submodule was released by an IO Supervisor

© Hilscher, 2015–2017

Name	Numerical value	Meaning
PN_ALARM_TYPE_PLG_WRONG_SM	0x000A	A wrong submodule was plugged
PN_ALARM_TYPE_RTN_OF_SM	0x000B	The submodules input process data is now valid again
PN_ALARM_TYPE_SYSTEM_REDUNDANCY	0x0015	PROFINET System Redundancy event occured
PN_ALARM_TYPE_PULL_MODULE	0x001F	The module was pulled
-	0x0020 to 0x007F	The submodules application issued a manufacturer specific alarm
-	0x0080 to 0xFFFF	The submodules application issued a profile specific alarm

Parameter `usAlarmPriority`

This parameter is zero for low priority alarms and one for high priority alarms.

Parameters `ulApi`, `usSlot`, `usSubslot`

These parameters provide the addressing information of the submodule associated with the alarm.

Parameter `usAlarmSpecifier`

This parameter provides additional information about the alarm. The lower 15 bits contain the alarm sequence number.

Parameters `usAlarmDataLen`, `abAlarmData[]`

The parameter usAlarmDataLen is the length of received alarm data payload. The payload itself is provided in parameter abAlarmData.

© Hilscher, 2015–2017

3.5.1.2 PNM_AP_RECV_ALARM_RES_T response

Packet structure reference

```
typedef struct PNM_AP_RECV_ALARM_RES_DATA_Ttag PNM_AP_RECV_ALARM_RES_DATA_T;
struct PNM_AP_RECV_ALARM_RES_DATA_Ttag
{
  PNM_AP_DEVICEHANDLE_T     usDeviceHandle;
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle;
  uint16_t                  usAlarmType;
  uint16_t                  usAlarmPriority;
};

typedef struct PNM_AP_RECV_ALARM_RES_Ttag PNM_AP_RECV_ALARM_RES_T;
struct PNM_AP_RECV_ALARM_RES_Ttag
{
  PNM_AP_PCK_HEADER_T               tHead;
  PNM_AP_RECV_ALARM_RES_DATA_T      tData;
};
```

Packet description

Structure PNM_AP_RECV_ALARM_RES_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	8	Packet data length in bytes
ulId	UINT32	mirrored	-
ulSta	UINT32	0	Set to zero
ulCmd	UINT32	0x9429	PNM_AP_CMD_RECV_ALARM_RES
ulExt	UINT32	mirrored	-
ulRoute	UINT32	mirrored	-
Structure PNM_AP_RECV_ALARM_RES_DATA_T			
usDeviceHandle	UINT16	mirrored	Handle of the device caused the alarm
usSubmoduleHandle	UINT16	mirrored	Handle of the submodule caused the alarm
usAlarmType	UINT16	mirrored	PROFINET type of the alarm
usAlarmPriority	UINT16	mirrored	Priority of the alarm

© Hilscher, 2015–2017

Parameter descriptions

Parameters `usDeviceHandle`, `usSubmoduleHandle`

These parameters specify the source of the PROFINET alarm. They correspond to the handles used when configuring the PROFINET IO Controller and must be set to the same value as in the indication packet.

Parameter `usAlarmType`

This parameter defines the type of the alarm received. It must be set to the same value as in the indication packet.

Parameter `usAlarmPriority`

This parameter defines the alarm priority. It must be set to the same value as in the indication packet.

© Hilscher, 2015–2017

3.5.2 Receive Diagnosis service

The receive diagnosis service is used by the PROFINET Controller Firmware to indicate the reception of a PROFINET diagnosis alarm to the host application. The Indication will be used for all alarm types which use diagnosis encoding for their payload. After reception of this service and processing of the alarm data the alarm must be confirmed by the host application using *Acknowledge Alarm service* on page 134. Otherwise the associated device cannot send other alarms of the same priority.

Note: This service will not be used for alarm types configured for automatic alarm handling in *Configure IO Controller service* on page 33. These alarms will be handled internally by the PROFINET controller firmware.

Note: This service will also be used to indicate diagnosis alarms generated by the PDEV submodules of the PROFINET IO Controller itself. These alarms are virtual alarms associated with the PDEV submodules of the controller. They are generated by the firmware in order to provide a unique interface to diagnosis events regardless of their origin. The indication of this internal diagnosis alarms depends on the setting of automatic alarm handling as well. These alarms must be confirmed similar to device diagnosis alarms.

3.5.2.1 PNM_AP_RECV_DIAGNOSIS_IND_T indication

Packet structure reference

```
#define PNM_AP_MAX_ALARM_DATA_LENGTH                (1024)

typedef struct PNM_AP_CHANNEL_DIAGNOSIS_DATA_Ttag PNM_AP_CHANNEL_DIAGNOSIS_DATA_T;
struct PNM_AP_CHANNEL_DIAGNOSIS_DATA_Ttag
{
  uint16_t usChannelNumber;
  uint16_t usChannelProperties;
  uint16_t usChannelErrorType;
};

typedef struct PNM_AP_EXT_CHANNEL_DIAGNOSIS_DATA_Ttag
PNM_AP_EXT_CHANNEL_DIAGNOSIS_DATA_T;
struct PNM_AP_EXT_CHANNEL_DIAGNOSIS_DATA_Ttag
{
  uint16_t usChannelNumber;
  uint16_t usChannelProperties;
  uint16_t usChannelErrorType;
  uint16_t usExtChannelErrorType;
  uint32_t ulExtChannelAddValue;
};

typedef struct PNM_AP_QUALIFIED_CHANNEL_DIAGNOSIS_DATA_Ttag
PNM_AP_QUALIFIED_CHANNEL_DIAGNOSIS_DATA_T;
struct PNM_AP_QUALIFIED_CHANNEL_DIAGNOSIS_DATA_Ttag
{
  uint16_t usChannelNumber;
  uint16_t usChannelProperties;
  uint16_t usChannelErrorType;
  uint16_t usExtChannelErrorType;
  uint32_t ulExtChannelAddValue;
  uint16_t usQualifiedChannelQualifier;
};
```


© Hilscher, 2015–2017

```
typedef struct PNM_AP_MANUF_SPECIFIC_DIAGNOSIS_DATA_Ttag
PNM_AP_MANUF_SPECIFIC_DIAGNOSIS_DATA_T;
struct PNM_AP_MANUF_SPECIFIC_DIAGNOSIS_DATA_Ttag
{
  uint8_t abData[PNM_AP_MAX_ALARM_DATA_LENGTH];
};

#define PNM_AP_MAX_CHANNELDIAG_CNT          (PNM_AP_MAX_ALARM_DATA_LENGTH /
sizeof(PNM_AP_CHANNEL_DIAGNOSIS_DATA_T))
#define PNM_AP_MAX_EXTCHANNELDIAG_CNT       (PNM_AP_MAX_ALARM_DATA_LENGTH/
sizeof(PNM_AP_EXT_CHANNEL_DIAGNOSIS_DATA_T))
#define PNM_AP_MAX_QUALIFIEDCHANNELDIAG_CNT (PNM_AP_MAX_ALARM_DATA_LENGTH /
sizeof(PNM_AP_QUALIFIED_CHANNEL_DIAGNOSIS_DATA_T))

typedef union PNM_AP_DIAGNOSIS_DATA_Utag PNM_AP_DIAGNOSIS_DATA_U;
union PNM_AP_DIAGNOSIS_DATA_Utag
{
  PNM_AP_CHANNEL_DIAGNOSIS_DATA_T           atChannelDiag[PNM_AP_MAX_CHANNELDIAG_CNT];
  PNM_AP_EXT_CHANNEL_DIAGNOSIS_DATA_T
atExtChannelDiag[PNM_AP_MAX_EXTCHANNELDIAG_CNT];
  PNM_AP_QUALIFIED_CHANNEL_DIAGNOSIS_DATA_T
atQualifChannelDiag[PNM_AP_MAX_QUALIFIEDCHANNELDIAG_CNT];
  PNM_AP_MANUF_SPECIFIC_DIAGNOSIS_DATA_T    tManufSpecificDiag;
};

typedef struct PNM_AP_RECV_DIAGNOSIS_IND_DATA_Ttag PNM_AP_RECV_DIAGNOSIS_IND_DATA_T;
struct PNM_AP_RECV_DIAGNOSIS_IND_DATA_Ttag
{
  PNM_AP_DEVICEHANDLE_T     usDeviceHandle;
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle;
  uint16_t                  usAlarmType;
  uint16_t                  usAlarmPriority;
  uint32_t                  ulApi;
  uint16_t                  usSlot;
  uint16_t                  usSubslot;
  uint16_t                  usAlarmSpecifier;
  uint16_t                  usAlarmDataLen;
  uint16_t                  usUSI;
  PNM_AP_DIAGNOSIS_DATA_U   uData;
};

typedef struct PNM_AP_RECV_DIAGNOSIS_IND_Ttag PNM_AP_RECV_DIAGNOSIS_IND_T;
struct PNM_AP_RECV_DIAGNOSIS_IND_Ttag
{
  PNM_AP_PCK_HEADER_T              tHead;
  PNM_AP_RECV_DIAGNOSIS_IND_DATA_T tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_RECV_DIAGNOSIS_IND_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	Handle	Ignore
ulSrc	UINT32	Handle	Ignore
ulDestId	UINT32	any	Will contain value from ulSrcId from RCX_REGISTER_APPLICATION_REQ request packet
ulSrcId	UINT32	0	-
ulLen	UINT32	20 to 260	Packet data length in bytes
ulId	UINT32	any	Ignore
ulSta	UINT32	0	-
ulCmd	UINT32	0x942A	PNM_AP_CMD_RECV_DIAGNOSIS_IND
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure PNM_AP_RECV_DIAGNOSIS_IND_DATA_T			
usDeviceHandle	UINT16	0, 1 to 128	Handle of the device causing the alarm
usSubmoduleHandle	UINT16	1 to 2048, 0xFFF0, 0xFFF1, 0xFFF2	Handle of the submodule causing the alarm
usAlarmType	UINT16		PROFINET Type of the alarm
usAlarmPriority	UINT16	0, 1	Priority of the alarm
ulApi	UINT32	any	API of the submodule associated with the alarm
usSlot	UINT16	any	Slot of the submodule associated with the alarm
usSubslot	UINT16	any	Subslot of the submodule associated with the alarm
usAlarmSpecifier	UINT16		PROFINET alarm specifier
usAlarmDataLen	UINT16	0, 2 to 240	Length of alarm data payload including user structure identifier
usUSI	UINT16		User structure Identifier
uData	UNION		Alarm data payload. Valid union element depends on value of usUSI

© Hilscher, 2015–2017

Parameter descriptions

Parameters `usDeviceHandle`, `usSubmoduleHandle`

These parameters specifiy the source of the PROFINET alarm. They correspond to the handles used when configuring the PROFINET IO Controller. For the special case of a virtual diagnosis alarm, the submodule handle will indicate the controllers PDEV submodule associated with this alarm. The following parameter combinations are defined:

Device handle	Submodule handle	Meaning
1 to 128	1 to 2048	The diagnosis alarm was generated by a profinet device connected with the controller
0	0xFFF0	The diagnosis alarm was generated by the interface submodule of the profinet controller
0	0xFFF1	The diagnosis alarm was generated by the first port submodule of the profinet controller
0	0xFFF2	The diagnosis alarm was generated by the second port submodule of the profinet controller

Parameter `usAlarmType`

This parameter defines the type of the alarm received. The following values are defined:

Name	Numerical value	Meaning
`PN_ALARM_TYPE_DIAGNOSIS`	0x0001	A diagnosis was added to the submodule
`PN_ALARM_TYPE_MEDIA_REDUNDANCY`	0x0007	PROFINET Media Redundancy Alarm was received
`PN_ALARM_TYPE_DIAGNOSIS_DIS`	0x000C	A diagnosis was removed from the submodule
`PN_ALARM_TYPE_MC_COMM_MISMATCH`	0x000D	An change of multicast communication was detect by the device
`PN_ALARM_TYPE_PORT_DATA_CH_N`	0x000E	A change of PD Port diagnosis was detected by the port submodule
`PN_ALARM_TYPE_SYNC_CH_NOT`	0x000F	A change of Sync state was detected by the interface submodule
`PN_ALARM_TYPE_ISO_PROBLEM`	0x0010	A change of isochronous mode diagnosis was detected by the submodule
`PN_ALARM_TYPE_NETWORK_PRB`	0x0011	A change of network problem diagnosis was detected by the submodule
`PN_ALARM_TYPE_TIME_CH_N`	0x0012	A change of time diagnosis was detected by the submodule
`PN_ALARM_TYPE_DFP_PROBLEM`	0x0013	A change of DFP diagnosis was detected by the submodule
`PN_ALARM_TYPE_MRPD_PROBLEM`	0x0014	A change of MRPD diagnosis was detected by the submodule
`PN_ALARM_TYPE_SYSTEM_REDUNDANCY`	0x0015	PROFINET System Redundancy event occurred

Parameter `usAlarmPriority`

This parameter is zero for low priority alarms and one for high priority alarms.

Parameters `ulApi`, `usSlot`, `usSubslot`

These parameters provide the addressing information of the submodule associated with the alarm.

© Hilscher, 2015–2017

Parameter `usAlarmSpecifier`

This parameter provides additional information about the alarm. The lower 15 bits contain the alarm sequence number.

Parameters `usAlarmDataLen`, `usUSI`, `abAlarmData[]`

The parameter usAlarmDataLen is the length of received alarm data payload. For diagnosis alarms the first item in payload must be the user structure identifier which is provided in parameter usUSI. The payload itself is provided in parameter uData whereby the valid union element depends on the value of usUSI:

Name	Numerical value	Meaning
	< 0x8000	A manufacturer specific diagnosis was reported. The parameter uData.tManufSpecificDiag contains the user specific diagnosis payload data
`PN_USERSTRUCT_IDENT_CHANNEL_DIAG`	0x8000	A channel diagnosis was reported. The parameter uData.tChannelDiag contains the decoded channel diag data.
`PN_USERSTRUCT_IDENT_EXT_CHANNEL_DIAG`	0x8002	An extended channel diagnosis was reported, The parameter uData.tExtChannelDiag contains the decoded diagnosis data
`PN_USERSTRUCT_IDENT_QUALIF_CHANNEL_DIAG`	0x8003	A qualified channel diagnosis was reported. The parameter uData.tQualifChannelDiag contains the decoded diagnosis data.

Note: Even when the field uData can hold up to (PNM_AP_MAX_ALARM_DATA_LENGTH - 2) byte the maximum alarm data length is additionally limited by the supported alarm data length of the device and controller.

© Hilscher, 2015–2017

3.5.2.2 PNM_AP_RECV_DIAGNOSIS_RES_T response

Packet structure reference

```
typedef struct PNM_AP_RECV_DIAGNOSIS_RES_DATA_Ttag PNM_AP_RECV_DIAGNOSIS_RES_DATA_T;
struct PNM_AP_RECV_DIAGNOSIS_RES_DATA_Ttag
{
  PNM_AP_DEVICEHANDLE_T     usDeviceHandle;
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle;
  uint16_t                  usAlarmType;
  uint16_t                  usAlarmPriority;
};

typedef struct PNM_AP_RECV_DIAGNOSIS_RES_Ttag PNM_AP_RECV_DIAGNOSIS_RES_T;
struct PNM_AP_RECV_DIAGNOSIS_RES_Ttag
{
  PNM_AP_PCK_HEADER_T               tHead;
  PNM_AP_RECV_DIAGNOSIS_RES_DATA_T      tData;
};
```

Packet description

Structure PNM_AP_RECV_DIAGNOSIS_RES_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	8	Packet data length in bytes
ulId	UINT32	mirrored	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x942B	PNM_AP_CMD_RECV_DIAGNOSIS_RES
ulExt	UINT32	mirrored	-
ulRoute	UINT32	mirrored	-
Structure PNM_AP_RECV_DIAGNOSIS_RES_DATA_T			
usDeviceHandle	UINT16	mirrored	Handle of the device caused the alarm
usSubmoduleHandle	UINT16	mirrored	Handle of the submodule caused the alarm
usAlarmType	UINT16	mirrored	PROFINET type of the alarm
usAlarmPriority	UINT16	mirrored	Priority of the alarm

© Hilscher, 2015–2017

Parameter descriptions

Parameters `usDeviceHandle`, `usSubmoduleHandle`

These parameters specify the source of the PROFINET alarm. They correspond to the handles used when configuring the PROFINET IO Controller and must be set to the same value as in the indication packet.

Parameter `usAlarmType`

This parameter defines the type of the alarm received. It must be set to the same value as in the indication packet.

Parameter `usAlarmPriority`

This parameter defines the alarm priority. It must be set to the same value as in the indication packet.

© Hilscher, 2015–2017

3.5.3 Store Remanent service

This service is used by the PROFINET IO Controller to indicate a change of remanent data. The host application shall store the remanent data passed along with this indication in non volatile memory. On Power On, the host application shall restore this remanent data using the Load Remanent service.

Note: Handling of remanent data is mandatory to pass PROFINET IO Controller Certification tests.

As the remanent data indication packet does not fit into the DPM mailboy as a whole, a fragmented packet transfer sequence is used for this service.

3.5.3.1 PNM_AP_STORE_REMANENT_IND_T indication

Packet structure reference

```
typedef struct PNM_AP_STORE_REMANENT_IND_DATA_Ttag PNM_AP_STORE_REMANENT_IND_DATA_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_STORE_REMANENT_IND_DATA_Ttag
{
  uint16_t              usLength;
  uint8_t               abData[];
};

typedef struct PNM_AP_STORE_REMANENT_IND_Ttag PNM_AP_STORE_REMANENT_IND_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_STORE_REMANENT_IND_Ttag
{
  PNM_AP_PCK_HEADER_T              tHead;
  PNM_AP_STORE_REMANENT_IND_DATA_T tData;
};
```

Packet description

Structure PNM_AP_STORE_REMANENT_IND_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	Handle	Ignore
ulSrc	UINT32	Handle	Ignore
ulDestId	UINT32	Any	Will contain value from ulSrcId from RCX_REGISTER_APPLICATION_REQ request packet
ulSrcId	UINT32	0	-
ulLen	UINT32	8194	Packet data length in bytes
ulId	UINT32	Any	Ignore
ulSta	UINT32	0	-
ulCmd	UINT32	0x9440	PNM_AP_CMD_STORE_REMANENT_IND
ulExt	UINT32	any	Fragmented transfer handling
ulRoute	UINT32	any	Ignore
Structure PNM_AP_STORE_REMANENT_IND_DATA_T			
usLength	UINT16	8192	Length of remanent data to be stored
abData[]	UINT8[]	any	Array of remanent data to be stored

© Hilscher, 2015–2017

Parameter descriptions

Parameter usLength

Specifies the length of the remanent data

Parameter abData

Array containing the remanent data.

3.5.3.2 PNM_AP_RECV_DIAGNOSIS_RES_T response

Packet structure reference

```
typedef PNM_AP_EMTPY_PCK_T PNM_AP_STORE_REMANENT_RES_T;
```

Packet description

Structure PNM_AP_STORE_REMANENT_RES_T			Type: Response
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	mirrored	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x9440B	PNM_AP_CMD_STORE_REMANENT_RES
ulExt	UINT32	mirrored	-
ulRoute	UINT32	mirrored	-

© Hilscher, 2015–2017

3.6 Get configuration services

This section describes the services for getting information about the current configuration of Controller.

Name of service	Page	Service affects
Services informing about the current configuration settings of the PROFINET IO Controller		
Get IO Controller service	191	Controller
Get IO Controller parameter list service	199	Controller
Get IO Controller parameter service	201	Controller
Services informing about the number of configured ARs, IOCRs and submodules.		
Get number of configured objects service	206	-
Services informing in detail about the current configuration settings of the PROFINET IO Device. This affects the ARs, IOCRs, submodules and records associated with the PROFINET IO Device.		
Get IO Device service	209	Device
Get AR parameter list service	216	Device
Get AR parameter service	218	Device
Get IOCR service	223	IOCR/ Submodule
Get submodule service	231	Submodule
Get record service	238	-

Table 16: Get configuration services

There are no sections directly dealing with modules, but there is a way to retrieve the module a given submodule belongs to described in section *Get submodule service* on page 231.

In order to access data, you first have to obtain offsets (namely the base offset and relative offsets):

1. The base offset `usDpmOffset` can be obtained from the Get IOCR service described in section *Get IOCR service* (page 223)
2. The following relative offsets can be obtained from the Get submodule service described in section *Get submodule service* (page 231):
   - usFrameOffsetInput
   - usFrameOffsetOutput
   - usIOCSFrameOffsetInput
   - usIOCSFrameOffsetOutput

© Hilscher, 2015–2017

3.6.1 Get IO Controller service

This service returns the basic parameters of IO Controller itself.

3.6.1.1 PNM_AP_CFG_GET_IOC_REQ_T request

The request packet has one parameter `ulStructVersion`.

Packet structure reference

```
/*****************************************************************************
 GET IO CONTROLLER PARAMETERS
*****************************************************************************/

/*
PNM_AP_CMD_CFG_GET_IOC_DATA_REQ
    return basic parameters of IO Controller itself
*/

/** request packet */
typedef struct PNM_AP_CFG_GET_IOC_REQ_DATA_Ttag PNM_AP_CFG_GET_IOC_REQ_DATA_T;
struct PNM_AP_CFG_GET_IOC_REQ_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t              ulStructVersion;
};

typedef struct PNM_AP_CFG_GET_IOC_REQ_Ttag PNM_AP_CFG_GET_IOC_REQ_T;

struct PNM_AP_CFG_GET_IOC_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T           tHead;
  PNM_AP_CFG_GET_IOC_REQ_DATA_T tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_CFG_GET_IOC_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	Handle	
ulSrc	UINT32	Handle	
ulDestId	UINT32	Any	Set to zero for future compatibility
ulSrcId	UINT32	0	-
ulLen	UINT32	4	Packet Data Length in bytes
ulId	UINT32	Any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x00009454	PNM_AP_CMD_CFG_GET_IOC_REQ
ulExt	UINT32	any	
ulRoute	UINT32	any	
Structure PNM_AP_CFG_GET_IOC_REQ_DATA_T			
ulStructVersion	UINT32	1	Structure version

Table 17: PNM_AP_CFG_GET_IOC_REQ_T request

Parameter descriptions

Parameter ulStructVersion

Specifies the structure version.

© Hilscher, 2015–2017

3.6.1.2 PNM_AP_CFG_GET_IOC_CNF_T confirmation

Packet structure reference

```
/*****************************************************************************
 GET IO CONTROLLER PARAMETERS
*****************************************************************************/
typedef struct PNM_AP_CFG_IOC_DATA_Ttag PNM_AP_CFG_IOC_DATA_T;
struct PNM_AP_CFG_IOC_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t              ulStructVersion;
  /** The definition of the ulSystemFlags are \ref PNM_AP_CFG_STARTMODE_FLAGS_Etag */
  uint32_t              ulSystemFlags;
  /** Automatic Alarm handling flags

      With the ulAlarmHandlingFlags you can define which alarms should be handled by the
fieldbus stack automatically.
      The definition of the ulAlarmHandlingFlags are \ref
PNM_AP_CFG_ALARM_HANDLING_FLAGS_Etag enumeration.
  */
  uint32_t              ulAlarmHandlingFlags;
  /** DPM Channel Watchdog time in ms

    \arg \c Deactivated 0
    \arg \c Min 0
    \arg \c Max 65535)
  */
  uint32_t              ulDpmWatchdogTime;
  /** VendorID to be used by IO Controller
        \note This parameter is for the OEMs to handover the own VendorID number.
        \par See Spec:
        Coding of fields related to Instance, DeviceID, VendorID
  */
  uint16_t              usVendorID;
  /** DeviceID to be used by IO Controller
        \note This parameter is for the OEMs to handover the Vendor specific DeviceID
number.
        \par See Spec:
        Coding of fields related to Instance, DeviceID, VendorID
  */
  uint16_t              usDeviceID;
  /** IP address to be used by IO Controller */
  uint32_t              ulIPAddr;
  /** subnet mask to be used by IO Controller */
  uint32_t              ulNetmask;
  /** Gateway address to be used by IO Controller */
  uint32_t              ulGateway;
  /** Device Type to be used by IO Controller, Pad With Zero:
      we don't need a length here, can be computed by searching for zeros
      \par See Spec:
       See Coding of the field DeviceType
  */
  uint8_t               abDeviceType[25];
  /** NameOfStation to be used by IO Controller, Pad With Zero:
      we don't need a length here, can be computed by searching for zeros
      \note The default name should be get from the GSDML attribute \c
DNS_CompatibleName.
      \par See Spec:
       Coding of the field NameOfStationValue
  */
  uint8_t               abNameOfStation[240];
  /** OrderId to be used by IO Controller, Pad With Zero:
      we don't need a length here, can be computed by searching for zeros
      \note  The GSDML Element \c OrderNumber.
      \par See Spec:
       Coding of the field OrderID
  */
  uint8_t               abOrderId[20];
```


© Hilscher, 2015–2017

```
  /** Start offset in DPM input area for slave status bitlists.
      Each bitlist has 128 bits (one for each IO Device) thus requires 16 byte.
      The bitlists are always in the fix order: configured, active, faulty

      The bitlist must be located outside of the process data area, that is,
      either before any process data or behind.
  */
  uint16_t              usBitlistStartOffset;
  /** Fix alignment of structure fields due to definition abDeviceType */
  uint8_t               bPadding;
  /** Start offset in DPM input area for the IRT cycle counter
   *
   * This field is available since structure version 2. Set to 0xFFFF
   * to disable cycle counter in input data area. */
  uint16_t              usIoTimingInfoOffset;
};

/** confirmation packet */
typedef struct PNM_AP_CFG_GET_IOC_CNF_Ttag PNM_AP_CFG_GET_IOC_CNF_T;
struct PNM_AP_CFG_GET_IOC_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T   tHead;
  PNM_AP_CFG_IOC_DATA_T tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_CFG_GET_IOC_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	322	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32	0	
ulCmd	UINT32	0x00009455	PNM_AP_CMD_CFG_GET_IOC_CNF
ulExt	UINT32	mirrored	
ulRoute	UINT32	mirrored	
Structure PNM_AP_CFG_IOC_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
ulSystemFlags	UINT32	0, 1, 2	System flags
ulAlarmHandlingFlags	UINT32		Automatic alarm handling flags
ulDpmWatchdogTime	UINT32	0, 20 … 65535	DPM channel watchdog time
usVendorID	UINT16		VendorID to be used by IO Controller
usDeviceID	UINT16		DeviceID to be used by IO Controller
ulIPAddr	UINT32		IP address to be used by IO Controller
ulNetmask	UINT32		Subnet mask to be used by IO Controller
ulGateway	UINT32		Gateway address to be used by IO Controller
abDeviceType[25]	UINT8[]		Device Type to be used by IO Controller
abNameOfStation[240]	UINT8[]		NameOfStation to be used by IO Controller
abOrderId[20]	UINT8[]		OrderId to be used by IO Controller
usBitlistStartOffset	UINT16		Start offset in DPM input area for slave status bit lists
bPadding	UINT8	0	Padding to fix structure field alignment. Set to zero for future compatibility.
usIoTimingInfoOffset	UINT16		Start offset in DPM input area for IO timing information (This field is available since structure version 2)

Table 18: PNM_AP_CFG_GET_IOC_CNF_T confirmation

Parameter descriptions

Parameter ulStructVersion

Version of this structure. This parameter is used for future extensions of the Configure Controller Service.

© Hilscher, 2015–2017

Parameter ulSystemFlags

The ulSystemFlags field is a bitmask of flags defined as follows:

Option	Numeric value	Meaning
PNM_AP_IOC_FLAG_STARTMODE_AUTOMATIC	0x00000000	The communication will be started automatically after the PROFINET IO Controller had been configured successfully.
PNM_AP_IOC_FLAG_STARTMODE_APPLICATION_CONTROLLED	0x00000001	The application must explicitly enable the bus to start communication.
PNM_AP_IOC_FLAG_FORCE_NAME_ASSIGNMENT	0x00000002	The controller will assign the Name Of Station even to devices which are already assigned a Name Of Station.

Parameter ulAlarmHandlingFlags

This parameter specifies which alarms shall be handled by the PROFINET IO Controller itself and which alarms shall be passed to the application for processing. The field ulAlarmHandlingFlags is defined as a bitmask where each bit corresponds to a particular alarm type. For each alarm to be handled by the PROFINET IO Controller the corresponding bit shall be set. As a special use case it is also possible to handle all manufacturer specific alarms within the PROFINET IO Controller using the bitmask PNM_AP_CFG_ALARM_HANDLING_FLAG_MANUFACTURER_ALARMS.

The following bitmasks are defined:

Flag	Value
PNM_AP_CFG_ALARM_HANDLING_FLAG_DIAGNOSIS	0x00000001
PNM_AP_CFG_ALARM_HANDLING_FLAG_PROCESS	0x00000002
PNM_AP_CFG_ALARM_HANDLING_FLAG_PULL	0x00000004
PNM_AP_CFG_ALARM_HANDLING_FLAG_PLUG	0x00000008
PNM_AP_CFG_ALARM_HANDLING_FLAG_PULL_MODULE	0x00000010
PNM_AP_CFG_ALARM_HANDLING_FLAG_PLUG_WRONG	0x00000020
PNM_AP_CFG_ALARM_HANDLING_FLAG_STATUS	0x00000040
PNM_AP_CFG_ALARM_HANDLING_FLAG_UPDATE	0x00000080
PNM_AP_CFG_ALARM_HANDLING_FLAG_MEDIA_REDUND	0x00000100
PNM_AP_CFG_ALARM_HANDLING_FLAG_CONTROLLED	0x00000200
PNM_AP_CFG_ALARM_HANDLING_FLAG_RELEASED	0x00000400
PNM_AP_CFG_ALARM_HANDLING_FLAG_RETURN_OF_SUBM	0x00000800
PNM_AP_CFG_ALARM_HANDLING_FLAG_MCR_MISMATCH	0x00001000
PNM_AP_CFG_ALARM_HANDLING_FLAG_PORT_DATA_CHANGE	0x00002000
PNM_AP_CFG_ALARM_HANDLING_FLAG_SYNC_DATA_CHANGE	0x00004000
PNM_AP_CFG_ALARM_HANDLING_FLAG_TIME_DATA_CHANGE	0x00008000
PNM_AP_CFG_ALARM_HANDLING_FLAG_ISOCHR_MODE_PROBLEM	0x00010000
PNM_AP_CFG_ALARM_HANDLING_FLAG_NETW_COMP_PROBLEM	0x00020000
PNM_AP_CFG_ALARM_HANDLING_FLAG_DFP_PROBLEM	0x00040000
PNM_AP_CFG_ALARM_HANDLING_FLAG_MRPD_PROBLEM	0x00080000
PNM_AP_CFG_ALARM_HANDLING_FLAG_MULT_INTF_MISMATCH	0x00100000
PNM_AP_CFG_ALARM_HANDLING_FLAG_UPLOAD_AND_RETRIVAL	0x00200000
PNM_AP_CFG_ALARM_HANDLING_FLAG_MANUFACTURER_ALARMS	0x80000000

Table 19: Definition of alarm handling flags

© Hilscher, 2015–2017

Parameter `ulDpmWatchdogTime`

This parameter specifies the DPM channel watchdog timeout in units of milliseconds.

- A value of 0 means DPM Channel Watchdog supervision is explicitly disabled.
- A value between 20 (Minimum) and 65535 (Maximum) means DPM Channel Watchdog supervision is prepared with the specified watchdog time. In order to activate the watchdog, the application must start the supervision using the standard DPM Watchdog Mechanism after the successful configuration of the PROFINET IO Controller.

Parameter `usVendorID`

This parameter specifies the PROFINET Vendor ID the controller shall use. The Vendor ID is a unique identification number of the controller's vendor assigned by the PNO.

Note: This parameter is for the OEMs to handover the own VendorID number.

Parameter `usDeviceID`

This parameter specifies the PROFINET Device ID the controller shall use. The device ID is a unique device id assigned by the vendor of the device.

Note: This parameter is for the OEMs to handover the Vendor specific DeviceID number.

Parameter `ulIPAddr`

IP address to be used by the IO Controller

Parameter `ulNetmask`

Subnet mask to be used by the IO Controller

Parameter `ulGateway`

Gateway address to be used by the IO Controller

Parameter `abDeviceType`

The device type is a short textual description of the device. It is used in device identification when performing network scans or when reading out identification information from a PROFINET Device or Controller. The value shall be padded with zero(es) to the full field length.

Parameter `abNameOfStation`

The name of station is the bus address of a PROFINET Device or Controller. The name of station must be unique across the communication network. According to the PROFINET specification the characters a-z, 0-9, '-' and '.' are allowed where '.' has the special meaning of a label separator.. The name must fulfill further conditions:

- it must not look like an IP address, e.g. 1.2.3.4
- a label must consist of at least one and at most 63 characters
- the name must not start with '`port-abc`' or '`port-abc-defgh`' where a,b,c,d,e,f,g,h are digits 0-9

The field shall be padded with zero(es) to full field length

© Hilscher, 2015–2017

Parameter abOrderId

The vendor's order id of the PROFINET IO Controller. The field shall be padded with zero(es) to full field length.

Parameter usBitlistStartOffset

The PROFINET IO Controller manages three bit lists within the DPM Input Area. Each bit list is 128 bits (=16 byte) wide and reflects the status when the DPM Input area was updated by the PROFINET IO Controller. The parameter `usBitlistStartOffset` specifies the offset in byte from the beginning of the DPM Input Area where these lists shall be placed. The lists are always present and cannot be deactivated. The following bit lists are defined:

Name	Offset in DPM Input Area	Description
Configured bit list	usBitListStartOffset	A bit is set to true if associated AR had been configured
Active bit list	usBitListStartOffset + 16	A bit is set to true if associated AR is communicating
Diagnosis	usBitListStartOffset + 32	A bit is set to true if associated AR is communicating and either a diagnosis is pending or a configuration difference was detected

The offset of the bit corresponding to a particular AR can be derived from the device handle as follows:

```
ByteOffset =         (usDeviceHandle - 1) / 8
BitMask    = 0x1 << ((usDeviceHandle -1 ) % 8)
```

Note: The bit list area and the input IOCR data are placed within the DPM Input Area. Thus the application must ensure that the bit lists memory area does not overlap with input IOCR data blocks.

Parameter usIoTimingInfoOffset

The PROFINET IO Controller firmware provides timing information about process data handling using the PNM_AP_IOTIMINGINFO_T structure. This information can be placed in DPM input area if required. This parameter configures the start offset of this data within the DPM input area. If the offset is larger than the size of the DPM input area or the data does not fully fit into the DPM input area the feature is disabled. Thus using a value 0xFFFF will disable this feature.

Note: This information is also provided in the DPM extended status block

© Hilscher, 2015–2017

3.6.2 Get IO Controller parameter list service

This service returns a list of all "ParameterTypes" that have been configured at the IO Controller.

3.6.2.1 PNM_AP_CFG_GET_IOC_PRM_LIST_REQ_T request

The request packet does not have any parameters.

Packet structure reference

```
/*****************************************************************************
 GET IO PRM CONTROLLER PARAMETERS
*****************************************************************************/

/*
PNM_AP_CMD_CFG_GET_IOC_PRM_LIST_REQ
    return all "ParameterTypes" that have been configured for IO Controller
*/

/** request packet */
typedef PNM_AP_EMTPY_PCK_T PNM_AP_CFG_GET_IOC_PRM_LIST_REQ_T;
```

Packet description

Structure PNM_AP_CFG_GET_IOC_PRM_LIST_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	Handle	
ulSrc	UINT32	Handle	
ulDestId	UINT32	Any	Set to zero for future compatibility
ulSrcId	UINT32	0	-
ulLen	UINT32	0	Packet Data Length in bytes
ulId	UINT32	Any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x00009456	PNM_AP_CMD_CFG_GET_IOC_PRM_LIST_REQ
ulExt	UINT32	any	
ulRoute	UINT32	any	

Table 20: PNM_AP_CFG_GET_IOC_PRM_LIST_REQ_T request

© Hilscher, 2015–2017

3.6.2.2 PNM_AP_CFG_GET_IOC_PRM_LIST_CNF_T confirmation

Packet structure reference

```
/*****************************************************************************
 GET IO PRM CONTROLLER PARAMETERS
*****************************************************************************/
#define PNM_AP_MAX_IOC_PRM_TYPES 4

typedef struct PNM_AP_CFG_GET_IOC_PRM_LIST_CNF_DATA_Ttag
PNM_AP_CFG_GET_IOC_PRM_LIST_CNF_DATA_T;

struct PNM_AP_CFG_GET_IOC_PRM_LIST_CNF_DATA_Ttag
{
    uint16_t uiPrmTypes[PNM_AP_MAX_IOC_PRM_TYPES];
    uint16_t uiPrmTypesSize;
};

/** confirmation packet */
typedef struct PNM_AP_CFG_GET_IOC_PRM_LIST_CNF_Ttag PNM_AP_CFG_GET_IOC_PRM_LIST_CNF_T;
struct PNM_AP_CFG_GET_IOC_PRM_LIST_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T                    tHead;
  PNM_AP_CFG_GET_IOC_PRM_LIST_CNF_DATA_T tData;
};
```

Packet description

Structure PNM_AP_CFG_GET_IOC_PRM_LIST_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	10	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32	0	
ulCmd	UINT32	0x00009457	PNM_AP_CMD_CFG_GET_IOC_PRM_LIST_CNF
ulExt	UINT32	mirrored	
ulRoute	UINT32	mirrored	
Structure PNM_AP_CFG_GET_IOC_PRM_LIST_CNF_DATA_T			
uiPrmTypes[4]	UINT16[]	Array	Type of parameter data, see *Table 10: Parameter usPrmType*
uiPrmTypesSize	UINT16	0…4	Size of the parameter data structure

Table 21. PNM_AP_CFG_GET_IOC_PRM_LIST_CNF_T confirmation

Parameter descriptions

Parameter uiPrmTypes

This parameter specifies an array containing type information about the parameter data

Parameter uiPrmTypesSize

This parameter specifies the size of the parameter data structure.

© Hilscher, 2015–2017

3.6.3 Get IO Controller parameter service

This service returns the requested configuration for "ParameterType" and "Port Id".

3.6.3.1 PNM_AP_CFG_GET_IOC_PRM_REQ_T request

Packet structure reference

```
typedef struct PNM_AP_CFG_GET_IOC_PRM_REQ_DATA_Ttag PNM_AP_CFG_GET_IOC_PRM_REQ_DATA_T;
struct PNM_AP_CFG_GET_IOC_PRM_REQ_DATA_Ttag
{
  uint16_t uiPrmType;
  uint8_t  bPortId;
};

/** request packet */
typedef struct PNM_AP_CFG_GET_IOC_PRM_REQ_Ttag PNM_AP_CFG_GET_IOC_PRM_REQ_T;
struct PNM_AP_CFG_GET_IOC_PRM_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T               tHead;
  PNM_AP_CFG_GET_IOC_PRM_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_CFG_GET_IOC_PRM_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	Handle	
ulSrc	UINT32	Handle	
ulDestId	UINT32	Any	Set to zero for future compatibility
ulSrcId	UINT32	0	-
ulLen	UINT32	3	Packet Data Length in bytes
ulId	UINT32	Any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x00009458	PNM_AP_CMD_CFG_GET_IOC_PRM_REQ
ulExt	UINT32	any	
ulRoute	UINT32	any	
Structure PNM_AP_CFG_GET_IOC_PRM_REQ_DATA_T			
uiPrmType	UINT16	1-8, 12, 13	Parameter type
bPortId	UINT8	0-2	Port ID. The values have the following meaning: 0 = Interface, 1 = Port 0, 2 = Port 1

Table 22: PNM_AP_CFG_GET_IOC_PRM_REQ_T request

Parameter descriptions

Parameter uiPrmType

This parameter specifies the parameter type, see *Table 10: Parameter usPrmType*.

Parameter bPortId

Some records are related to a specific port of the controller. This parameter specifies the port ID of that port.

© Hilscher, 2015–2017

3.6.3.2 PNM_AP_CFG_GET_IOC_PRM_CNF_T confirmation

Packet structure reference

```
typedef struct PNM_AP_CFG_IOC_PRM_DATA_Ttag PNM_AP_CFG_IOC_PRM_DATA_T;

/* parameter union */
typedef union PNM_AP_CFG_IOC_PRM_UNION_Ttag PNM_AP_CFG_IOC_PRM_UNION_T;
union PNM_AP_CFG_IOC_PRM_UNION_Ttag
{
  PNM_AP_CFG_PRM_PDINTERFACEADJUST_T         tIntfAdjust;
  PNM_AP_CFG_PRM_SYNC_T                      tSyncData;
  PNM_AP_CFG_PRM_IRDATA_GLOBAL_T             tIrtGlobal;
  PNM_AP_CFG_PRM_IRDATA_PHASES_T             tIrtPhases;
  PNM_AP_CFG_PRM_IRDATA_FRAME_T              tIrtFrames;
  PNM_AP_CFG_PRM_PDINTFMRPDATACHECK_T        tIntfMRPDataCheck;
  PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST_T       tIntfMRPDataAdjust;
  PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST_T       tPortMRPDataAdjust;
  PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_T         tPortDataAdjust;
  PNM_AP_CFG_PRM_PDPORTDATA_CHECK_T          tPortDataCheck;
  PNM_AP_CFG_PRM_PDNCDATACHECK_T             tNCDataCheck;
  PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_T tIsochronousControllerData;
};

struct PNM_AP_CFG_IOC_PRM_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t                  ulStructVersion;

  /** identifier see \ref PNM_AP_CFG_PRM_TYPE_E */
  uint16_t                  usPrmType;
  /** Port specifier.
   *
   * @details Some records are related to a specific port of the controller.
   * Allowed values: 0 = Interface, 1 = Port 0, 2 = Port 1 */
  uint8_t                    bPortId;
  /** Reserved (Padding) */
  uint8_t                    bPadding;
  /** parameter data */
  PNM_AP_CFG_IOC_PRM_UNION_T  uData;
} ;

/** confirmation packet */
typedef struct PNM_AP_CFG_GET_IOC_PRM_CNF_Ttag PNM_AP_CFG_GET_IOC_PRM_CNF_T;
struct PNM_AP_CFG_GET_IOC_PRM_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T       tHead;
  PNM_AP_CFG_IOC_PRM_DATA_T tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_CFG_GET_IOC_PRM_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	See *Table 24*	Packet data length in bytes. Depends on contents of usPrmType
ulId	UINT32	mirrored	
ulSta	UINT32	0	
ulCmd	UINT32	0x00009459	PNM_AP_CMD_CFG_GET_IOC_PRM_CNF
ulExt	UINT32	mirrored	
ulRoute	UINT32	mirrored	
Structure PNM_AP_CFG_IOC_PRM_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usPrmType	UINT16	1 to 8, 12, 13	Parameter type (identifier)
bPortId	UINT8	0 to 2	Reference of controller port to apply this parameters to
bPadding	UINT8	0	Padding. Set to zero for future compatibility.
uData	PNM_AP_CFG_IOC_PRM_UNION_T		Parameter data

Table 23: PNM_AP_CFG_GET_IOC_PRM_CNF_T confirmation

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used for future extensions to this structure.

© Hilscher, 2015–2017

Parameter usPrmType, uData

The parameter usPrmType specifies the type of the parameter data within this request packet. This means in particular which field within uData is to be used.

Name	Numeric value	Union element	Packet length	Usage
PNM_AP_CFG_PRM_PDINTERFACEADJUST	13	tIntfAdjust	8 + 8	uData.tIntfAdjust contains a PD Interface Adjust Configuration to be written to an Interface Submodule. See *PNM_AP_CFG_PRM_PDINTERFACEADJUST_T structure* (page 110) for details.
PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST	8	tIntfMRPAdjust	8 + 4 + 276	uData.tIntfMRPDataAdjust contains a PD Interface MRP Data Adjust Configuration to be written to a Interface Submodule. See *PNM_AP_CFG_PRM_PDINTFMRPDATAADJUST_T structure* (page 101) for details on the structure.
PNM_AP_CFG_PRM_PDINTFMRPDATACHECK	7	tIntfMRPCheck	8 + 4 + 28	uData.tIntfMRPDataCheck contains a PD Interface MRP Data Check Configuration to be written to a Interface Submodule. See *PNM_AP_CFG_PRM_PDINTFMRPDATACHECK_T structure* (page 99) for details on the structure
PNM_AP_CFG_PRM_PDIRDATA_PDIRBEGINENDDATA	5	tIrtPhases	4 + 16 * Number of Phases	uData.tIrtPhases contains a PD IR Data Adjust BeginEnd configuration to be written to a Port submodule. See *PNM_AP_CFG_PRM_IRDATA_PHASES_T structure* (page 95) for details on the structure. Note: For structural reasons the PNM_AP_CFG_PRM_IRDATA_PHASES_T is associated with a Port Submodule. Nevertheless will the parameter write occur on the associated interface submodule (as defined by PROFINET Specification).
PNM_AP_CFG_PRM_PDIRDATA_PDIRFRAMEDATA	4	tIrtFrames	8 + 36	uData.tIrtFrames contains a PD IR Data Adjust Frame configuration to be written to an Interface submodule. See *PNM_AP_CFG_PRM_IRDATA_FRAME_T structure* (page 93) for details on the structure.
PNM_AP_CFG_PRM_PDIRDATA_PDIRGLOBALDATA	3	tIrtGlobal	8 + 36 + 16 * 2	uData.tIrtGlobal contains a PD IR Data Adjust Global configuration to be written to the interface submodule of the PROFINET Controller Firmware. See *PNM_AP_CFG_PRM_IRDATA_GLOBAL_T structure* (page 90)for details on the structure.
PNM_AP_CFG_PRM_PDNCDATACHECK	12	tNCDataCheck	8 + 20	uData.tNCDataCheck contains a PD NC Data check configuration to be written to an Interface/Port Submodule. See *PNM_AP_CFG_PRM_PDNCDATACHECK_T structure* (page 108) for details on the structure.
PNM_AP_CFG_PRM_PDPORTDATAADJUST	2	tPortDataAdjust	8 + 36	uData.tPortDataAdjust contains a PD Port Data Adjust configuration to be written to a Port submodule. See *PNM_AP_CFG_PRM_PDPORTDATA_ADJUST_T structure* (page 86) for details on the structure
PNM_AP_CFG_PRM_PDPORTDATACHECK	1	tPortDataCheck	8 + 272	uData.tPortDataCheck contains a PD Port Data Check configuration to be written to a Port submodule. See *PNM_AP_CFG_PRM_PDPORTDATA_CHECK_T structure* (page 84) for details on the structure

© Hilscher, 2015–2017

Name	Numeric value	Union element	Packet length	Usage
PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST	9	tPortMRPDataAdjust	8 + 20	uData.tPortMRPDataAdjust contains a PD Port MRP Data Adjust Configuration to be written to a Port Submodule. See *PNM_AP_CFG_PRM_PDPORTMRPDATAADJUST_T structure* (page 104) for details on the structure.
PNM_AP_CFG_PRM_PDSYNCDATA	6	tSyncData	8 + 280	uData.tSyncData contains a Sync Data for SyncID 0 Configuration to be written to an Interface Submodule. See *PNM_AP_CFG_PRM_SYNC_T structure* (page 96) for details on the structure. Due to the limitations of the PROFINET Controller Firmware this parameter must describe a sync master configuration.
PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_T	17	tIsochronousControllerData	8 + 24	tIsochronousControllerData contains a Isochronous Controller Data parameter. See PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA_T structure (page 116) for details on the structure.

Table 24: Parameter usPrmType

Parameter bPortId

This parameter defines to which PD instance the parameter shall be applied:

Numerical value	Meaning
0	Interface (submodule) of PROFINET IO Controller
1	Port 1 (submodule) of PROFINET IO Controller
2	Port 2 (submodule of PROFINET IO Controller

© Hilscher, 2015–2017

3.6.4 Get number of configured objects service

This service returns the numbers of configured

- ARs
- IOCRs
- submodules

3.6.4.1 PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_REQ_T request

The request packet does not have any parameters.

Packet structure reference

```
/*****************************************************************************
 GET NUMBER OF CONFIGURED OBJECTS
*****************************************************************************/

/** request packet */
typedef PNM_AP_EMTPY_PCK_T PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_REQ_T;
```

Packet description

Structure P PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	Handle	
ulSrc	UINT32	Handle	
ulDestId	UINT32	Any	Set to zero for future compatibility
ulSrcId	UINT32	0	-
ulLen	UINT32	0	Packet Data Length in bytes
ulId	UINT32	Any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x00009450	PNM_AP_CMD_CFG_GET_NUM_CONFIGURED_OBJECTS_REQ
ulExt	UINT32	any	
ulRoute	UINT32	any	

Table 25: PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_REQ_T request

© Hilscher, 2015–2017

3.6.4.2 PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_CNF_T confirmation

Packet structure reference

```
/*****************************************************************************
 GET NUMBER OF CONFIGURED OBJECTS
*****************************************************************************/

/** confirmation packet */
typedef struct PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_CNF_Ttag
PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_CNF_T;
struct PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T                              tHead;
  PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_REQ_DATA_T tData;
};

typedef struct PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_REQ_DATA_Ttag
PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_REQ_DATA_T;
struct PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_REQ_DATA_Ttag
{

  /** number of configured ARs */
  uint16_t usNumAR;
  /** number of configured IOCRs */
  uint16_t usNumIocr;
  /** number of configured Submodules */
  uint16_t usNumSubmodule;
};

/*****************************************************************************
```

Packet description

Structure PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	6	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32	0	
ulCmd	UINT32	0x00009451	PNM_AP_CMD_CFG_GET_NUM_CONFIGURED_OBJECTS_CNF
ulExt	UINT32	mirrored	
ulRoute	UINT32	mirrored	
Structure PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_REQ_DATA_T			
usNumAR	UINT16	1…128	Number of configured ARs
usNumIocr	UINT16	1…256	Number of configured IOCRs
usNumSubmodule	UINT16	1…2048	Number of configured Submodules

Table 26: PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_CNF_T confirmation

© Hilscher, 2015–2017

Parameter descriptions

Parameter `usNumAR`

This parameter contains the number of configured ARs

Parameter `usNumIocr`

This parameter contains the number of configured IOCRs

Parameter `usNumSubmodule`

This parameter contains the number of configured Submodules

© Hilscher, 2015–2017

3.6.5 Get IO Device service

This service returns basic parameters of an IO Device referred to by "DeviceHandle".

3.6.5.1 PNM_AP_CFG_GET_IOD_REQ_T request

Packet structure reference

```
/*****************************************************************************
 GET IO-DEVICE PARAMETERS
*****************************************************************************/
typedef uint16_t PNM_AP_DEVICEHANDLE_T;

typedef struct PNM_AP_CFG_GET_IOD_REQ_DATA_Ttag PNM_AP_CFG_GET_IOD_REQ_DATA_T;
struct PNM_AP_CFG_GET_IOD_REQ_DATA_Ttag
{
  uint32_t              ulStructVersion;
  PNM_AP_DEVICEHANDLE_T usDeviceHandle;
};

/** request packet */
typedef struct PNM_AP_CFG_GET_IOD_REQ_Ttag PNM_AP_CFG_GET_IOD_REQ_T;
struct PNM_AP_CFG_GET_IOD_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T           tHead;
  PNM_AP_CFG_GET_IOD_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_CFG_GET_IOD_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	Handle	
ulSrc	UINT32	Handle	
ulDestId	UINT32	Any	Set to zero for future compatibility
ulSrcId	UINT32	0	-
ulLen	UINT32	6	Packet Data Length in bytes
ulId	UINT32	Any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x0000945A	PNM_AP_CMD_CFG_GET_IOD_REQ
ulExt	UINT32	any	
ulRoute	UINT32	any	
Structure PNM_AP_CFG_GET_IOD_REQ_DATA_T			
ulStructVersion	UINT32	1	Structure version
usDeviceHandle	PNM_AP_DEVICEHANDLE_T	1 … 128	Unique handle of this IO-Device defined by the sender of this packet

Table 27: PNM_AP_CFG_GET_IOD_REQ_DATA_T request

Parameter descriptions

Parameter ulStructVersion

Structure version of the configuration structure. Used for future extensions.

Parameter usDeviceHandle

This parameter specifies the handle of the AR to be configured.

© Hilscher, 2015–2017

3.6.5.2 PNM_AP_CFG_GET_IOD_CNF_T confirmation

Packet structure reference

```
/*****************************************************************************
 GET IO-DEVICE PARAMETERS
*****************************************************************************/

enum PNM_AP_CFG_IOD_FLAG_Etag
{
  /** the IO-Device supports WriteMultiple */
  PNM_AP_CFG_IOD_FLAG_WRITE_MULTIPLE_SUPPORTED = 0x01,
  /** IO Controller firmware shall not check parameters inside this request */
  PNM_AP_CFG_IOD_FLAG_DISABLE_PRM_CHECK        = 0x02,
  /** The AR is initially disabled. If this flag is set, the profinet controller firmware
   * will not establish this AR on startup */
  PNM_AP_CFG_IOD_FLAG_DISABLED                 = 0x04,

  PNM_AP_CFG_IOD_FLAG_MAX,
};

/** Device address modes */
enum PNM_AP_CFG_IOD_ADDRESS_MODE_Etag
{
  /** The controller identifies the device using its name of station
   * or alias name and will assign name of station and ip if required.
   *
   * This is the default operation mode as defined in specification */
  PNM_AP_CFG_IOD_ADDRESS_MODE_DCP_IDENTIFY_ASSIGN = 0x00,
  /** The controller discovers the device using the configured name of station
   * but does not configure the ipsuite. The controller will use the ip address
   * reported by the device to establish communication. The controller will
   * not establish communication if no ip address was preconfigured in the device
   */
  PNM_AP_CFG_IOD_ADDRESS_MODE_DCP_DISCOVER        = 0x01,
  /** The controller discovers the device using its ip address by means of arp
   * no dcp is sent
   */
  PNM_AP_CFG_IOD_ADDRESS_MODE_ARP_DISCOVER        = 0x02,
};

typedef enum PNM_AP_CFG_IOD_ADDRESS_MODE_Etag PNM_AP_CFG_IOD_ADDRESS_MODE_E;

typedef enum PNM_AP_CFG_IOD_FLAG_Etag PNM_AP_CFG_IOD_FLAG_E;

/* AR Types */
enum PNM_AP_CFG_IOD_AR_TYPE_Etag
{
  /* standard AR Type for RT communication */
  PNM_AP_CFG_IOD_AR_TYPE_SINGLE           = 0x01,
  /* IO Supervisor AR (with IO-data), not supported */
  PNM_AP_CFG_IOD_AR_TYPE_SUPERVISOR       = 0x06,
  /* standard AR Type for IRT communication */
  PNM_AP_CFG_IOD_AR_TYPE_RTC3             = 0x10,
  /* SystemRedundancy AR, not supported */
  PNM_AP_CFG_IOD_AR_TYPE_SYSTEMREDUNDANCY = 0x20,
};

typedef enum PNM_AP_CFG_IOD_AR_TYPE_Etag PNM_AP_CFG_IOD_AR_TYPE_E;

/* AR Properties */
enum PNM_AP_CFG_IOD_AR_PROP_FLAGS_Etag
{
  /* a supervisor is allowed to take over submodules of this AR */
  PNM_AP_CFG_IOD_AR_PROP_FLAG_SUPERVISOR_TAKEOVER_ALLOWED     = 0x00000008,
  /** startup according to Profinet specification 2.3, if not set the legacy startup is
used */
  PNM_AP_CFG_IOD_AR_PROP_FLAG_STARTUPMODE_ADVANCED            = 0x40000000,
  /* alarm type PULL MODULE is supported */
  PNM_AP_CFG_IOD_AR_PROP_FLAG_PULL_MODULE_ALARM_ALLOWED       = 0x80000000,
```


© Hilscher, 2015–2017

```
  /* AR is an device access AR, only valid in conjunction with
   * PNM_AP_CFG_IOD_AR_TYPE_SUPERVISOR */
  PNM_AP_CFG_IOD_AR_PROP_FLAG_DEVICEACCESS                    = 0x00000100,
};

typedef enum PNM_AP_CFG_IOD_AR_PROP_FLAGS_Etag PNM_AP_CFG_IOD_AR_PROP_FLAGS_E;

typedef uint16_t PNM_AP_DEVICEHANDLE_T;

enum PNM_AP_DEVICEHANDLE_Etag
{
  /** Device handle of the controller itself */
  PNM_AP_DEVICEHANDLE_IOC        = 0x0000,

  /** First valid devicehandle to use for IO AR */
  PNM_AP_DEVICEHANDLE_IOAR_FIRST = 0x0001,
  /** Last valid devicehandle to use for IO AR */
  PNM_AP_DEVICEHANDLE_IOAR_LAST  = 0x0100,

  /** First valid devicehandle to use for DA AR */
  PNM_AP_DEVICEHANDLE_DAAR_FIRST = 0xF000,
  /** Last valid devicehandle to use for DA AR */
  PNM_AP_DEVICEHANDLE_DAAR_LAST  = 0xF000,
};


/* RTA Retries */
#define PNM_AP_CFG_IOD_RTA_RETRIES_MIN            (0x0003)  /**< minimum value allowed
for RTA retries */
#define PNM_AP_CFG_IOD_RTA_RETRIES_MAX            (0x000F)  /**< maximum value allowed
for RTA retries */

/* RTA timeout factor */
#define PNM_AP_CFG_IOC_RTA_TIMEOUT_FACT_MIN       (0x0001)
#define PNM_AP_CFG_IOC_RTA_TIMEOUT_FACT_MAX       (0x0064)

/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_IOD_STRUCT_VERSION_1    (0x0001)

/** request packet data */
typedef struct PNM_AP_CFG_IOD_REQ_DATA_Ttag PNM_AP_CFG_IOD_REQ_DATA_T;
struct PNM_AP_CFG_IOD_REQ_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t              ulStructVersion;
  /** unique handle of this IO-Device defined by the sender of this packet */
  PNM_AP_DEVICEHANDLE_T usDeviceHandle;
  /** AR Type see \ref PNM_AP_CFG_IOD_AR_TYPE_E */
  uint8_t               bARType;
  /** Address mode see \ref PNM_AP_CFG_IOD_ADDRESS_MODE_E */
  uint8_t               bAddressMode;
  /** flags see \ref PNM_AP_CFG_IOD_FLAG_E */
  uint32_t              ulFlags;
  /** AR UUID - has to be unique over all configured IO-Devices
      \par See Spec:
       Coding of the field ARUUID
  */
  PNM_UUID_T            tArUuid;
  /** AR Properties see \ref PNM_AP_CFG_IOD_AR_PROP_FLAGS_E
      \par See Spec:
       Coding of the field ARProperties
  */
  uint32_t              ulArProperties;
  /** VendorID */
  uint16_t              usVendorID;
  /** DeviceID */
  uint16_t              usDeviceID;
  /** InstanceID (GSDML parameter ObjectUUID Instance) */
  uint16_t              usInstanceID;
  /** max. alarm data length */
```


© Hilscher, 2015–2017

```
  uint16_t             usMaxAlarmDataLength;
  /* RTA Timeout Factor
     \par See Spec:
      Coding of the field RTATimeoutFactor
  */
  uint16_t             usRTATimeoutFact;
  /** RTA Retries
      \par See Spec:
       Coding of the field RTARetries
  */
  uint16_t             usRTARetries;
  /** NameOfStation of the IO Device
      zero padded: length determined by searching first zero
      \note  The GSDML Element \c OrderNumber.
      \par See Spec:
       Coding of the field NameOfStationValue
  */
  uint8_t              abNameOfStation[240];
  /** IP address */
  uint32_t             ulIPAddr;
  /** network mask */
  uint32_t             ulNetworkMask;
  /** Gateway address */
  uint32_t             ulGatewayAddr;
};

/** confirmation packet */
typedef struct PNM_AP_CFG_GET_IOD_CNF_Ttag PNM_AP_CFG_GET_IOD_CNF_T;
struct PNM_AP_CFG_GET_IOD_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T       tHead;
  PNM_AP_CFG_IOD_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_CFG_GET_IOD_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	296	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32	0	
ulCmd	UINT32	0x0000945B	PNM_AP_CMD_CFG_GET_IOD_CNF
ulExt	UINT32	mirrored	
ulRoute	UINT32	mirrored	
Structure PNM_AP_CFG_IOD_REQ_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usDeviceHandle	PNM_AP_DEVICEHANDLE_T	1 … 128	Unique handle of this IO-Device defined by the sender of this packet
bARType	UINT8	0x01, 0x10	AR Type
bAddressMode	UINT8	0 .. 1	IP address mode
ulFlags	UINT32		Flags controlling AR behavior
tArUuid	PNM_UUID_T		UUID associated with this AR
ulArProperties	UINT32		Properties of this AR.

© Hilscher, 2015–2017

usVendorID	UINT16	GSDML	PROFINET Vendor ID of the Device
usDeviceID	UINT16	GSDML	PROFINET Device ID of the Device
usInstanceID	UINT16	GSDML	PROFINET Instance ID of the Device
usMaxAlarmDataLength	UINT16	200 … 224	Maximum alarm data length
usRTATimeoutFact	UINT16	1 … 100	RTA Timeout Factor
usRTARetries	UINT16	3 … 15	RTA Retries
abNameOfStation[240]	UINT8[]		NameOfStation of the IO Device
ulIPAddr	UINT32	Valid IP address	IP address
ulNetworkMask	UINT32	Valid network mask	Network mask
ulGatewayAddr	UINT32	Valid IP address	Gateway address (i.e. IP address of gateway)

Table 28: PNM_AP_CFG_GET_IOD_CNF_T confirmation

Parameter descriptions

Parameter `ulStructVersion`

Structure version of the configuration structure. Used for future extensions.

Parameter `usDeviceHandle`

This parameter specifies the handle of the AR to be configured.

Parameter `bARType`

This parameter describes the type of the AR.

Name	Numeric value	Meaning
PNM_AP_CFG_IOD_AR_TYPE_SINGLE	0x01	AR Type for RT communication
PNM_AP_CFG_IOD_AR_TYPE_SUPERVISOR	0x06	IO Supervisor AR (with IO-data), not supported
PNM_AP_CFG_IOD_AR_TYPE_RTC3	0x10	AR Type for IRT communication
PNM_AP_CFG_IOD_AR_TYPE_SYSTEMREDUNDANCY	0x20	SystemRedundancy AR, not supported

Table 29: Parameter bARType - Type of the AR

Parameter `bAddressMode`

This parameter describes the possible modes of IP address resolution.

Name	Numeric value	Meaning
PNM_AP_CFG_IOD_ADDRESS_MODE_DCP_IDENTIFY_ASSIGN	0 (default)	In this mode, the controller discovers the device using its station name or alias name (only if topology info available) and will assign the configured IP address if required.
PNM_AP_CFG_IOD_ADDRESS_MODE_DCP_DISCOVER	1	The controller identifies the device using its station name or alias name and will use the IP address reported by the device. The IP address of device will not be reassigned in this mode. Please note that the Controller will not establish the communication if the IP address set in IO Device is set to zero or unreachable.
PNM_AP_CFG_IOD_ADDRESS_MODE_ARP_DISCOVER	2	The controller discovers the device using its ip address by means of ARP. No DCP is sent.

Table 30: Parameter bAddressMode - Modes of IP address resolution

© Hilscher, 2015–2017

Parameter ulFlags

The parameter ulFlags is a bitmask of flag bits specifying additional properties of the AR. The following table lists the defined bits

Name	Numeric value	Meaning
PNM_AP_CFG_IOD_FLAG_WRITE_MULTIPLE_SUPPORTED	0x01	Use multiple write to parameterize the device. This flag should be set if the PROFINET IO Device associated with this AR supports multiple write. (Extracted from GSDML) The PROFINET IO Controller will pack multiple parameter writes into one RPC Service to save Startup Time.
PNM_AP_CFG_IOD_FLAG_DISABLE_PRM_CHECK	0x02	The AR is initially disabled. If this flag is set, the PROFINET controller firmware will not establish this AR on start-up.
PNM_AP_CFG_IOD_FLAG_DISABLED	0x04	The AR is disabled.

Table 31: Parameter ulFlags - Flag bits specifying additional properties of the AR

Parameter tArUUID

Unique identifier of the AR. The AR UUID shall be generated by the engineering software or application. It is a unique ID of the AR and must be non-zero.

Parameter ulArProperties

The parameter ulARProperties is a bitmask describing PROFINET Properties of this AR. A combination of the following values is allowed:

Name	Numeric value	Meaning
PNM_AP_CFG_IOD_AR_PROP_FLAG_SUPERVISOR_TAKEOVER_ALLOWED	0x00000008	A supervisor is allowed to take over submodules of this AR.
PNM_AP_CFG_IOD_AR_PROP_FLAG_DEVICEACCESS	0x00000100	AR is an device access AR, only valid in conjunction with PNM_AP_CFG_IOD_AR_TYPE_SUPERVISOR.
PNM_AP_CFG_IOD_AR_PROP_FLAG_STARTUPMODE_ADVANCED	0x40000000	AR start-up is performed according to PROFINET specification 2.3 Advanced Startup Mode. (Extract from GSDML file). If the bitmask is not set, then the PROFINET V2.2 Startup Mode is used.
PNM_AP_CFG_IOD_AR_PROP_FLAG_PULL_MODULE_ALARM_ALLOWED	0x80000000	The alarm type PULL MODULE is supported by the device. (Extract from GSDML file)

Table 32: Parameter ulArProperties - Bitmask describing PROFINET Properties of the AR

Parameter usVendorID

VendorID to be used by IO Controller for addressing the IO Device. (Extract from GSDML file)

Parameter usDeviceID

DeviceID to be used by IO Controller for addressing the IO Device (Extract from GSDML file)

Parameter usInstanceID

The instance ID of the device. (Extract from GSDML file, Attribute "ObjectUUID Instance")

Parameter usMaxAlarmDataLength

The maximum length of payload data to be transported in an alarm. The minimum value is 200 byte and must be supported by any PROFINET Device. The upper limit of the PROFINET IO Controller is 224 byte.

© Hilscher, 2015–2017

Parameter `usRTATimeoutFact`

This parameter is used to specify the timeout of acyclic alarm services. When one side of the AR sends an alarm it is expected that the other side acknowledges the alarm within `usRTATimeoutFact` * 100ms. If the alarm is not acknowledged, up to `usRTARetries` will be performed. If still no acknowledge is received, the AR will be aborted. Default value is 1. For maximum compatibility values of the range 1 to 100 should be used.

Parameter `usRTARetries`

Number of times an alarm shall be retried before giving up and aborting the AR, Default value is 3

Parameter `abNameOfStation[240]`

The parameter abNameOfStation is the PROFINET Name Of Station to be used for this AR. The PROFINET IO Controller will establish the AR with a PROFINET IO Device with that NameOfStation. If multiple devices with same NameOfStation exists no connection will be made. The filed should be zero padded to full field length.

Parameter `ulIPAddr`

IP address to be assigned to the PROFINET IO Device. This IP Address Setting is to be specified by the application and will be assigned to the PROFINET IO Device by the PROFINET IO Controller on Startup.

Parameter `ulNetmask`

Network Mask to be assigned to the PROFINET IO Device. This IP Address Setting is to be specified by the application and will be assigned to the PROFINET IO Device by the PROFINET IO Controller on Startup.

Parameter `ulGatewayAddr`

Gateway Address to be assigned to the PROFINET IO Device. This IP Address Setting is to be specified by the application and will be assigned to the PROFINET IO Device by the PROFINET IO Controller on Startup.

© Hilscher, 2015–2017

3.6.6 Get AR parameter list service

This service returns all "ParameterTypes" that have been configured for IO Device referred to by the "DeviceHandle".

3.6.6.1 PNM_AP_CFG_GET_AR_PRM_LIST_REQ_T request

Packet structure reference

```
/*****************************************************************************
 GET AR PRM PARAMETERS
*****************************************************************************/

typedef struct PNM_AP_CFG_GET_AR_PRM_LIST_REQ_DATA_Ttag
PNM_AP_CFG_GET_AR_PRM_LIST_REQ_DATA_T;
struct PNM_AP_CFG_GET_AR_PRM_LIST_REQ_DATA_Ttag
{
    uint16_t uiDeviceHandle;
};

/** request packet */
typedef struct PNM_AP_CFG_GET_AR_PRM_LIST_REQ_Ttag PNM_AP_CFG_GET_AR_PRM_LIST_REQ_T;
struct PNM_AP_CFG_GET_AR_PRM_LIST_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T                   tHead;
  PNM_AP_CFG_GET_AR_PRM_LIST_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_CFG_GET_AR_PRM_LIST_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	Handle	
ulSrc	UINT32	Handle	
ulDestId	UINT32	Any	Set to zero for future compatibility
ulSrcId	UINT32	0	-
ulLen	UINT32	2	Packet Data Length in bytes
ulId	UINT32	Any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x0000945C	PNM_AP_CMD_CFG_GET_AR_PRM_LIST_REQ
ulExt	UINT32	any	
ulRoute	UINT32	any	
Structure PNM_AP_CFG_GET_AR_PRM_LIST_REQ_DATA_T			
uiDeviceHandle	UINT16	1…128	Device handle

Table 33: PNM_AP_CFG_GET_AR_PRM_LIST_REQ_T request

Parameter descriptions

Parameter uiDeviceHandle

This parameter specifies the handle of the AR to be configured.

© Hilscher, 2015–2017

3.6.6.2 PNM_AP_CFG_GET_AR_PRM_LIST_CNF_T confirmation

Packet structure reference

```
/*****************************************************************************
 GET AR PRM PARAMETERS
*****************************************************************************/

#define PNM_AP_MAX_AR_PRM 4

typedef struct PNM_AP_CFG_GET_AR_PRM_LIST_CNF_DATA_Ttag
PNM_AP_CFG_GET_AR_PRM_LIST_CNF_DATA_T;
struct PNM_AP_CFG_GET_AR_PRM_LIST_CNF_DATA_Ttag
{
    uint16_t uiPrmTypes[PNM_AP_MAX_AR_PRM];
    uint16_t uiPrmTypesSize;
};

/** confirmation packet */
typedef struct PNM_AP_CFG_GET_AR_PRM_LIST_CNF_Ttag PNM_AP_CFG_GET_AR_PRM_LIST_CNF_T;
struct PNM_AP_CFG_GET_AR_PRM_LIST_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T                   tHead;
  PNM_AP_CFG_GET_AR_PRM_LIST_CNF_DATA_T tData;
};
```

Packet description

Structure PNM_AP_CFG_GET_AR_PRM_LIST_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	10	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32	0	
ulCmd	UINT32	0x0000945D	PNM_AP_CMD_CFG_GET_AR_PRM_LIST_CNF
ulExt	UINT32	mirrored	
ulRoute	UINT32	mirrored	
Structure PNM_AP_CFG_GET_AR_PRM_LIST_CNF_DATA_T			
uiPrmTypes[4]	UINT16	Array	Type of parameter data, see *Table 10: Parameter usPrmType*
uiPrmTypesSize	UINT16		Size of the parameter data structure

Table 34: PNM_AP_CFG_GET_AR_PRM_LIST_CNF_T confirmation

Parameter descriptions

Parameter uiPrmTypes

This parameter specifies an array containing type information about the parameter data

Parameter uiPrmTypesSize

This parameter specifies the size of the parameter data structure.

© Hilscher, 2015–2017

3.6.7 Get AR parameter service

This service returns the requested configuration for "ParameterType" given in request for the AR referenced by using its "DeviceHandle".

3.6.7.1 PNM_AP_CFG_GET_AR_PRM_REQ_T request

Packet structure reference

```
typedef struct PNM_AP_CFG_GET_AR_PRM_REQ_DATA_Ttag PNM_AP_CFG_GET_AR_PRM_REQ_DATA_T;
struct PNM_AP_CFG_GET_AR_PRM_REQ_DATA_Ttag {
  uint16_t uiDeviceHandle;
  uint16_t uiPrmType;

  /* PNM_AP_CFG_AR_PRM_IDENT_ARVENDORBLOCKREQ */
  uint16_t uiVendorHandle;

  /* PNM_AP_CFG_AR_PRM_IDENT_IRDATA */
  /* Nothing */
};

/** request packet */
typedef struct PNM_AP_CFG_GET_AR_PRM_REQ_Ttag PNM_AP_CFG_GET_AR_PRM_REQ_T;
struct PNM_AP_CFG_GET_AR_PRM_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T              tHead;
  PNM_AP_CFG_GET_AR_PRM_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_CFG_GET_AR_PRM_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	Handle	
ulSrc	UINT32	Handle	
ulDestId	UINT32	Any	Set to zero for future compatibility
ulSrcId	UINT32	0	-
ulLen	UINT32	6	Packet Data Length in bytes
ulId	UINT32	Any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x0000945E	PNM_AP_CMD_CFG_GET_AR_PRM_REQ
ulExt	UINT32	any	
ulRoute	UINT32	any	
Structure PNM_AP_CFG_GET_AR_PRM_REQ_DATA_T			
uiDeviceHandle	UINT16	1 … 128	Device handle
uiPrmType	UINT16	1 to 8, 12, 13	Parameter type (identifier)
uiVendorHandle	UINT16	1 … 256	Vendor handle

Table 35: PNM_AP_CFG_GET_AR_PRM_REQ_T request

© Hilscher, 2015–2017

Parameter descriptions

Parameter `uiDeviceHandle`

This parameter specifies the handle of the AR to be configured.

Parameter `uiPrmTypes`

This parameter specifies an array containing type information about the parameter data.

Parameter `uiVendorHandle`

This parameter specifies the handle to the AR vendor block to be configured.

© Hilscher, 2015–2017

3.6.7.2 PNM_AP_CFG_GET_AR_PRM_CNF_T confirmation

Packet structure reference

```
typedef uint16_t PNM_AP_DEVICEHANDLE_T;

typedef uint16_t PNM_AP_ARVENDORBLOCKHANDLE_T;

enum PNM_AP_ARVENDORBLOCKHANDLE_Etag
{
  /** The first handle value to use for AR Vendorblocks*/
  PNM_AP_ARVENDORBLOCKHANDLE_FIRST = 0x0001,
  /** The last handle value to use for AR Vendorblocks*/
  PNM_AP_ARVENDORBLOCKHANDLE_LAST  = 0x0100,
};

/* AR Vendor Block, refer to GSDML ARVendorBlock */
/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_Ttag PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_T;
struct PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_Ttag
{
  /** structure version of this structure */
  uint32_t                     ulStructVersion;
  /** handle of this AR VendorBlock, required to read out the Response at runtime */
  PNM_AP_ARVENDORBLOCKHANDLE_T  usArVendorBlockHandle;
  /** AP Structure Identifier according to GSDML */
  uint16_t                     usAPStructureIdentifier;
  /** API according to GSDML */
  uint32_t                     ulApi;
  /** Array of the VendorBlock data, sizeof array depend on parameter usDataLen  */
  uint8_t                      abData[ARRAYS_OF_LENGTH_ZERO];
} ;

/* IR Data */
/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_AR_PRM_IRDATA_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_AR_PRM_IRDATA_Ttag PNM_AP_CFG_AR_PRM_IRDATA_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_AR_PRM_IRDATA_Ttag
{
  /** structure version of this structure */
  uint32_t              ulStructVersion;
  /* RTC3 domain UUID of this IRT IO device
     needs to match the UUID of IO Controllers Sync parameters
     needs to match the UUID of IO Devices PDSyncData parameters
     \par See Spec:
      Coding of the field IRDataUUID
  */
  PNM_UUID_T            tIrDataUuid;
};

typedef struct PNM_AP_CFG_AR_PRM_PARAMETERSERVER_BLOCK_Ttag
PNM_AP_CFG_AR_PRM_PARAMETERSERVER_BLOCK_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_CFG_AR_PRM_PARAMETERSERVER_BLOCK_Ttag
{
  /* external Parameter server ObjUUID */
  PNM_UUID_T            tPrmServerUuid;
  uint32_t              ulPrmServerProp;
  uint32_t              ulPrmServerActivityTimeoutFactor;
  /** NameOfStation, zero padded: length determined by searching
   * first zero */
  uint8_t               abPrmServerNameOfStation[240];
};
#endif

typedef union
{
  PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_T          tArVendorBlock;
```


© Hilscher, 2015–2017

```
  PNM_AP_CFG_AR_PRM_IRDATA_T                tIrData;
#ifdef PNM_SUPPORT_PARAMETER_SERVER
  PNM_AP_CFG_AR_PRM_PARAMETERSERVER_BLOCK_T  tPrmServerBlock;
#endif
} PNM_AP_CFG_AR_PRM_UNION_T;

typedef struct PNM_AP_CFG_AR_PRM_DATA_Ttag PNM_AP_CFG_AR_PRM_DATA_T;
/** request packet data */
struct PNM_AP_CFG_AR_PRM_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t               ulStructVersion;
  /** IO-Device handle the submodule belongs to */
  PNM_AP_DEVICEHANDLE_T   usDeviceHandle;
  /** identifier see \ref PNM_AP_CFG_AR_PRM_IDENTIFIER_E */
  uint16_t               usIdentifier;
  /** parameter data */
  PNM_AP_CFG_AR_PRM_UNION_T uData;
};

/** confirmation packet */
typedef struct PNM_AP_CFG_GET_AR_PRM_CNF_Ttag PNM_AP_CFG_GET_AR_PRM_CNF_T;
struct PNM_AP_CFG_GET_AR_PRM_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T      tHead;
  PNM_AP_CFG_AR_PRM_DATA_T tData;
};
```

Packet description

Structure PNM_AP_CFG_GET_AR_PRM_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	8+length(PNM_AP_CFG_AR_PRM_UNION_T)	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32	0	
ulCmd	UINT32	0x0000945F	PNM_AP_CMD_CFG_GET_AR_PRM_CNF
ulExt	UINT32	mirrored	
ulRoute	UINT32	mirrored	
Structure PNM_AP_CFG_AR_PRM_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usDeviceHandle	PNM_AP_DEVICEHANDLE_T	1 … 128	Handle of AR this AR Parameter belongs to
usIdentifier	UINT16	0 … 2	Identifier for the type
uData	PNM_AP_CFG_AR_PRM_UNION_T		Parameter data

Table 36: PNM_AP_CFG_GET_AR_PRM_CNF_T confirmation

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulStructVersion`

Version of this structure. Used for future extensions.

Parameter `usDeviceHandle`

Reference to the AR for which to configure the AR parameter for.

Parameters `usIdentifier, uData`

The values of `usIdentifier` specifies which kind of AR parameter is configured and which field of `uData` is valid

Name	Numeric value	Packet length	Meaning
`PNM_AP_CFG_AR_PRM_IDENT_ARVENDORBLOCKREQ`	0	8 + 12 + Length of ARVendorBlockReq data	`uData.tArVendorBlock` contains an AR Vendor Block Request to configure for the AR

Table 37: Parameters `usIdentifier`, `uData`

Structure `PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_T`

Variable	Type	Value / Range	Description
`ulStructVersion`	UINT32	1	Structure version of this structure
`usArVendorBlockHandle`	UINT16	1 to 256	Handle to this AR vendor block. This a configuration wide unique handle. That means different ARs must use different handles here.
`usAPStructureIdentifier`	UINT16	0 to 0x7FFF, 0x8000	AP Structure Identifier according to GSDML
`ulApi`	UINT32	any	API according to GSDML
`abData[]`	UINT8[]		Byte array containing the AR VendorBlock data. Size of this array depends on parameter `tHead.ulLen`. Allowed length of array: 0 (no data) - 512

Table 38: Structure `PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_T`

© Hilscher, 2015–2017

3.6.8 Get IOCR service

This service returns the current configuration of the requested IOCR. The IOCR is chosen by specifying the respective IOCR handle in parameter `usIocrHandle` of the request packet.

3.6.8.1 PNM_AP_CFG_GET_IOCR_REQ_T request

Packet structure reference

```
/*****************************************************************************
 GET IOCR PARAMETERS
*****************************************************************************/
typedef uint16_t PNM_AP_IOCR_HANDLE_T;

/*
PNM_AP_CMD_CFG_GET_IOCR_REQ
    return IOCR configuration of the requested IOCR
*/

typedef struct PNM_AP_CFG_GET_IOCR_REQ_DATA_Ttag PNM_AP_CFG_GET_IOCR_REQ_DATA_T;
struct PNM_AP_CFG_GET_IOCR_REQ_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t               ulStructVersion;

  PNM_AP_IOCR_HANDLE_T    usIocrHandle;
};

/** request packet */
typedef struct PNM_AP_CFG_GET_IOCR_REQ_Ttag PNM_AP_CFG_GET_IOCR_REQ_T;
struct PNM_AP_CFG_GET_IOCR_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T            tHead;
  PNM_AP_CFG_GET_IOCR_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_CFG_GET_IOCR_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	Handle	
ulSrc	UINT32	Handle	
ulDestId	UINT32	Any	Set to zero for future compatibility
ulSrcId	UINT32	0	-
ulLen	UINT32	6	Packet Data Length in bytes
ulId	UINT32	Any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x00009460	PNM_AP_CMD_CFG_GET_IOCR_REQ
ulExt	UINT32	any	
ulRoute	UINT32	any	
Structure PNM_AP_CFG_GET_IOCR_REQ_DATA_T			
ulStructVersion	UINT32	1	Structure version
usIocrHandle	PNM_AP_IOCR_HANDLE_T	0x1000...0x107F, 0x2000...0x207F	The handle of the configured IOCR to be accessed

Table 39: PNM_AP_CFG_GET_IOCR_REQ_T request

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulStructVersion`

Structure version of this structure. Used for future extensions of this structure.

Parameter `usIocrHandle`

The parameter `usIocrHandle` is a reference to the IOCR object whose configuration is to be determined. The following values are defined

Minimum value	Maximum value	Usage
0x1000	0x103F	Input IOCR for RT Class 1 / RT Class 3
0x1040	0x107F	Input IOCR for RT Class 1
0x2000	0x203F	Output IOCR for RT Class 1 / RT Class 3
0x2040	0x207F	Output IOCR for RT Class 1

© Hilscher, 2015–2017

3.6.8.2 PNM_AP_CFG_GET_IOCR_CNF_T confirmation

Packet structure reference

```
/*****************************************************************************
 GET IOCR PARAMETERS
*****************************************************************************/

typedef uint16_t PNM_AP_DEVICEHANDLE_T;
typedef uint16_t PNM_AP_IOCR_HANDLE_T;

typedef struct PNM_AP_CFG_IOCR_DATA_Ttag PNM_AP_CFG_IOCR_DATA_T;

/** request packet data */
struct PNM_AP_CFG_IOCR_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t               ulStructVersion;
  /** this IOCRs unique handle */
  PNM_AP_IOCR_HANDLE_T    usIocrHandle;
  /** IO-Device handle the IOCR belongs to */
  PNM_AP_DEVICEHANDLE_T   usDeviceHandle;

  /** Flags for future use, set to 0 */
  uint32_t               ulFlags;
  /** IOCR properties see \ref PNM_AP_CFG_IOCR_PROP_FLAG_E
      \par See Spec:
       Coding of the field IOCRProperties
  */
  uint32_t               ulIocrProp;
  /** IOCR type see \ref PNM_AP_CFG_IOCR_TYPE_E
      \par See Spec:
       Coding of the field IOCRType
  */
  uint16_t               usIocrType;
  /** IOCR multicast MAC address for future use, currently unsupported, set to 0
      \par See Spec:
       Coding of the field IOCRMulticastMACAdd
  */
  uint8_t                abMcastMACAddr[6];
  /** IOCR data length, the length of the cyclic frame represented by this IOCR
      \par See Spec:
       Coding of the field DataLength
  */
  uint16_t               usDataLen;
  /** IOCR sendclock factor - see above
      \par See Spec:
       Coding of the field SendClockFactor
  */
  uint16_t               usSendClockFact;
  /** IOCR reduction ratio - see above
      \par See Spec:
       Coding of the field ReductionRatio
  */
  uint16_t               usReductRatio;
  /** IOCR phase - see above
      \par See Spec:
       Coding of the field Phase
  */
  uint16_t               usPhase;
  /** IOCR sequence number
      \par See Spec:
       Coding of the field Sequence
  */
  uint16_t               usSequence;
  /** IOCR datahold factor - see above
      \par See Spec:
       Coding of the field DataHoldFactor
  */
```


© Hilscher, 2015–2017

```
  uint16_t               usDataHoldFact;
  /** IOCR FrameSendOffset - see above
      \par See Spec:
       Coding of the field FrameSendOffset
  */
  uint32_t               ulFrameSendOffs;

  /** base address for the whole IO data of this IOCR in IO Controllers DPM input/output
area */
  uint16_t               usDpmOffset;
};


/*
PNM_AP_CMD_CFG_GET_IOCR_REQ
    return IOCR configuration of the requested IOCR
*/

typedef struct PNM_AP_CFG_GET_IOCR_REQ_DATA_Ttag PNM_AP_CFG_GET_IOCR_REQ_DATA_T;
struct PNM_AP_CFG_GET_IOCR_REQ_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t               ulStructVersion;

  PNM_AP_IOCR_HANDLE_T    usIocrHandle;
};

/** confirmation packet */
typedef struct PNM_AP_CFG_GET_IOCR_CNF_Ttag   PNM_AP_CFG_GET_IOCR_CNF_T;
struct PNM_AP_CFG_GET_IOCR_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T    tHead;
  PNM_AP_CFG_IOCR_DATA_T tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_CFG_GET_IOCR_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	42	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32	0	
ulCmd	UINT32	0x00009461	PNM_AP_CMD_CFG_GET_IOCR_CNF
ulExt	UINT32	mirrored	
ulRoute	UINT32	mirrored	
Structure PNM_AP_CFG_IOCR_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usIocrHandle	PNM_AP_IOCR_HANDLE_T	0x1000...0x107F,0x2000...0x207F	Unique handle of the IOCR to configure
usDeviceHandle	PNM_AP_DEVICEHANDLE_T		IO-Device handle (handle of the AR) the IOCR is associated with
ulFlags	UINT32	0	Flags for future use, set to 0
ulIocrProp	UINT32	1 … 3	Properties of the IOCR
usIocrType	UINT16	1 … 3	Type of the IOCR
abMcastMACAddr[6]	UINT8[]	00:00:00:00:00:00	Multicast MAC Address to be used for this IOCR. Reserved for future usage. Set to zero for compatibility.
usDataLen	UINT16	1… 1440 (% 4 == 0)	Length of the data part of the IOCR without Frame Header and APDU Status (C_SDU Length). Must be a multiple of 4.
usSendClockFact	UINT16	8, 16, 32, 64, 128	Send Clock Factor to use for this IOCR. (Only values equal to a power of 2 are allowed.)
usReductRatio	UINT16	1 … 512	Reduction Ratio for this IOCR. (Only values equal to a power of 2 are allowed.)
usPhase	UINT16	1 … 512	Phase for this IOCR. Must be smaller or equal usReductionRatio
usSequence	UINT16	0	IOCR sequence number. Currently unused. Set to zero for future compatibility
usDataHoldFact	UINT16	1 … 7680	Data Hold Factor for this IOCR
ulFrameSendOffs	UINT32	0 … 3999999	Frame send offset for RTC Class 3 IOCRs. Set to zero for non RTC Class 3 IOCRs
usDpmOffset	UINT16	0 … 5700 (% 4 == 0)	Base address for the whole IO data of this IOCR in IO Controllers DPM input/output area. Must be 4-byte aligned.

Table 40: PNM_AP_CFG_GET_IOCR_CNF_T confirmation

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulStructVersion`

Structure version of this structure. Used for future extensions of this structure.

Parameter `usIocrHandle`

The parameter `usIocrHandle` is a reference to the IOCR object whose configuration has been requested. The following values are defined:

Minimum value	Maximum value	Usage
0x1000	0x103F	Input IOCR for RT Class 1 / RT Class 3
0x1040	0x107F	Input IOCR for RT Class 1
0x2000	0x203F	Output IOCR for RT Class 1 / RT Class 3
0x2040	0x207F	Output IOCR for RT Class 1

The PROFINET IO Controller will internally generate the frame id of the Input IOCRs and RT Class 3 Output IOCRs based on the frame id value. For details see section *Configuration of PROFINET IO Controller* on page 13.

Parameter `ulIocrProp`

The IOCR properties parameter `ulIocrProp` is a bit field describing the IOCR in more detail. The following values are currently defined

Symbolic name	Numeric value	Usage
`PNM_AP_CFG_IOCR_PROP_RT_CLASS_1_LEGACY`	0x01	RT_CLASS_1
`PNM_AP_CFG_IOCR_PROP_RT_CLASS_1`	0x02	RT_CLASS_1
`PNM_AP_CFG_IOCR_PROP_RT_CLASS_3`	0x03	RT_CLASS_3 (IRT)

Parameter `usIocrType`

This parameter determines the type of the IOCR. Due to the internal firmware structure the type is determined by the IOCR's handle value.

Symbolic name	Numeric value	Usage
`PNM_AP_CFG_IOCR_TYPE_INPUT`	1	For Input IOCR with 0x1000 <= usIocrHandle <= 0x107F
`PNM_AP_CFG_IOCR_TYPE_OUTPUT`	2	For Output IOCR with 0x2000 <= usIocrHandle <= 0x207F

Parameter `abMcastMacAddr`

This parameter is reserved for future usage. Set to zero.

Parameter `usDataLen`

The length of data part of the IOCR is defined by this parameter. It determines how many byte the PROFINET IO Controller firmware will copy to / from the DPM Input / Output Area for this IOCR. Internally, the data exchange is restricted to 4 byte- aligned operations, thus the parameter usDataLen must be a multiple of 4.

The PROFINET IO Controller will internally pad the IOCR data block (CSDU) to the minimum size of 40 byte if a value smaller than 40 byte is specified. This does not affect the used memory space in DPM area. E.g. if a usDataLen = 20 byte is specified, a block of 20 byte will be used in DPM and the IOCR C_SDU length will be 40.

© Hilscher, 2015–2017

When computing the length of an IOCR, the process data, provider states and consumer states must be considered.

Parameter usSendClockFact, usReductRatio and usPhase

These parameters are used to configure the cycle time of the IOCR and thus the associated AR and device. PROFINET uses the concept of a Macro and Micro Cycle to provide some kind of flexibility for Process Data Timing. The sendclock factor determines the base clock for the IOCR (The Micro Cycle). The Send Cycle interval is calculated as follows:

```
usSendClockFact * 31.25 µs
```

All RT Class 3 IOCRs must use the same send clock factor. This micro cycle is then multiplied by the reduction ratio usReductRatio to define the Data Cycle (Macro Cycle). The reduction ratio may be different for different IOCRs. The data cycles defines the process data exchange interval and is calculated as follows

```
usReductRatio * usSendclockFact * 31.25 µs
```

A network telegram will be issued for an IOCR every data cycle. Finally the parameter usPhase configures in which Send Cycle of a Data Cycle the IOCR shall be transmitted. This is especially useful in case of RT Class 3 IOCR as the Data Cycles of all RT Class 3 devices are synchronized. It is thus possible to distribute the network load more homogenous.

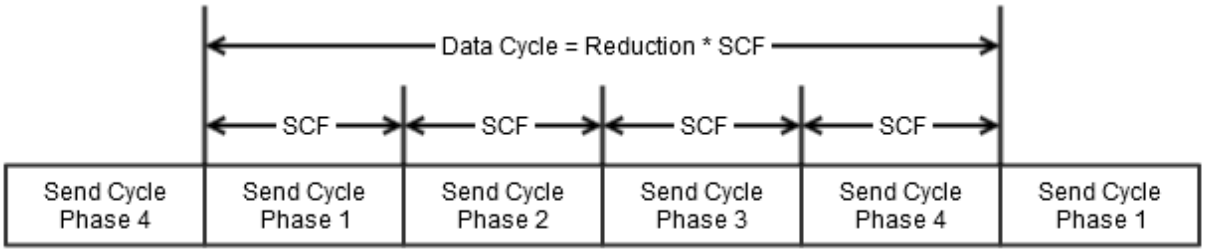

Figure 13: Definition of Send Clock Factor, Reduction Ration and Phase

The following limits apply to this parameters:

- `8 <= usSendClockFact <= 128` (allowed values for IO Controller see Technical data, for IO Devices see their GSDML)
- `usSendClockFact` of all RT Class 3 IOCRs must be identical
- `1 <= usReduction <= 512` for RT Class 1 IOCRs (allowed values for IO Devices see their GSDML)
- `1 <= usReduction <= 16` for RT Class 3 IOCRs (allowed values for IO Devices see their GSDML)
- `1 <= usPhase <= usReduction` for one IOCR

Parameter usSequence

Currently not used. Set to zero for future compatibility.

© Hilscher, 2015–2017

Parameter `usDataHoldFact`

This parameter defines the timeout for the IOCR watchdog. If no IOCR data could be exchanged for usDataHoldFact Data Cycle repetitions, the AR will be aborted with an timeout error. The Data Hold Timeout time can be calculated as follows:

```
Data Hold Timeout =
(usDataHoldFact + 1) * usReductRatio * usSendclockFact * 31.25
[microseconds]
```

The maximum allowed value for the Data Hold Timeout is 1,92 seconds.

It is strongly recommended to set this timeout to a value greater or equal than 3. Lower values might cause problems with occasional network transmission failures due to external electromagnetic noise.

Parameter `ulFrameSendOffs`

This parameter defines the send offset of the IOCR frame in respect to send clock cycle start. It is used in RT Class 3 mode. The parameter must match the corresponding value in the IR Data Sets of the associated device. In RT Class 1 mode set this parameter to 0.

Parameter `usDpmOffset`

This parameter defines where the PROFINET IO Controller will take the data for an Output IOCR from the DPM Output Area or to put the data of an Input IOCR into the DPM Input Area. The value must be 4-byte aligned due to the internal implementation oft the PROFINET IO Controller. The following image shows the relationships between the parameter usDPMOffset, usDataLen and the DPM Area. The sum of `usDPMOffset` and `usDataLen` shall not exceed the length of DPM Area (5700 byte for Output Image, 5652 byte for Input Image).

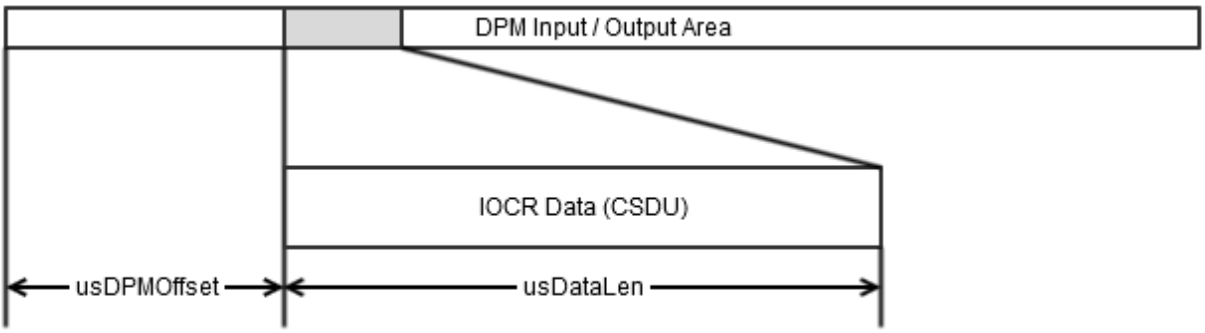

Figure 14: Memory mapping of IOCR data (CSDU) to DPM input/output area

© Hilscher, 2015–2017

3.6.9 Get submodule service

This service returns the submodule parameters of the requested submodule.

3.6.9.1 PNM_AP_CFG_GET_SUBMODULE_REQ_T request

Packet structure reference

```
/*****************************************************************************
 GET SUBMODULES PARAMETERS
*****************************************************************************/

typedef uint16_t PNM_AP_SUBMODULE_HANDLE_T;

/*
PNM_AP_CMD_CFG_GET_SUBMODULE_REQ
    return submodule parameters of the requested submodule
*/

typedef struct PNM_AP_CFG_GET_SUBMODULE_REQ_DATA_Ttag
PNM_AP_CFG_GET_SUBMODULE_REQ_DATA_T;
struct PNM_AP_CFG_GET_SUBMODULE_REQ_DATA_Ttag
{
  /** requested structure version for confirmation */
  uint32_t                  ulStructVersion;
  /** handle of submodule to retrieve */
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle;
};

/** request packet */
typedef struct PNM_AP_CFG_GET_SUBMODULE_REQ_Ttag PNM_AP_CFG_GET_SUBMODULE_REQ_T;
struct PNM_AP_CFG_GET_SUBMODULE_REQ_Ttag
{
  PNM_AP_PCK_HEADER_T                tHead;
  PNM_AP_CFG_GET_SUBMODULE_REQ_DATA_T tData;
};
```

Packet description

Structure PNM_AP_CFG_GET_SUBMODULE_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	Handle	
ulSrc	UINT32	Handle	
ulDestId	UINT32	Any	Set to zero for future compatibility
ulSrcId	UINT32	0	-
ulLen	UINT32	6	Packet Data Length in bytes
ulId	UINT32	Any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x00009462	PNM_AP_CMD_CFG_GET_SUBMODULE_REQ
ulExt	UINT32	any	
ulRoute	UINT32	any	
Structure PNM_AP_CFG_GET_SUBMODULE_REQ_DATA_T			
ulStructVersion	UINT32	1	Structure version
usSubmoduleHandle	PNM_AP_SUBMODULE_HANDLE_T		Handle of submodule to retrieve

Table 41: PNM_AP_CFG_GET_SUBMODULE_REQ_T request

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulStructVersion`

Structure version of the configuration structure. Used for future extensions.

Parameter `usSubmoduleHandle`

Handle for accessing the requested submodule.

© Hilscher, 2015–2017

3.6.9.2 PNM_AP_CFG_GET_SUBMODULE_CNF_T confirmation

Packet structure reference

```
/*****************************************************************************
 GET SUBMODULES PARAMETERS
*****************************************************************************/

/*
PNM_AP_CMD_CFG_GET_SUBMODULE_REQ
    return submodule parameters of the requested submodule
*/

typedef uint16_t PNM_AP_SUBMODULE_HANDLE_T;
typedef uint16_t PNM_AP_DEVICEHANDLE_T;
typedef uint16_t PNM_AP_IOCR_HANDLE_T;

/** Submodule Properties Bitmask */
typedef enum
{
  PNM_AP_CFG_SUBMODULE_PROPERTIES_INPUT                       = 0x0001, /* Set If Input
Data Available */
  PNM_AP_CFG_SUBMODULE_PROPERTIES_OUTPUT                      = 0x0002, /* Set If Output
Daata Available*/

  PNM_AP_CFG_SUBMODULE_PROPERTIES_ACCESS_FULL                 = 0x0000, /* full access
to submodule (default)*/
  PNM_AP_CFG_SUBMODULE_PROPERTIES_ACCESS_SHARED               = 0x0004, /* Shared Input
submodule, no alarms and no parameterization */

  PNM_AP_CFG_SUBMODULE_PROPERTIES_ZERO_INPUT_DATA_LENGTH      = 0x0008, /* not
supported, for future use */
  PNM_AP_CFG_SUBMODULE_PROPERTIES_ZERO_OUTPUT_DATA_LENGTH     = 0x0010, /* not
supported, for future use */

  PNM_AP_CFG_SUBMODULE_PROPERTIES_USE_IOXS                    = 0x0000, /* IOXS enabled,
(default) */
  PNM_AP_CFG_SUBMODULE_PROPERTIES_DISCARD_IOXS                = 0x0020, /* not
supported, for future use */
} PNM_AP_CFG_SUBMODULE_PROPERTIES_E;

/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_SUBMODULE_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_SUBMODULE_DATA_Ttag PNM_AP_CFG_SUBMODULE_DATA_T;
/** request packet data */
struct PNM_AP_CFG_SUBMODULE_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t                  ulStructVersion;
  /** unique submodule handle (freely defined by the user) */
  PNM_AP_SUBMODULE_HANDLE_T usSubmoduleHandle;
  /** IO-Device handle the submodule belongs to */
  PNM_AP_DEVICEHANDLE_T     usDeviceHandle;
  /** consumer IOCR handle the submodule belongs to */
  PNM_AP_IOCR_HANDLE_T      usInputIocrHandle;
  /** provider IOCR handle the submodule belongs to */
  PNM_AP_IOCR_HANDLE_T      usOutputIocrHandle;
  /** module identifier according to GSDML */
  uint32_t                  ulModuleIdentNumber;
  /** submodule identifier according to GSDML */
  uint32_t                  ulSubmoduleIdentNumber;
  /** API the submodule belongs to */
  uint32_t                  ulApi;
  /** slot the submodule is expected in */
  uint16_t                  usSlot;
  /* subslot the submodule is expected in */
  uint16_t                  usSubslot;
  /** submodule flags see \ref PNM_AP_CFG_SUBMODULE_FLAGS_E */
  uint16_t                  usSubmoduleProperties;
```


© Hilscher, 2015–2017

```
  /** length of the data this submodule provides according to GSDML */
  uint16_t                  usDataLenInput;
  /** length of the data this submodule consumes according to GSDML */
  uint16_t                  usDataLenOutput;
  /** offset inside the frame where the input data of this submodule shall reside */
  uint16_t                  usFrameOffsetInput;
  /** offset inside the frame where the output data of this submodule shall reside */
  uint16_t                  usFrameOffsetOutput;
  /** offset inside the frame where IOCS of input data of this submodule shall reside.
      Note that this is an offset in the output IOCR */
  uint16_t                  usIOCSFrameOffsetInput;
  /** offset inside the frame where IOCS of output data of this submodule shall reside.
      Note that this is an offset in the input IOCR */
  uint16_t                  usIOCSFrameOffsetOutput;
};

/** confirmation packet */
typedef struct PNM_AP_CFG_GET_SUBMODULE_CNF_Ttag PNM_AP_CFG_GET_SUBMODULE_CNF_T;
struct PNM_AP_CFG_GET_SUBMODULE_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T         tHead;
  PNM_AP_CFG_SUBMODULE_DATA_T tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_CFG_GET_SUBMODULE_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	42	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32	0	
ulCmd	UINT32	0x00009463	PNM_AP_CMD_CFG_GET_SUBMODULE_CNF
ulExt	UINT32	mirrored	
ulRoute	UINT32	mirrored	
Structure PNM_AP_CFG_SUBMODULE_DATA_T			
ulStructVersion	UINT32	1	Structure version used for further extensions to this packet.
usSubmoduleHandle	PNM_AP_SUBMODULE_HANDLE_T	1...2048	Unique submodule handle (can freely be defined by the user)
usDeviceHandle	PNM_AP_DEVICEHANDLE_T	1...128	IO-Device handle the submodule belongs to
usInputIocrHandle	PNM_AP_IOCR_HANDLE_T	0x1000...0x107F	Consumer IOCR handle the submodule belongs to
usOutputIocrHandle	PNM_AP_IOCR_HANDLE_T	0x2000...0x207F	Provider IOCR handle the submodule belongs to
ulModuleIdentNumber	UINT32	any	Module identifier according to GSDML
ulSubmoduleIdentNumber	UINT32	any	Submodule identifier according to GSDML
ulApi	UINT32		API the submodule belongs to
usSlot	UINT16		Slot the submodule is expected in
usSubslot	UINT16		Subslot the submodule is expected in
usSubmoduleProperties	UINT16		Submodule flags
usDataLenInput	UINT16	0...1439	Length of the data this submodule provides according to GSDML
usDataLenOutput	UINT16	0...1439	Length of the data this submodule consumes according to GSDML
usFrameOffsetInput	UINT16	0...1439	Offset inside the frame where the input data of this submodule shall reside
usFrameOffsetOutput	UINT16	0...1439	Offset inside the frame where the output data of this submodule shall reside
usIOCSFrameOffsetInput	UINT16	0...1439	Offset inside the frame where IOCS of input data of this submodule shall reside. Note that this is an offset in the output IOCR
usIOCSFrameOffsetOutput	UINT16	0...1439	Offset inside the frame where IOCS of output data of this submodule shall reside. Note that this is an offset in the input IOCR

Table 42: PNM_AP_CFG_GET_SUBMODULE_CNF_T confirmation

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulStructVersion`

Structure version of the configuration structure. Used for future extensions.

Parameter `usSubmoduleHandle`

The `usSubmoduleHandle` can be freely chosen from the user within the allowed range. It is a globally unique handle associated with a submodule. Each submodule within the whole configuration must be assigned an unique handle. This handle will be used in other services to reference to the submodule. (E.g. Record Object Read/Write)

Parameter `usDeviceHandle`

The submodule will be associated with the AR referenced by `usDeviceHandle`. This is used when establishing the connection with a PROFINET IO Device in order to verify the configuration of this particular device.

Parameters `usIocrInputHandle, usIocrOutputHandle`

Each submodule must be associated with an Input IOCR which is used to transfer the submodules input process data and provider status and output process data consumer status to the PROFINET IO Controller. Likewise each submodule must be associated with an Output IOCR which is used the transfer the submodules output process data and provider status and input process data consumer status to the PROFINET IO Device. The referenced IOCRs must be associated with the same AR as referenced by parameter usDeviceHandle.

Parameters `ulModuleIdenNumber, ulSubmoduleIdentNumber`

The identification numbers are unique IDs defined by the PROFINET IO Device vendor within the GSDML file. They are used to uniquely identify the Module and Submodule

Icon

Note: The Module ID of Submodules with same `ulApi` and `ulSlot` must be identical.

Parameters `ulApi`, `usSlot`, `usSubslot`

The parameters `ulApi`, `usSlot`, `usSubslot` describe where a particular submodule has been configured within the PROFINET IO Device. These parameters depend on the actual configuration of the PROFINET IO Device. Allowed values are described in the GSDML file of the device.

Parameter `usSubmoduleProperties`

The usSubmoduleProperties is a bit field describing various properties of the submodule. The following flags are defined

Flag	Numeric value	Usage
`PNM_AP_CFG_SUBMODULE_PROPERTIES_INPUT`	0x0001	Must be set if usDataLenInput > 0 or usDataLenOutput == 0
`PNM_AP_CFG_SUBMODULE_PROPERTIES_OUTPUT`	0x0002	Must be set if usDataLenOutput > 0

Parameters `usDataLenInput`, `usDataLenOutput`

The parameters usDataLenInput and usDataLenOutput specify the lengths of the submodules input and output process data. It only contains the IO data length without IOPS. If a submodule has

© Hilscher, 2015–2017

neither input nor output data it is considered to be an input submodule with 0 byte of process data. As consequence an input provider status and input consumer status must still be allocated within the Input and Output IOCRs associated with this submodule.

Parameters `usFrameOffsetInput`, `usFrameOffsetOutput`, `usIOCSFrameOffsetInput`, `usIOCSFrameOffsetOutput`

These parameters describe the placement of the input and output process data and its associated provider and consumer states within the Input and Output IOCRs and the DPM. According to PROFINET Specification:

- usFrameOffsetInput **(1)** and usIOCSFrameOffsetOutput **(2)** must be valid for input submodules. These are submodule with input process data.
- usFrameOffsetOutput **(3)** and usIOCSFrameOffsetInput **(4)** must be valid for output submodules. These are submodules with output process data.
- usFrameOffsetInput **(1)**, usIOCSFrameOffsetOutput **(2)**, usFrameOffsetOutput **(3)** and usIOCSFrameOffsetInput **(4)** must be valid for input-output submodules. These are submodule with input and output process data.

The length of the provider and consumer data status is fixed to one byte. The mapping of the submodules data to IOCRs and DPM areas is shown in the following figure:

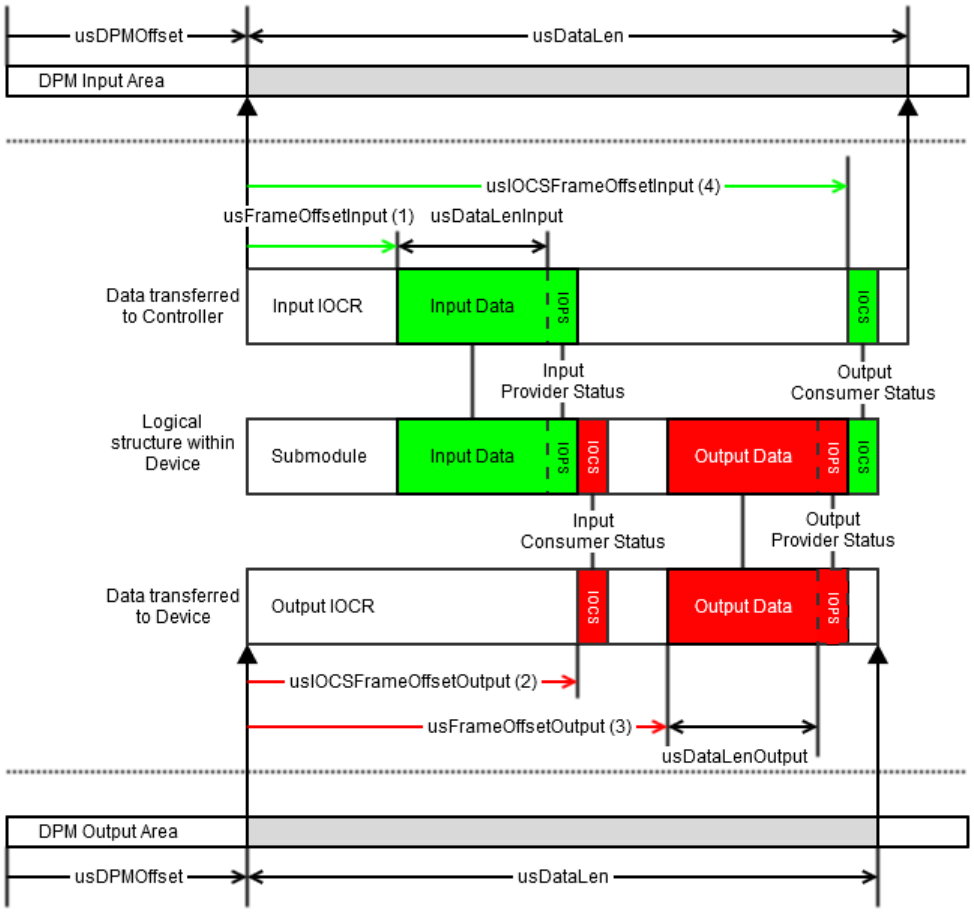

Figure 15: Mapping of submodule data to IOCRs and DPM areas

`usDPMOffset` and `usDataLen` are parameterized in Configure IOCR Service.

© Hilscher, 2015–2017

3.6.10 Get record service

This service shall be used to read out configured record parameter objects which have been written by the controller to the device when establishing the AR. This service is optional.

Note: The record parameters are internally stored within a storage associated with the AR. The storage has a limited size. Please refer to section *Technical data* on page 5 for details on the available memory amount for startup record object parameterization. Additionally to the record data, each parameter consumes some byte for management information. Details are described in *PNM_AP_CFG_RECORD_REQ_T request* (page 71).

3.6.10.1 PNM_AP_CFG_GET_RECORD_DATA_REQ_T request

The request packet does not have any parameters.

Packet structure reference

```
/*****************************************************************************
 GET RECORD PARAMETERS
*****************************************************************************/

/*
   shall return ONLY the following parameters
 * PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA
 * PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA
 * PNM_AP_CFG_PRM_RECORD (to deliver this Type in Requestor has to provide the index)

   other parameter types cannot be read by application

PNM_AP_CMD_CFG_GET_RECORD_DATA_REQ
   get records and parameters for a specific submodule
*/

#if 0
/** request packet */
typedef PNM_AP_EMTPY_PCK_T PNM_AP_CFG_GET_RECORD_DATA_REQ_T;
```

Packet description

Structure PNM_AP_CFG_GET_RECORD_DATA_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	Handle	
ulSrc	UINT32	Handle	
ulDestId	UINT32	Any	Set to zero for future compatibility
ulSrcId	UINT32	0	-
ulLen	UINT32	0	Packet Data Length in bytes
ulId	UINT32	Any	-
ulSta	UINT32	0	Status unused for request. Set to zero.
ulCmd	UINT32	0x00009464	PNM_AP_CMD_CFG_GET_RECORD_REQ
ulExt	UINT32	any	
ulRoute	UINT32	any	

Table 43: PNM_AP_CFG_GET_RECORD_DATA_REQ_T request

© Hilscher, 2015–2017

3.6.10.2 PNM_AP_CFG_GET_RECORD_DATA_CNF_T confirmation

Packet structure reference

```
/*****************************************************************************
 GET RECORD PARAMETERS
*****************************************************************************/

/* allowed values for field ulStructVersion */
#define PNM_AP_CFG_RECORD_STRUCT_VERSION_1    (0x0001)

typedef struct PNM_AP_CFG_RECORD_DATA_Ttag PNM_AP_CFG_RECORD_DATA_T;

struct PNM_AP_CFG_RECORD_DATA_Ttag
{
  /** structure version of this structure */
  uint32_t               ulStructVersion;
  /* the index of the record - defined in GSDML */
  uint16_t               usIndex;
  /* the transfersequence value - defined in GSDML */
  uint16_t               usTransferSequence;
  /** Array of Record Data, length of the data is derived from packet length  */
  uint8_t                abRecordData[ARRAYS_OF_LENGTH_ZERO];
};

/*
   shall return ONLY the following parameters
 * PNM_AP_CFG_PRM_ISOCHRONOUSCONTROLLERDATA
 * PNM_AP_CFG_PRM_ISOCHRONOUSMODEDATA
 * PNM_AP_CFG_PRM_RECORD (to deliver this Type in Requestor has to provide the index)

   other parameter types cannot be read by application

PNM_AP_CMD_CFG_GET_RECORD_DATA_REQ
   get records and parameters for a specific submodule
*/

#if 0
/** confirmation packet */
typedef struct PNM_AP_CFG_GET_RECORD_DATA_CNF_Ttag PNM_AP_CFG_GET_RECORD_DATA_CNF_T;
struct PNM_AP_CFG_GET_RECORD_DATA_CNF_Ttag
{
  PNM_AP_PCK_HEADER_T      tHead;
  PNM_AP_CFG_RECORD_DATA_T tData;
};
```


© Hilscher, 2015–2017

Packet description

Structure PNM_AP_CFG_GET_RECORD_DATA_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	8 + length of record data	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32	0	
ulCmd	UINT32	0x00009465	PNM_AP_CMD_CFG_GET_RECORD_CNF
ulExt	UINT32	mirrored	
ulRoute	UINT32	mirrored	
Structure PNM_AP_CFG_RECORD_DATA_T			
ulStructVersion	UINT32	1	Structure version of this structure
usIndex	UINT16	1 … 0x7FFF	The index of the manufacturer specific record object
usTransferSequence	UINT16	any	The transfer sequence for the record object - defined in GSDML
abRecordData[ARRAYS_OF_LENGTH_ZERO]	UINT8[]	any	Array of record data

Table 44: PNM_AP_CFG_GET_RECORD_DATA_CNF_T confirmation

Parameter descriptions

Parameter ulStructVersion

Version of this structure. Used for future extensions.

Parameter usIndex

The record object index of the record object to write. This parameter is provided in the GSDML file.

Parameter usTransferSequence

This parameter can be used to configure the order in which the record object shall be written. This value is provided in the GSDML file of the PROFINET IO Device. The default value is 0.

Parameter abRecordData[]

This parameter contains the payload (the actual record data to be written). The length of the data is determined from the packet length.

© Hilscher, 2015–2017

3.7 Legacy services

The PROFINET IO Controller version 3 as LFW supports some services of the previous version 2 LFW. These services are provided for easier transition of existing applications.

Whenever possible the new API services for Read / Write record should be used:

- Read submodule record service (page 118)
- Write submodule record service (page 124)
- Read implicit record service (page 128)

3.7.1 Read Record service (legacy)

This service can be used by the application to read a record object from a submodule in data exchange.

3.7.1.1 APIOC_READ_REQ_T request

Packet structure reference

```
typedef struct
{
  TLR_UINT32      ulHandle;
  TLR_UINT32      ulApi;
  TLR_UINT32      ulDataLength;
  TLR_UINT16      usSlot;
  TLR_UINT16      usSubSlot;
  TLR_UINT16      usIndex;
} APIOC_READ_REQ_DATA_T;

typedef struct
{
  PNM_PNIOAPCTL_PCK_HEADER_T            tHead;
  APIOC_READ_REQ_DATA_T                 tData;
} APIOC_READ_REQ_T;
```


© Hilscher, 2015–2017

Packet description

Structure APIOC_READ_REQ_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Unused by Stack Set to zero for future compatibility
ulSrcId	UINT32	any	-
ulLen	UINT32	18	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x0C1A	PNIO_APCTL_CMD_READ_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	0	-
Structure APIOC_READ_REQ_DATA_T			
ulHandle	UINT32	0, 1 to 128	Handle of the device to read the record from
ulApi	UINT32	any	API of the submodule to read the record from
ulDataLength	UINT32	1 to 1024	Maximum amount of data to read from the submodule
usSlot	UINT16	0 to 0xFFFF	Slot of the submodule to read the record from
usSubSlot	UINT16	0 to 0xFFFF	Subslot of the submodule to read the record from
usIndex	UINT16	0 to 0xFFFF	Index of record object to read

Parameter descriptions

Parameter ulHandle

The handle of the (previously configured) device to read the record object from.

Parameters ulApi, usSlot, usSubslot

Addressing parameters of the submodule to read the record object from.

Parameter usIndex

The index of the record object to read.

Parameter ulDataLength

The maximum number of bytes to read from the object.

© Hilscher, 2015–2017

3.7.1.2 APIOC_READ_CNF_T confirmation

Packet structure reference

```
typedef struct
{
  TLR_UINT32      ulHandle;
  TLR_UINT32      ulPnio;
  TLR_UINT32      ulApi;
  TLR_UINT32      ulDataLength;
  TLR_UINT16      usSlot;
  TLR_UINT16      usSubSlot;
  TLR_UINT16      usIndex;
  TLR_UINT16      usAddVal1;
  TLR_UINT16      usAddVal2;
  TLR_UINT16      usAlign;
  TLR_UINT8      abReadData[1];
} APIOC_READ_CNF_DATA_T;

typedef struct
{
  PNM_PNIOAPCTL_PCK_HEADER_T            tHead;
  APIOC_READ_CNF_DATA_T                 tData;
} APIOC_READ_CNF_T;
```

Packet description

Structure APIOC_READ_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_PNIOAPCTL_PCK_HEADER_T			
ulDest	UINT32		
ulSrc	UINT32		
ulDestId	UINT32	0	Ignore for future compatibility
ulSrcId	UINT32	mirrored	
ulLen	UINT32	28 to 1052	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x0C1B	PNIO_APCTL_CMD_READ_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure APIOC_READ_CNF_DATA_T			
ulHandle	UINT32	mirrored	Handle of the device the record was read from
ulPnio	UINT32	any	The PROFINET Status Code from the read operation
ulApi	UINT32	mirrored	API of the submodule the record was read from
ulDataLength	UINT32	0 to 1024	Amount of data read from the record object
usSlot	UINT16	mirrored	Slot of the submodule the record was read from
usSubSlot	UINT16	mirrored	Subslot of the submodule the record was read from
usIndex	UINT16	mirrored	Index of record object read
usAddVal1	UINT16	any	Additional value returned by PROFINET Device when reading the record object
usAddVal2	UINT16	any	Additional value returned by the PROFINET Device when reading the record object
usAlign	UINT16	any	Alignment. Ignore for future Compatibility
abReadData[]	UINT8		The data read from the record object

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulHandle`

The handle of the (previously configured) device the record object was read from.

Parameters `ulApi`, `usSlot`, `usSubslot`

Addressing parameters of the submodule the record object was read from.

Parameter `usIndex`

The index of the record object read.

Parameter `ulDataLength`

The number of bytes read from the record object

Parameter `ulPnio`

The PROFINET status code of the acyclic operation. This field is non-zero if a PROFINET error occurred during processing the read service. The error code is either generated by the PROFINET IO Controller or by the associated PROFINET IO Device. Section *PROFINET Status Code* on page 279 describes the meanings of the PROFINET status code.

Parameters `usAddVal1`, `usAddVal2`

The PROFINET specification defines additional values to be passed from the device in read record object service responses. The content of these values can be found in this parameters.

Parameter `abReadData[]`

This parameter contains the actual record data (payload) returned by the device.

© Hilscher, 2015–2017

3.7.2 Write Record service (legacy)

This service can be used by the application to write a record object of a submodule in data exchange.

3.7.2.1 APIOC_WRITE_REQ_T request

Packet structure reference

```
typedef struct
{
  TLR_UINT32      ulHandle;
  TLR_UINT32      ulApi;
  TLR_UINT32      ulDataLength;
  TLR_UINT16      usSlot;
  TLR_UINT16      usSubSlot;
  TLR_UINT16      usIndex;
  TLR_UINT16      usAlign;
  TLR_UINT8      abWriteData[1];
} APIOC_WRITE_REQ_DATA_T;

typedef struct
{
  PNM_PNIOAPCTL_PCK_HEADER_T            tHead;
  APIOC_WRITE_REQ_DATA_T                tData;
} APIOC_WRITE_REQ_T;
```

Packet description

Structure APIOC_READ_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_PNIOAPCTL_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Unused by Stack Set to zero for future compatibility
ulSrcId	UINT32	any	-
ulLen	UINT32	21 … 1044	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x0C1C	PNIO_APCTL_CMD_WRITE_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	0	-
Structure APIOC_WRITE_REQ_DATA_T			
ulHandle	UINT32	1...128	Handle of the device to write the record to
ulApi	UINT32	any	API of the submodule to write the record to
ulDataLength	UINT32	1 ...1024	Amount of data to write to the record object
usSlot	UINT16	0...0xFFFF	Slot of the submodule to write the record to
usSubSlot	UINT16	0...0xFFFF	Subslot of the submodule to write the record to
usIndex	UINT16	0...0xFFFF	Index of record object to write
usAlign	UINT16	0	Set to zero for future compatibility
abWriteData[]	UINT8		The data to write to the record object

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulHandle`

The handle of the (previously configured) device to write the record object to

Parameters `ulApi`, `usSlot`, `usSubslot`

Addressing parameters of the submodule to write the record object to

Parameter `usIndex`

The index of the record object to write.

Parameter `ulDataLength`

The number of bytes to write to the record object.

Parameter `abWriteData[]`

The data to write to the record object.

© Hilscher, 2015–2017

3.7.2.2 APIOC_WRITE_CNF_T confirmation

Packet structure reference

```
typedef struct
{
  TLR_UINT32      ulHandle;
  TLR_UINT32      ulPnio;
  TLR_UINT32      ulApi;
  TLR_UINT16      usSlot;
  TLR_UINT16      usSubSlot;
  TLR_UINT16      usIndex;
  TLR_UINT16      usAddVal1;
  TLR_UINT16      usAddVal2;
} APIOC_WRITE_CNF_DATA_T;

typedef struct
{
  PNM_PNIOAPCTL_PCK_HEADER_T            tHead;
  APIOC_WRITE_CNF_DATA_T                tData;
} APIOC_WRITE_CNF_T;
```

Packet description

Structure APIOC_WRITE_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_PNIOAPCTL_PCK_HEADER_T			
ulDest	UINT32		
ulSrc	UINT32		
ulDestId	UINT32	0	Ignore for future compatibility
ulSrcId	UINT32	mirrored	
ulLen	UINT32	22	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x0C1D	PNIO_APCTL_CMD_WRITE_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure APIOC_WRITE_CNF_DATA_T			
ulHandle	UINT32	mirrored	Handle of the device the record data was written to
ulPnio	UINT32	any	The PROFINET Status Code from the write operation
ulApi	UINT32	mirrored	API of the submodule the record was written to
usSlot	UINT16	mirrored	Slot of the submodule the record was written to
usSubSlot	UINT16	mirrored	Subslot of the submodule the record was written to
usIndex	UINT16	mirrored	Index of record object written
usAddVal1	UINT16	any	Additional value returned by PROFINET Device when writing the record object
usAddVal2	UINT16	any	Additional value returned by the PROFINET Device when writing the record object

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulHandle`

The handle of the (previously configured) device the record data was written to.

Parameters `ulApi`, `usSlot`, `usSubslot`

Addressing parameters of the submodule the record data was written to.

Parameter `usIndex`

The index of the record object written.

Parameter `ulPnio`

The PROFINET status code of the acyclic operation. This field is non-zero if an PROFINET error occurred during processing the write service. The error code is either generated by the PROFINET IO Controller or by the associated PROFINET IO Device. Section *PROFINET Status Code* on page 279 describes the meanings of the PROFINET status code.

Parameters `usAddVal1`, `usAddVal2`

The PROFINET specification defines additional values to be passed from the device in write record object service responses. The content of these values can be found in this parameters.

© Hilscher, 2015–2017

3.7.3 Read Record Implicit service (legacy)

This service can be used by the application to implicitly read a record object of an device. This service can be used to read PROFINET record objects from any PROFINET IO Device with an assigned IP Address in the network.

Note: The service involves RPC Transactions which might timeout due to network problems or unreachable devices. Due to these timeouts the confirmation packet might be delayed by several seconds. This must be considered for the application implementation.

Note: Only one Read implicit record service may be active at the same time. The application must wait for the confirmation before the next read can be performed.

© Hilscher, 2015–2017

3.7.3.1 APIOC_READ_IMPL_REQ_T request

Packet structure reference

```
typedef struct
{
  TLR_UINT32      ulIPAddress;
  TLR_UINT16      usDeviceId;
  TLR_UINT16      usVendorId;
  TLR_UINT16      usInstanceId;
  TLR_UINT16      usReserved;
  TLR_UINT32      ulApi;
  TLR_UINT32      ulDataLength;
  TLR_UINT16      usSlot;
  TLR_UINT16      usSubSlot;
  TLR_UINT16      usIndex;
} APIOC_READ_IMPL_REQ_DATA_T;

typedef struct
{
  PNM_PNIOAPCTL_PCK_HEADER_T            tHead;
  APIOC_READ_IMPL_REQ_DATA_T            tData;
} APIOC_READ_IMPL_REQ_T;
```

Packet description

Structure APIOC_READ_IMPL_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_PNIOAPCTL_PCK_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Unused by Stack Set to zero for future compatibility
ulSrcId	UINT32	any	-
ulLen	UINT32	26	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x0C1E	PNIO_APCTL_CMD_READ_IMPL_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	0	-
Structure APIOC_READ_IMPL_REQ_DATA_T			
ulIPAddr	UINT32	any	IP Address of the device to read the record from
usDeviceId	UINT16	any	PROFINET Device Id of the device to read the record from
usVendorId	UINT16	any	PROFINET Vendor Id of the device to read the record from
usInstanceId	UINT16	any	Instance Id of the device to read the record from
usReserved	UINT16	0	Padding. Set to zero for future compatibility.
ulApi	UINT32	any	API of the submodule to read the record from
ulDataLength	UINT32	1 to 1024	Maximum amount of data to read from the submodule
usSlot	UINT16	0 to 0xFFFF	Slot of the submodule to read the record from
usSubSlot	UINT16	0 to 0xFFFF	Subslot of the submodule to read the record from
usIndex	UINT16	0 to 0xFFFF	Index of record object to read

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulIPAddr`

The IP Address of the device to read the record object from.

Parameters `usVendorId`, `usDeviceId`, `usInstanceId`

PROFINET Device identification according GSDML file of the device to read the record object from.

Parameters `ulApi`, `usSlot`, `usSubslot`

The addressing parameters of the submodule to read the record object from.

Parameter `usIndex`

The index of the record object to read.

Parameter `ulDataLength`

The maximum number of bytes to read from the object.

© Hilscher, 2015–2017

3.7.3.2 APIOC_READ_IMPL_CNF_T confirmation

Packet structure reference

```
typedef struct
{
  TLR_UINT32      ulPnio;
  TLR_UINT32      ulApi;
  TLR_UINT32      ulDataLength;
  TLR_UINT16      usSlot;
  TLR_UINT16      usSubSlot;
  TLR_UINT16      usIndex;
  TLR_UINT16      usAddVal1;
  TLR_UINT16      usAddVal2;
  TLR_UINT16      usAlign;
  TLR_UINT8      abReadData[1];
} APIOC_READ_IMPL_CNF_DATA_T;

typedef struct
{
  PNM_PNIOAPCTL_PCK_HEADER_T             tHead;
  APIOC_READ_IMPL_CNF_DATA_T             tData;
} APIOC_READ_IMPL_CNF_T;
```

Packet description

Structure APIOC_READ_IMPL_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_PNIOAPCTL_PCK_HEADER_T			
ulDest	UINT32		
ulSrc	UINT32		
ulDestId	UINT32	0	Ignore for future compatibility
ulSrcId	UINT32	mirrored	
ulLen	UINT32	24 … 1048	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x0C1F	PNIO_APCTL_CMD_READ_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure APIOC_READ_IMPL_CNF_DATA_T			
ulPnio	UINT32	any	The PROFINET Status Code from the read operation
ulApi	UINT32	mirrored	API of the submodule the record was read from
ulDataLength	UINT32	0 … 1024	Amount of data read from the record object
usSlot	UINT16	mirrored	Slot of the submodule the record was read from
usSubSlot	UINT16	mirrored	Subslot of the submodule the record was read from
usIndex	UINT16	mirrored	Index of record object read
usAddVal1	UINT16	any	Additional value returned by PROFINET Device when reading the record object
usAddVal2	UINT16	any	Additional value returned by the PROFINET Device when reading the record object
usAlign	UINT16	any	Alignment. Ignore for future Compatibility
abReadData[]	UINT8		The data read from the record object

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulSta`

Contains a non zero error code if the request was rejected by the PROFINET IO Controller. For details on error codes refer to Status/Error codes.

Parameters `ulApi`, `usSlot`, `usSubslot`

Addressing parameters of the submodule the record object was read from.

Parameter `usIndex`

The index of the record object read.

Parameter `ulDataLength`

The number of bytes read from the record object

Parameter `ulPnio`

The PROFINET status code of the acyclic operation. This field is non-zero if an PROFINET error occurred during processing the read service. The error code is either generated by the PROFINET IO Controller or by the associated PROFINET IO Device. Section *PROFINET Status Code* on page 279 describes the meanings of the PROFINET status code.

Parameters `usAddVal1`, `usAddVal2`

The PROFINET specification defines additional values to be passed from the device in read record object service responses. The content of these values can be found in this parameters.

Parameter `abReadData[]`

This parameter contains the actual record data returned by the device.

© Hilscher, 2015–2017

3.8 Common services

The following sections describe common services provided by the PROFINET IO Controller.

3.8.1 Get Slave Handle service

This API defines a service common across all master protocol stacks. The service is used to retrieve a list of slave handles for a specific slave state.

Note: The value of handles returned in the service confirmation are equal to the handles used when configuring the devices using the *Configure IO Device service* on page 46.

3.8.1.1 RCX_GET_SLAVE_HANDLE_REQ_T request

Packet structure reference

```
typedef __TLR_PACKED_PRE  struct RCX_PACKET_GET_SLAVE_HANDLE_REQ_DATA_Tag
{
  TLR_UINT32 ulParam;
} __TLR_PACKED_POST  RCX_PACKET_GET_SLAVE_HANDLE_REQ_DATA_T;

#define RCX_LIST_CONF_SLAVES                0x00000001
#define RCX_LIST_ACTV_SLAVES                0x00000002
#define RCX_LIST_FAULTED_SLAVES             0x00000003

typedef __TLR_PACKED_PRE struct RCX_PACKET_GET_SLAVE_HANDLE_REQ_Tag
{
  TLR_PACKET_HEADER_T                      tHead;
  RCX_PACKET_GET_SLAVE_HANDLE_REQ_DATA_T  tData;
} __TLR_PACKED_POST RCX_PACKET_GET_SLAVE_HANDLE_REQ_T;
```

Packet description

Structure RCX_GET_SLAVE_HANDLE_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure TLR_PACKET_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Ignored. Set to zero for future compatibility.
ulSrcId	UINT32	any	-
ulLen	UINT32	4	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x2F08	RCX_GET_SLAVE_HANDLE_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	0	-
Structure RCX_PACKET_GET_SLAVE_HANDLE_REQ_DATA_T			
ulParam	UINT32	1, 2, 3	Which list to retrieve

© Hilscher, 2015–2017

Parameter descriptions

Parameter `ulParam`

This parameter selects which list of slave handles shall be retrieved.

Name	Numerical value	Meaning
RCX_LIST_CONF_SLAVES	0x00000001	Request the list of slave handles of configured slaves
RCX_LIST_ACTV_SLAVES	0x00000002	Request the list of slave handles of active (communicating) slaves
RCX_LIST_FAULTED_SLAVES	0x00000003	Request the list of slave handles of faulty slaves

© Hilscher, 2015–2017

3.8.1.2 RCX_GET_SLAVE_HANDLE_CNF_T confirmation

Packet structure reference

```
typedef __TLR_PACKED_PRE  struct RCX_PACKET_GET_SLAVE_HANDLE_CNF_DATA_Tag
{
  TLR_UINT32 ulParam;
  TLR_UINT32 aulHandle[1];
} __TLR_PACKED_POST  RCX_PACKET_GET_SLAVE_HANDLE_CNF_DATA_T;

typedef __TLR_PACKED_PRE struct RCX_PACKET_GET_SLAVE_HANDLE_CNF_Tag
{
  TLR_PACKET_HEADER_T                      tHead;
  RCX_PACKET_GET_SLAVE_HANDLE_CNF_DATA_T  tData;
} __TLR_PACKED_POST RCX_PACKET_GET_SLAVE_HANDLE_CNF_T;
```

Packet description

Structure RCX_PACKET_GET_SLAVE_HANDLE_CNF_T			**Type: Confirmation**
Variable	**Type**	**Value / Range**	**Description**
Structure TLR_PACKET_HEADER_T			
ulDest	UINT32		
ulSrc	UINT32		
ulDestId	UINT32	0	Ignore for future compatibility.
ulSrcId	UINT32	mirrored	
ulLen	UINT32	4 … 516	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x2F09	RCX_GET_SLAVE_HANDLE_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure RCX_PACKET_GET_SLAVE_HANDLE_CNF_DATA_T			
ulParam	UINT32	1, 2, 3	Mirrored from request
aulHandle[]	UINT32[]		Array of handles

Parameter descriptions

Parameter aulHandle

The parameter aulHandle is an array containing the list of slave handles associated with the request state. The number of entries in the list is determined by the packet length.

© Hilscher, 2015–2017

3.8.2 Get Slave Connection Info service

This API defines a service common across all master protocol firmwares. The service is used to retrieve the status of slave connection.

Note: The value of the handle used to specifiy the slave connection is equal to the value of the handle used when configuring the device with the *Configure IO Device service* on page 46.

3.8.2.1 RCX_GET_SLAVE_CONN_INFO_REQ_T request

Packet structure reference

```
typedef __TLR_PACKED_PRE  struct RCX_PACKET_GET_SLAVE_CONN_INFO_REQ_DATA_Tag
{
  TLR_UINT32 ulHandle;
} __TLR_PACKED_POST  RCX_PACKET_GET_SLAVE_CONN_INFO_REQ_DATA_T;

typedef __TLR_PACKED_PRE struct RCX_PACKET_GET_SLAVE_CONN_INFO_REQ_Tag
{
  TLR_PACKET_HEADER_T                        tHead;
  RCX_PACKET_GET_SLAVE_CONN_INFO_REQ_DATA_T  tData;
} __TLR_PACKED_POST RCX_PACKET_GET_SLAVE_CONN_INFO_REQ_T
```

Packet description

Structure RCX_GET_SLAVE_CONN_INFO_REQ_T			**Type: Request**
Variable	**Type**	**Value / Range**	**Description**
Structure TLR_PACKET_HEADER_T			
ulDest	UINT32	0x20 (LFW) Handle (LOM)	-
ulSrc	UINT32	0 (LFW) Handle (LOM)	-
ulDestId	UINT32	0	Set to zero for future compatibility.
ulSrcId	UINT32	any	-
ulLen	UINT32	4	Packet data length in bytes
ulId	UINT32	any	-
ulSta	UINT32	0	-
ulCmd	UINT32	0x2F0A	RCX_GET_SLAVE_CONN_INFO_REQ
ulExt	UINT32	0	-
ulRoute	UINT32	0	-
Structure RCX_PACKET_GET_SLAVE_CONN_INFO_REQ_DATA_T			
ulHandle	UINT32		The handle of the connection

Parameter descriptions

Parameter ulHandle

This parameter is the handle of the connection to retrieve the information for. Valid handles can be retrieved using the *Get Slave Handle service* (see page 254) Furthermore the handle used in the *Configure IO Device service* (see page 46) can be used.

© Hilscher, 2015–2017

3.8.2.2 RCX_GET_SLAVE_CONN_INFO_CNF_T confirmation

Packet structure reference

```
typedef __TLR_PACKED_PRE  struct RCX_PACKET_GET_SLAVE_CONN_INFO_CNF_DATA_Tag
{
  TLR_UINT32 ulHandle;
  TLR_UINT32 ulStructID;

  /* Feldbus specific structure follows see below */
} __TLR_PACKED_POST RCX_PACKET_GET_SLAVE_CONN_INFO_CNF_DATA_T;

typedef __TLR_PACKED_PRE struct RCX_PACKET_GET_SLAVE_CONN_INFO_CNF_Tag
{
  TLR_PACKET_HEADER_T                       tHead;
  RCX_PACKET_GET_SLAVE_CONN_INFO_CNF_DATA_T  tData;
} __TLR_PACKED_POST  RCX_PACKET_GET_SLAVE_CONN_INFO_CNF_T;

typedef struct PNM_AP_ACTIVE_SLAVE_CONNECT_INFO_Ttag PNM_AP_ACTIVE_SLAVE_CONNECT_INFO_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_ACTIVE_SLAVE_CONNECT_INFO_Ttag
{
  uint32_t                      ulLenName;
  uint32_t                      ulLenType;
  uint32_t                      ulIPAddress;
  uint32_t                      ulDiagFlags;
  uint16_t                      usDeviceID;
  uint16_t                      usVendorID;
  uint8_t                       abMac[PNIO_LEN_MAC_ADDR];
  uint16_t                      usReserved;
  uint8_t                       abName[PNIO_MAX_NAME_OF_STATION];
  uint8_t                       abType[PNIO_MAX_TYPE_OF_STATION];
};

typedef struct PNM_AP_INACTIVE_SLAVE_CONNECT_INFO_Ttag
PNM_AP_INACTIVE_SLAVE_CONNECT_INFO_T;
__PACKED_PRE struct __PACKED_POST PNM_AP_INACTIVE_SLAVE_CONNECT_INFO_Ttag
{
  uint32_t                      ulLenName;
  uint32_t                      ulDiagFlags;
  uint16_t                      usDeviceID;
  uint16_t                      usVendorID;
  uint8_t                       abName[PNIO_MAX_NAME_OF_STATION];
};

enum PNM_AP_IOD_DIAGFLAG_Etag
{
  /** the device does not exist in the network */
  PNM_AP_IOD_DIAGFLAG_NOT_EXISTING       = 0x00000001,
  /** the device is not in cyclic dataexchange */
  PNM_AP_IOD_DIAGFLAG_NOT_COMMUNICATING  = 0x00000002,
  /** Multiple stations with the ip or the name exist */
  PNM_AP_IOD_DIAGFLAG_ADDRESS_CONFLICT   = 0x00000004,
  /** The device sent an faulty DCP response */
  PNM_AP_IOD_DIAGFLAG_FAULTY_DCP         = 0x00000008,
  /** The device sent an RPC response with an error */
  PNM_AP_IOD_DIAGFLAG_NEGATIVE_RPC       = 0x00000010,
  /** The device sent an RPC response with an error */
  PNM_AP_IOD_DIAGFLAG_DEACTIVATED        = 0x00000020,
  /** Device has a module diff block identification related entries */
  PNM_AP_IOD_DIAGFLAG_MODULEDIFF_WRONG_OR_MISSING = 0x00000800,
  /** Device has a module diff block with diagnosis related entries */
  PNM_AP_IOD_DIAGFLAG_MODULEDIFF_DIAGNOSIS        = 0x00002000,
};
```


© Hilscher, 2015–2017

Packet description

Structure RCX_PACKET_GET_SLAVE_CONN_INFO_CNF_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure TLR_PACKET_HEADER_T			
ulDest	UINT32		
ulSrc	UINT32		
ulDestId	UINT32	0	Ignore for future compatibility.
ulSrcId	UINT32	mirrored	
ulLen	UINT32	260 or 508	Packet data length in bytes
ulId	UINT32	mirrored	
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	2F0B	RCX_GET_SLAVE_CONN_INFO_CNF
ulExt	UINT32	any	Ignore
ulRoute	UINT32	any	Ignore
Structure RCX_PACKET_GET_SLAVE_CONN_INFO_CNF_DATA_T			
ulHandle	UINT32		Mirrored from request
ulStructID	UINT32	0x2244, 0x2245	Identifier for the following data

Parameter descriptions

Parameter ulStructID

The parameter ulStructID denotes the type of the following data structure:

Name	Numerical value	Meaning
-	0x2244	A structure of type PNM_AP_INACTIVE_SLAVE_CONNECT_INFO_T is appended to the packet
-	0x2245	A structure of type PNM_AP_ACTIVE_SLAVE_CONNECT_INFO_T is appended to the packet

Structure PNM_AP_ACTIVE_SLAVE_CONNECT_INFO_T			
ulLenName	UINT32	1 … 240	Length of the NameOfStation
ulLenType	UINT32	0	Length of the TypeOfStation. Always Zero, as not supported
ulIPAddress	UINT32		IP Address of the device
ulDiagFlags	UINT32		Bitmask of diagnosis flags
usDeviceID	UINT16		PROFINET Device ID of the device
usVendorID	UINT16		PROFINET Vendor ID of the device
abMac[6]	UINT8		Ethernet Mac Address of the device
usReserved	UINT16		Reserved for future usage. Ignore
abName[240]	UINT8		The NameOfStation
abType[240]	UINT8		The TypeOfStation. For backwards compatibility. Not supported.

© Hilscher, 2015–2017

Structure PNM_AP_INACTIVE_SLAVE_CONNECT_INFO_T			
ulLenName	UINT32	1 … 240	Length of the NameOfStation
ulDiagFlags	UINT32		Bitmask of diagnosis flags
usDeviceID	UINT16		PROFINET Device ID of the device
usVendorID	UINT16		PROFINET Vendor ID of the device
abName[240]	UINT8		The NameOfStation

© Hilscher, 2015–2017

3.8.3 Link Status Changed service

The Link Status Changed service will be used by the PROFINET IO Controller to notify the application about link status changes. The application needs to be registered with the PROFINET IO Controller in order to receive the service. Right after registering the protocol will generate an artificial Link Status Changed service to synchronize the application with the current state.

Note: The application must generate a response packet for every Link Status Change indication. If the response is not generated, no further Link Status Changed services will be generated by the PROFINET IO Controller.

3.8.3.1 RCX_LINK_STATUS_CHANGE_IND_T indication

Packet structure reference

```
typedef __TLR_PACKED_PRE struct RCX_LINK_STATUS_Ttag
{
  TLR_UINT32  ulPort;
  TLR_BOOLEAN fIsFullDuplex;
  TLR_BOOLEAN fIsLinkUp;
  TLR_UINT32  ulSpeed;
} __TLR_PACKED_POST RCX_LINK_STATUS_T;

typedef __TLR_PACKED_PRE struct RCX_LINK_STATUS_CHANGE_IND_DATA_Ttag
{
  RCX_LINK_STATUS_T  atLinkData[2];
} __TLR_PACKED_POST RCX_LINK_STATUS_CHANGE_IND_DATA_T;

typedef __TLR_PACKED_PRE struct RCX_LINK_STATUS_CHANGE_IND_Ttag
{
  TLR_PACKET_HEADER_T               tHead;
  RCX_LINK_STATUS_CHANGE_IND_DATA_T tData;
} __TLR_PACKED_POST RCX_LINK_STATUS_CHANGE_IND_T;
```


© Hilscher, 2015–2017

Packet description

Structure RCX_LINK_STATUS_CHANGE_IND_T			Type: Request
Variable	**Type**	**Value / Range**	**Description**
Structure TLR_PACKET_HEADER_T			
ulDest	UINT32	Handle	Ignore
ulSrc	UINT32	Handle	Ignore
ulDestId	UINT32	0	Ignored. Set to zero for future compatibility.
ulSrcId	UINT32	any	Will contain value from ulSrcId from RCX_REGISTER_APPLICATION_REQ request packet
ulLen	UINT32	0	Ignore for future compatibility
ulId	UINT32	32	Packet Data Length in bytes
ulSta	UINT32	any	Ignore
ulCmd	UINT32	0	Zero in indication
ulExt	UINT32	0x2F8A	RCX_LINK_STATUS_CHANGE_IND
ulRoute	UINT32	any	Ignore
Structure RCX_LINK_STATUS_CHANGE_IND_DATA_T			
atLinkData[2]	STRUCT		

Structure RCX_LINK_STATUS_T			
ulPort	UINT32	0 to 1	Number of the Port
fIsFullDuplex	BOOL32		Non-Zero if Full Duplex Link used
fIsLinkUp	BOOL32		Non-Zero if link is up
ulSpeed	UINT32	0, 10, 100	Speed of the link in MBit/s

Parameter descriptions

Parameters atLinkData[2]

This parameter is an array of link status structures, one for each physical port.

© Hilscher, 2015–2017

3.8.3.2 RCX_LINK_STATUS_CHANGE_RES_T response

Packet structure reference

```
typedef TLR_EMPTY_PACKET_T      RCX_LINK_STATUS_CHANGE_RES_T;
```

Packet description

Structure RCX_LINK_STATUS_CHANGE_RES_T			Type: Confirmation
Variable	**Type**	**Value / Range**	**Description**
Structure TLR_PACKET_HEADER_T			
ulDest	UINT32	mirrored	-
ulSrc	UINT32	mirrored	-
ulDestId	UINT32	mirrored	-
ulSrcId	UINT32	mirrored	-
ulLen	UINT32	0	Packet data length in bytes
ulId	UINT32	mirrored	-
ulSta	UINT32		=0: no error <> 0: see section *Status codes / Error codes* on page 276.
ulCmd	UINT32	0x2F8B	RCX_LINK_STATUS_CHANGE_RES
ulExt	UINT32	mirrored	-
ulRoute	UINT32	mirrored	-

© Hilscher, 2015–2017

3.9 DPM

This section describes additional data structures used by the PROFINET IRT Controller firmware for the DPM interface.

3.9.1 Extended Status Block

The extend status block provides additional information about the firmwware state. This block contains structures common for all protocol firmwares and specific structures. The following structure is derived from the NETX_EXTENDED_STATE_FIELD_DEFINITION_T described in the DPM manual.

Structure reference

```
typedef __RCX_PACKED_PRE struct PNM_AP_EXTENDED_STATE_FIELD_DEFINITION_Ttag
{
  PNM_AP_IOTIMINGINFO_T tIotimingInfo;
  uint8_t               abReserved[140];
  NETX_EXTENDED_STATE_FIELD_T tExtStateField;
} __RCX_PACKED_POST PNM_AP_EXTENDED_STATE_FIELD_DEFINITION_T;
```

Structure description

Structure PNM_AP_EXTENDED_STATE_FIELD_DEFINITION			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_TIMEINFO_T			
tIoTimingInfo	PNM_AP_IOTIMINGINFO_T		Process Data Timing information synchronized to DPM input area handshake.
abReserved	UINT8[140]		Reserved for future usage
tExtStateField	NETX_EXTENDED_STATE_FIELD_T		Descriptor for state field lists in DPM input/output area. See DPM manual for details

© Hilscher, 2015–2017

3.9.2 Process data timing information

The following structures are used in the context of process data timing measurements.

Structure reference

```
typedef struct PNS_AP_TIMEINFO_Ttag PNS_AP_TIMEINFO_T;

__PACKED_PRE struct __PACKED_POST PNS_AP_TIMEINFO_Ttag
{
  uint32_t ulCycleCounter;
  uint32_t ulTimeSinceCycleStart;
};
```

Structure description

Structure PNM_AP_TIMEINFO_T			**Type: Structure**
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_TIMEINFO_T			
ulCycleCounter	UINT32		The value of the PROFINET synchronous cycle counter
ulTimeSinceCycle Start	UINT32		Time since start of the bus cycle

Parameter descriptions

Parameter ulCycleCounter

The value of the PROFINET Cycle Counter when the time measurement was taken

Parameter ulTimesinceCycleStart

The actual time of the event/state relative to the beginning of the bus cycle. The time is given in nanoseconds.

Structure reference

```
typedef struct PNM_AP_IOTIMINGINFO_Ttag PNM_AP_IOTIMINGINFO_T;

__PACKED_PRE struct __PACKED_POST PNM_AP_IOTIMINGINFO_Ttag
{
  PNS_AP_TIMEINFO_T tOutputUpdateStart;
  PNS_AP_TIMEINFO_T tOutputUpdateFinish;
  PNS_AP_TIMEINFO_T tInputUpdateStart;
  PNS_AP_TIMEINFO_T tInputUpdateFinish;
};
```


© Hilscher, 2015–2017

Structure description

Structure PNM_AP_IOTIMINGNFO_T			Type: Structure
Variable	**Type**	**Value / Range**	**Description**
Structure PNM_AP_IOTIMINGNFO_T			
tOutputUpdateStart	PNS_AP_TIMEINFO_T		At this time the controller firmware started updating the transmit buffer from DPM output area
tOutputUpdateFinish	PNS_AP_TIMEINFO_T		At this time the controller firmware finished updating the transmit buffer from DPM output area
tInputUpdateStart	PNS_AP_TIMEINFO_T		At this time the controller firmware started updating the DPM input area from the receive buffer.
tInputUpdateFinish	PNS_AP_TIMEINFO_T		At this time the controller firmware finished updating the DPM input area from the receive buffer.

Parameter descriptions

Parameter tOutputUpdateStart

This parameter represents the time when the PROFINET controller firmware starts copying the data from the DPM output area into the transmit buffer. This happens when the host handshakes the DPM output area.

Parameter tOutputUpdateFinish

This field is the time when the PROFINET controller firmware finished copying the data from the DPM output area into the transmit buffer. The data will be transmitted in the next buscycle after the given timepoint.

Note: The firmware does not handshake the DPM output area at this time

Parameter tInputUpdateStart

This parameter represents the time when the PROFINET controller firmware starts copying the data from the receive buffers to the the DPM input area.

Note: This time is internally linked to the end of the IRT red phase. The host must have handshaked the DPM input area prior to this time event in order for the copying to take place.

Parameter tInputUpdateFinish

This field is the time when the PROFINET controller firmware finished copying the data from the receive buffers to the the DPM input area. This event occurs right before the firmware handshakes the DPM input area.

© Hilscher, 2015–2017

3.10 Fragmented packet transfer

In several situations the size of a packet exceeds the size of the DPM mailbox. In such cases a fragmented packet transfer sequence is used: The data part of a packet is splitted into parts. Each part is prepended with the usual packet header where the ulExt, ulCmd and ulId fields are used to manage the fragmented transfer. The following image shows a fragmented request/confirmation sequence.

Figure 16: Fragmented request/Confirmation sequence

Please note that the image shows a bi-directional fragmented transfer for one service. In most cases only one of request or confirmation will be fragmented. Also not shown in the image is the usage of other bits in ulExt field.

© Hilscher, 2015–2017

In a similar manner an Indication/Response service can be transferred using fragments: This is shown in the next figure:

Figure 17: Fragmented indication/response sequence

The next tabelle lists all services which use fragmentation:

Service	Details
Load Remanent service	Fragmented Request
Read submodule record service	Fragmented Confirmation
Write submodule record service	Fragmented Request
Read implicit record service	Fragmented Confirmation
Store Remanent service	Fragmented Indication

Table 45: Services using packet fragmentation

© Hilscher, 2015–2017

4 Media Redundancy

4.1 Overview

Media Redundancy provides a means to recover in a short time from network cable breaks. The PROFINET Standard defines the Media Redundancy Protocol (MRP) and Media Redundancy with Planned Duplication of Frames (MRPD) for this purpose.

MRP ensures, when devices are connected in a ring topology, that frames do not circle in the ring forever and a fast reconfiguring of the switch forwarding tables in case of a ring break is done. It does not provide a reliable solution for bumpless media redundancy. That means that when a cable break occurs, connections between the controller and the devices might be interrupted for a short time.

MRPD requires MRP protocol to manage the ring topology and is an extension to IRT communication. It provides bumpless media redundancy for IRT communication even in case of synchronous applications. That means when the ring breaks, it is guaranteed that there will be no communication abort and no synchronization loss.

Note: Using MRP Manager functionality is covered by patents. Please contact the PROFINET User Organization for further details if the Hilscher PROFINET IO Controller is configured for MRP Manager role.

In order to prevent accidental activation of MRP Manager role a License Bit for the Security EEPROM has been implemented. The PROFINET Controller Firmware will reject any Configuration where the PROFINET Controller is configured for MRP Manager Role if this bit is not activated.

Note: MRPD is currently not supported by the Hilscher PROFINET IO Controller firmware.

© Hilscher, 2015–2017

4.2 Topologies

Generally, all devices which are connected in a ring topology must support the MRP protocol in client mode. At least one device in the ring must support MRP Manager. The following image shows the simple example of a MRP ring topology:

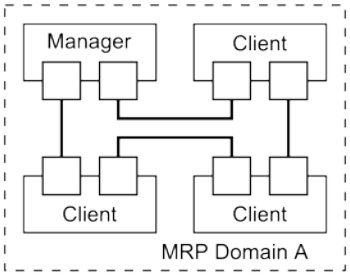

Figure 18: MRP single ring topology

The image shows a ring manager connected with three ring clients in a ring topology. While it is often the case, the ring manager and the profinet controller need not to be the same device. Each device supporting ring manager capability can be configured as manager. It is also possible to have multiple rings in a configuration as shown in the next image:

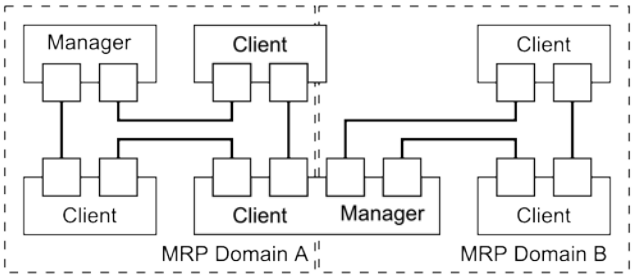

Figure 19: Ring topology with two rings

In such a case each ring must be assigned an individual MRP domain and own MRP domain UUID. Rings must not be mixed, e.g. it is not allowed to share a network cable between two rings.

© Hilscher, 2015–2017

4.3 Configuration parameters

The MRP Configuration consists of several parameters which depend on communication recovery time requirements and rings size. (number of nodes in the ring).

Manager Parameters

Parameter	Value range	Default value	Meaning
Manager parameters			
Topology change Interval	1 … 100 ms	10 ms	The time interval between two topology change interval network frames. The topology change frame is used by the MRP manager to indicate changes in topology (ring break/close) to the MRP clients.
Topology change repeat count TOP_{NRmax}	1 … 5	5	The count how often the MRP manager will repeat the topology change frame. This value multiplied with the topology change interval determines the delay between the detection of the ring break and the reorganization of the network switches.
Test short interval	1 … 500 ms	2 ms	Short interval between ring test frames sent by the MRP manager to detect the ring state. The short interval is used when ever a ring transitions between closed or open state occurs to verify the ring state.
Test default interval	1 … 1000 ms	10 ms	Default interval between ring test frames sent by the MRP manager to detect the ring state. The default interval is used when the ring is in open or in closed state to detect changes of the ring state.
Test NR Max TST_{NRmax}	2 ... 10	3	The number of ring test frames which cause a state transition in MRP manager. E.g. when the ring breaks the MRP manager will wait until Test NR max frames are not passed through the ring before considering the ring as broken. This value multiplied by the short test interval and the default test interval determines the delay between the physical ring break/close and the Ring manager considering the ring as open/closed
Client parameters			
Link Down Timeout	1 … 1000 ms	20 ms	Time interval between two MRP link down frames sent by the MRP client when a link down on a port occurs.
Link Up Timeout	1 … 1000 ms	20 ms	Time interval between two MRP link up frames sent by the MRP client when a link up on port occurs.
Link NR Max	1 … 5	5	The maximum number of Link/Down frames the MRP client will sent while waiting for a reaction of the MRP manager.

Table 46 MRP parameter summary

© Hilscher, 2015–2017

From the parameters above the following times can be calculated

- Ring test time T_{test}
  $T_{test} = T_{TSTdefault} \times TST_{NRmax}$
- Traveling time of one ring test frame through the ring T_{ring}:
  $T_{ring} = N \times (T_{switch} + T_{queue} + T_{bit} + T_{line})$
  Where N is the number of Nodes in the ring, T_{switch} and T_{queue} are the times for passing a MRP test frame through the switch, T_{bit} is the time for sending a frame through the network and T_{line} the propagation time of the Ethernet signal.
- Detection time of a ring failure T_{detect}:
  $T_{detect} = T_{test} + T_{ring}$
- Time to hold switch forwarding table before flushing T_{hold}
  $T_{hold} = T_{topochginterval} \times TOP_{NRmax}$
- Time for flushing the switch forwarding table of all devices T_{flush}:
  $T_{flush} = T_{hold} + T_{ring} + T_{FDB}$
  where T_{FDB} is the time required for a device to flush its forwarding table
- Total time for reconfiguring the switches after a ring break/close T_{rec}:
  $T_{rec} = T_{detect} + T_{flush}$

The following estimates can be used in the calculations:

- T_{line} = 0,5 µs (For 100 m cable length)
- T_{bit} = 5,12 µs (For MRP Test Frame)
- T_{queue} = 122 µs (worst case value with one large frame in send queue)
- T_{switch} = 10 µs (depends on switches, typical value for store & forward)
- T_{FDB} = 500µs (depends on switch)

© Hilscher, 2015–2017

5 LED

The following leds are defined for the PROFINET IO IRT Controller firmware:

- SYS LED: Indicates the status of the system, colors: green, yellow
- COM0 LED: BF LED (Bus Fail), color: red
- COM1 LED: SF LED (System Fail), color: red

The table summarizes all available led states of the loadable firmware:

SYS	SF	BF	Description
Firmware and Configuration			
Off	Off	Off	Power supply for the device is missing or hardware defect.
On, yellow	Off	Off	No second stage bootloader found in Flash memory.
Flashing, green/yellow, cyclic	Off	Off	No firmware file found in Flash file system.
On, green	On, red	Off	PROFINET IO Controller is not configured.
On, green	Off	On, red	No Ethernet port has a link. E.g. no cable connected to any of the Ethernet ports.
On, green	Off	Flashing, red, 2 Hz	PROFINET IO Controller is not online (Bus is switched to Off).
PROFINET communication			
On, green	Off or On, red	Flashing, red, 1 Hz	Not all configured devices are in data exchange.
On, green	On, red	-	One IO Device connected to the PROFINET IO Controller reports a problem.
On, green	Off	Off	All devices are in data exchange and no problem has been reported by any device.
PROFINET Controller operation			
On, green	Flashing, red, 1 Hz, 3 s	Off	A PROFINET DCP Set Signal has been received.
On, green	Flashing red, 2 Hz	Flashing, red, 2 Hz	The PROFINET IO Controller has detected an address conflict. Another device in the network is using the same Name of Station or IP address as the PROFINET IO Controller. or Watchdog error.
On, green	On, red	On, red	No valid Master license.

Table 47: PROFINET IO Controller LED status (Loadable firmware)

© Hilscher, 2015–2017

6 Certification requirements for applications

To pass the PROFINET V2.3 certification examination for IO Controllers, the application that uses the Hilscher PROFINET IO Controller firmware has to fulfill several requirements. The list of important topics compiled in this section provides tips on the essential points, but it is not exhaustive. The official PI documents (e.g. specifications for tests and test cases) are valid and subject to modification by PI. Check these documents and contact your PI test lab in time to find out which requirements are relevant for your application.

General requirements

The configuration of IO Controller firmware has to be set to “application-controlled startup”. The application has to

- register at the IO Controller firmware via Register Application Request (see reference [2])
- handle the services for remanent data *Load Remanent service* (see page 81) and Store Remanent service (see page 188)
- use the *Configure OEM Parameter service* (see page 25) to set correct values for identification information (e.g. OrderID, software version,…)

Moreover, a special GSDML file containing information on controller identification has to be provided in order to be able to run automated test cases.

Requirements for IO data handling

The application used for certification tests has to demonstrate its ability to access IOPS and IOCS received from an IO Device to the test lab. The test lab will check the reaction to “IOPS invalid”.

Requirements for acyclic data handling

The application has to be able to perform a read request

- reading exactly 1 byte of a specific record (1 byte ONLY).
- reading 16 KB of a specific record.

Requirements for alarm handling

The application has to be able to demonstrate received alarms and to identify the submodule that triggered them. During the test, alarm type and alarm payload have to be demonstrated as well.

Requirements for automated DCP test case

- Via SYCON.net (activate “Accept DCP Set via network”) or the *Configure IO Controller service* (page 33, set PNM_AP_IOC_FLAG_ENABLE_DCP_SET) the configuration has to set “accept DCP SET parameters” of the IO Controller.
- The firmware must be set to an application-controlled startup.
- The application must handle remanent data services.

© Hilscher, 2015–2017

Requirements for netload tests

During the netload tests, the execution of the following specific sequence of read requests is expected for a certain device and submodule. The application has to supervise the AR to see whether the AR of the configured IO devices is active. In case of an AR shutdown, the application has to demonstrate this shutdown. The application has to store permanently that the “AR got lost”. Human interaction is required to delete this information after the completion of the test which takes several hours.

Specific read request sequence:

1. I&M0 record (0xAFF0)
2. Diagnosis data for one device (0xF80C)
3. Diagnosis data for one API (0xF00C)
4. Diagnosis data for one AR (0xE00C)
5. Diagnosis data for one Slot (0xC00C)
6. Diagnosis data for one Slot (0x800C)

The max. interval between consecutive read requests is 100 ms. The application has to provide a counter for performed read requests and the status of the last read confirmation.

More recommendations

Experience shows that the application should display helpful diagnosis information for fault cases. We therefore recommend supporting the following items:

- Station diagnosis to indicate and support the following device states:
  - not connected
  - connected
  - has information on diagnosis

  Use the Get Slave Connection Info service (see page 257) to get the device states.
- Read request with a variable max. amount of data to be read from the submodule
- Event logs from Controller and all devices using the *Get Logbook Service* (see page 156). The time when the event occurs and the local machine time have to be synchronized.

© Hilscher, 2015–2017

7 Status codes / Error codes

7.1 Error Codes of PROFINET IO Controller

Status code / Error code	Definition/description
0x00000000	TLR_S_OK Status ok. No error.
Invalid parameter	**Definition/description**
0xC0CB0001	PNM_AP_CFG_INVALID_PARAMETER
0xC0CB0002	PNM_AP_CFG_INVALID_STRUCT_VERSION
0xC0CB0003	PNM_AP_CFG_INVALID_IDENTIFIER
AR	**Definition/description**
0xC0CB0004	PNM_AP_CFG_INVALID_DEVICE_HANDLE
0xC0CB0005	PNM_AP_CFG_INVALID_SEND_CLOCK_FACTOR
0xC0CB0006	PNM_AP_CFG_INVALID_REDUCTION_RATIO
0xC0CB0007	PNM_AP_CFG_INVALID_DATA_HOLD_FACTOR
0xC0CB0008	PNM_AP_CFG_INVALID_PHASE
0xC0CB0009	PNM_AP_CFG_INVALID_FRAME_SEND_OFFSET
0xC0CB000A	PNM_AP_CFG_INVALID_DOMAIN_NAME
0xC0CB000B	PNM_AP_CFG_INVALID_UUID
0xC0CB000C	PNM_AP_CFG_INVALID_NUMBER_OF_ENTRIES
Config IOC and Config IOD	**Definition/description**
0xC0CB000D	PNM_AP_CFG_INVALID_PORT_NUM
0xC0CB000E	PNM_AP_CFG_INVALID_STATION_NAME
0xC0CB000F	PNM_AP_CFG_INVALID_PORT_NAME
0xC0CB0010	PNM_AP_CFG_INVALID_IP_ADDRESS
Config IOCR	**Definition/description**
0xC0CB0011	PNM_AP_CFG_INVALID_IOCR_HANDLE
0xC0CB0012	PNM_AP_CFG_INVALID_IOCR_TYPE
0xC0CB0013	PNM_AP_CFG_INVALID_IOCR_PROP
0xC0CB0014	PNM_AP_CFG_INVALID_IOCR_DATA_LENGTH
0xC0CB0015	PNM_AP_CFG_INVALID_DPM_OFFSET
Config submodule	**Definition/description**
0xC0CB0017	PNM_AP_CFG_INVALID_SUBMODULE_HANDLE
0xC0CB0018	PNM_AP_CFG_INVALID_SUBMODULE_TYPE
0xC0CB0019	PNM_AP_CFG_INVALID_SUBMODULE_FLAG
0xC0CB001A	PNM_AP_CFG_INVALID_INPUT_DATA_LENGTH
0xC0CB001B	PNM_AP_CFG_INVALID_INPUT_FRAME_OFFSET
0xC0CB001C	PNM_AP_CFG_OVERLAPPING_INPUT_FRAME_OFFSET
0xC0CB001D	PNM_AP_CFG_INVALID_INPUT_IOCS_OFFSET
0xC0CB001E	PNM_AP_CFG_OVERLAPPING_INPUT_IOCS_OFFSET
0xC0CB001F	PNM_AP_CFG_INVALID_OUTPUT_DATAT_LENGTH
0xC0CB0020	PNM_AP_CFG_INVALID_OUTPUT_FRAME_OFFSET

© Hilscher, 2015–2017

Status code / Error code	Definition/description
0xC0CB0021	PNM_AP_CFG_OVERLAPPING_OUTPUT_FRAME_OFFSET
0xC0CB0022	PNM_AP_CFG_INVALID_OUTPUT_IOCS_OFFSET
0xC0CB0023	PNM_AP_CFG_OVERLAPPING_OUTPUT_IOCS_OFFSET
IRT	**Definition/description**
0xC0CB0024	PNM_AP_CFG_INVALID_PLL_WINDOW
0xC0CB0025	PNM_AP_CFG_INVALID_PTCTP_TIMEOUT
0xC0CB0026	PNM_AP_CFG_INVALID_TAKEOVER_TIMEOUT
0xC0CB0027	PNM_AP_CFG_INVALID_PTCP_STARTUP_TIME
0xC0CB0028	PNM_AP_CFG_INVALID_PTCP_MASTER_PRIO
0xC0CB0029	PNM_AP_CFG_INVALID_NUM_IRT_PHASES
0xC0CB002A	PNM_AP_CFG_INVALID_NUM_IRT_FRAMES
0xC0CB002B	PNM_AP_CFG_UNUSED
0xC0CB002C	PNM_AP_CFG_INVALID_GREEN_PERIOD_BEGIN
0xC0CB002E	PNM_AP_CFG_INVALID_ORANGE_PERIOD_BEGIN
0xC0CB002F	PNM_AP_CFG_INVALID_FRAME_LENGTH
0xC0CB0030	PNM_AP_CFG_INVALID_LINE_DELAY
0xC0CB0031	PNM_AP_CFG_INVALID_PREAMBLE_LENGTH
0xC0CB0032	PNM_AP_CFG_INVALID_PARAMETER_FLAG
0xC0CB0033	PNM_AP_CFG_INVALID_PARAMETER_TYPE
FSU	**Definition/description**
0xC0CB0034	PNM_AP_CFG_INVALID_FSU_MODE
0xC0CB0035	PNM_AP_CFG_INVALID_FS_HELLO_INTERVAL
0xC0CB0036	PNM_AP_CFG_INVALID_FS_HELLO_RETRY
0xC0CB0037	PNM_AP_CFG_INVALID_FS_HELLO_DELAY
0xC0CB0038	PNM_AP_CFG_INVALID_SYNC_MODE
0xC0CB0039	PNM_AP_CFG_INVALID_MAUTYPE_MODE
0xC0CB003A	PNM_AP_CFG_INVALID_DOMAIN_BOUNDARY
0xC0CB003B	PNM_AP_CFG_INVALID_DCP_BOUNDARY
0xC0CB003C	PNM_AP_CFG_INVALID_PEERTOPEER_BOUNDARY
0xC0CB003D	PNM_AP_CFG_INVALID_MULTIPLE_INTERFACE_MODE
0xC0CB003E	PNM_AP_CFG_INVALID_MRP_INSTANCE
0xC0CB003F	PNM_AP_CFG_INVALID_MRP_CHECK
0xC0CB0040	PNM_AP_CFG_INVALID_MRP_ROLE
0xC0CB0041	PNM_AP_CFG_INVALID_MRP_PARAMETERS
0xC0CB0042	PNM_AP_CFG_INVALID_MRP_MANAGER_PRIO
0xC0CB0043	PNM_AP_CFG_INVALID_MRP_TOPO_CHANGE_INTERVAL
0xC0CB0044	PNM_AP_CFG_INVALID_MRP_TOPO_REPEAT_COUNT
0xC0CB0045	PNM_AP_CFG_INVALID_MRP_SHORT_TEST_INTERVAL
0xC0CB0046	PNM_AP_CFG_INVALID_MRP_DEFAULT_TEST_INTERVAL
0xC0CB0047	PNM_AP_CFG_INVALID_MRP_TEST_MONITOR_COUNT
0xC0CB0048	PNM_AP_CFG_INVALID_MRP_LINK_DOWN_INTERVAL
0xC0CB0049	PNM_AP_CFG_INVALID_MRP_LINK_UP_INTERVAL

© Hilscher, 2015–2017

Status code / Error code	Definition/description
0xC0CB004A	PNM_AP_CFG_INVALID_MRP_LINK_CHANGE_COUNT
0xC0CB004B	PNM_AP_CFG_INVALID_FIBEROPTIC_PARAMETERS
0xC0CB004C	PNM_AP_CFG_DUPLICAT_UUID
0xC0CB004D	PNM_AP_CFG_DUPLICAT_NAME_OF_STATION
0xC0CB004E	PNM_AP_CFG_DUPLICAT_IP_ADDRESS
0xC0CB004F	PNM_AP_CFG_INVALID_RTA_TIMEOUT_FACTOR
0xC0CB0050	PNM_AP_CFG_INVALID_RTA_RETRIES
0xC0CB0051	PNM_AP_CFG_INVALID_MAX_ALARM_DATA_LENGTH
0xC0CB0052	PNM_AP_CFG_AR_TYPE_NOT_SUPPORTED
0xC0CB0053	PNM_AP_CFG_INVALID_AR_TYPE
0xC0CB0054	PNM_AP_CFG_INVALID_SLOT
0xC0CB0055	PNM_AP_CFG_INVALID_MODULE_IDENT
0xC0CB0056	PNM_AP_CFG_INVALID_SUBSLOT
0xC0CB0057	PNM_AP_CFG_NO_PORT_SUBMODULE
0xC0CB0058	PNM_AP_CFG_INVALID_MAX_BRIDGE_DELAY
0xC0CB0059	PNM_AP_CFG_INVALID_MAX_PORT_TX_DELAY
0xC0CB005A	PNM_AP_CFG_INVALID_MAX_PORT_RX_DELAY
0xC0CB005B	PNM_AP_CFG_INVALID_MAX_LINE_DELAY
0xC0CB005C	PNM_AP_CFG_INVALID_YELLOW_TIME
0xC0CB005D	PNM_AP_CFG_INVALID_FRAME_DATA_PROP
0xC0CB005E	PNM_AP_CFG_INVALID_FRAME_ID
0xC0CB005F	PNM_AP_CFG_INVALID_FRAME_DETAIL_SYNC_FRAME_MASK
0xC0CB0060	PNM_AP_CFG_INVALID_FRAME_DETAIL_FRAME_SEND_OFFSET
0xC0CB0061	PNM_AP_CFG_INVALID_NUM_IRT_ASSIGNMENT
0xC0CB0062	PNM_AP_CFG_DUPLICAT_SYNC_MASTER
0xC0CB0063	PNM_AP_CFG_INVALID_NUMBER_OF_PORTS
0xC0CB0064	PNM_AP_CFG_TOPO_PORT_ALREADY_CONFIGURED
0xC0CB0065	PNM_AP_CFG_TOPO_INFO_MISMATCH
0xC0CB0066	PNM_AP_CFG_INVALID_CONFIG_STATE
0xC0CB0067	PNM_AP_CFG_MISSING_IOCR
0xC0CB0068	PNM_AP_CFG_MAX_NUMBER_IOCR_EXCEEDED
0xC0CB0069	PNM_AP_CFG_INVALID_DAP_CONFGURATION
0xC0CB006A	PNM_AP_CFG_INVALID_DROP_BUDGET
0xC0CB006B	PNM_AP_CFG_INVALID_REDORANGE_PERIOD_BEGIN
0xC0CB006C	PNM_AP_CFG_PERFORMANCELIMIT_EXCEEDED
0xC0CB006D	PNM_AP_CFG_FEATURE_NOT_SUPPORTED
0xC0CB006E	PNM_AP_CFG_IRT_LEGACY_ADVANCED_MIXED

Table 48: Status codes / Error codes: PROFINET IO Controller

© Hilscher, 2015–2017

7.2 Sockets

Status code / Error code	Definition/description
0x00000000	TLR_S_OK Status ok
0xC0C90001	TLR_E_SOCK_UNSUPPORTED_SOCKET Unsupported socket domain, type and protocol combination.
0xC0C90002	TLR_E_SOCK_INVALID_SOCKET_HANDLE Invalid socket handle.
0xC0C90003	TLR_E_SOCK_SOCKET_CLOSED Socket was closed.
0xC0C90004	TLR_E_SOCK_INVALID_OP The command is invalid for the particular socket.
0xC0C90005	TLR_E_SOCK_INVALID_ADDRESS_FAMILY An invalid address family was used for this socket
0xC0C90006	TLR_E_SOCK_IN_USE The specified address is already in use.
0xC0C90007	TLR_E_SOCK_HUP The remote side closed the connection.
0xC0C90008	TLR_E_SOCK_WOULDBLOCK The operation would block.

Table 49: Status codes / Error codes: Sockets

7.3 PROFINET Status Code

The PROFINET protocol uses the PROFINET status code to indicate success or failure. This section describes the meanings of the PROFINET status code. The according field is named `ulPnio`.

These services uses the PROFINET status code:

- Read submodule record service (page 118)
- Write submodule record service (page 124)
- Read implicit record service (page 128)

The PROFINET status code is an unsigned 32 bit integer value that is structured into four fields as shown in the following figure.

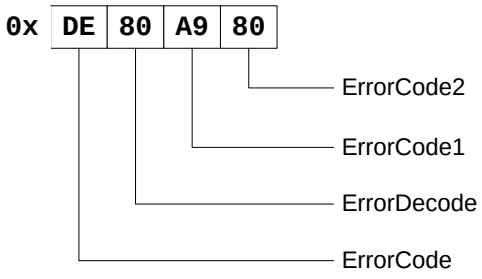

Figure 20: Structure of the PROFINET Status Code

The fields define a hierarchy on the error codes. The ordering is as follows:

- ErrorCode,
- ErrorDecode,
- ErrorCode1 and
- ErrorCode2.

The meaning of lower order fields depends on the values of the higher order fields.

Value 0x00000000 means no error or success.

© Hilscher, 2015–2017

7.3.1 The ErrorCode Field

This field defines the domain, in which the error occured. The following table shows the valid values:

Value	Meaning/Use	Description
0x20 - 0x3F	Manufacturer specific: for Log Book	To be used when the application generates a log book entry. Meaning of the values is manufacturer specific (Defined by manufacturer of the device.)
0x81	PNIO: for all errors not covered elsewhere	Used for all errors not covered by the other domains
0xCF	RTA error: used in RTA error PDUs	Used in alarm error telegrams (Low Level)
0xDA	AlarmAck: used in RTA data PDUs	Used in alarm data telegrams (High Level)
0xDB	IODConnectRes: RPC Connect Response	Used to indicate errors occurred when handling a Connect request.
0xDC	IODReleaseRes: RPC Release Response	Used to indicate errors occurred when handling a Release request.
0xDD	IODControlRes: RPC Control Response	Used to indicate errors occurred when handling a Control request.
0xDE	IODReadRes: RPC Read Response	Used to indicate errors occurred when handling a Read request.
0xDF	IODWriteRes: RPC Write Response	Used to indicate error occurred when handling a Write response

Table 50: Coding of PNIO Status ErrorCode (Excluding reserved Values)

7.3.2 The ErrorDecode Field

This field defines the context, under which the error occurred.

Value	Meaning/Use	Description
0x80	PNIORW: Application errors of the services Read and Write	Used in the context of Read and Write services handled by either the PROFINET Device stack or by the Application
0x81	PNIO: Other Services	Used in context with any other services.
0x82	Manufacturer Specific: LogBook Entries	Used in context with LogBook entries generated by the stack or by the application.

Table 51: Coding of PNIO status ErrorDecode (Excluding reserved Values)

© Hilscher, 2015–2017

7.3.3 The ErrorCode1 and ErrorCode2 Fields

The ErrorCode1 and ErrorCode2 fields finally specify the error. The meaning of the both fields depends on the value of the ErrorDecode Field.

7.3.3.1 ErrorCode1 and ErrorCode2 for ErrorDecode = PNIORW

If the field ErrorDecode is set to the value PNIORW, the field ErrorCode2 shall be encoded user specific and the ErrorCode1 field is split into an ErrorClass Nibble and an ErrorDecode Nibble as shown in Table 52.

Note: As the ErrorCode2 field can be freely chosen in this case, the application may use it to provide more detailed information about the error (e.g. why writing the record into the module was not possible, which value of the parameter was wrong…)

ErrorClass (Bit 7 – 4)	Meaning	ErrorDecode (Bit 3- 0)	Meaning
0xA	Application	0x0	Read Error
		0x1	Write Error
		0x2	Module Failure
		0x7	Busy
		0x8	Version Conflict
		0x9	Feature Not Supported
		0xA-0xF	User Specific
0xB	Access	0x0	Invalid Index
		0x1	Write Length Error
		0x2	Invalid Slot/Subslot
		0x3	Type Conflict
		0x4	Invalid Area/Api
		0x5	State Conflict
		0x6	Access Denied
		0x7	Invalid Range
		0x8	Invalid Parameter
		0x9	Invalid Type
		0xA	Backup
		0xB-0xF	User specific
0xC	Resource	0x0	Read Constrain Conflict
		0x1	Write Constrain Conflict
		0x2	Resource Busy
		0x3	Resource Unavailable
		0x8-0xF	User specific
0xD-0xF	User Specific	0x0-0xF	User specific

Table 52: Coding of ErrorCode1 for ErrorDecode = PNIORW (Excluding reserved Values)

© Hilscher, 2015–2017

7.3.4 ErrorCode1 and ErrorCode2 for ErrorDecode = PNIO

The meaning of ErrorCode1 and ErrorCode2 for ErrorDecode = PNIO is shown in the following table. This kind of PROFINET status codes should only be used in conjunction with the AR Abort Request Service (See end of table).

Coding of ErrorCode1 and ErrorCode 2 for ErrorDecode = PNIO:

ErrCode1	Description	ErrCode2	Description / Use
0x01	Connect Parameter Error. Faulty ARBlockReq	0x00-0x0D	Error in one of the Block Parameters
0x02	Connect Parameter Error. Faulty IOCRBlockReq	0x00-0x1C	Error in one of the Block Parameters
0x03	Connect Parameter Error. Faulty ExpectedSubmoduleBlockReq	0x00-0x10	Error in one of the Block Parameters
0x04	Connect Parameter Error. Faulty AlarmCRBlockReq	0x00-0x0F	Error in one of the Block Parameters
0x05	Connect Parameter Error. Faulty PrmServerBlockReq	0x00-0x08	Error in one of the Block Parameters
0x06	Connect Parameter Error. Faulty MCRBlockReq	0x00-0x08	Error in one of the Block Parameters
0x07	Connect Parameter Error. Faulty ARRPCBlockReq	0x00-0x04	Error in one of the Block Parameters
0x08	Read Write Record Parameter Error. Faulty Record	0x00-0x0C	Error in one of the Block Parameters
0x09	Connect Parameter Error Faulty IRInfoBlock	0x00-0x05	Error in one of the Block Parameters
0x0A	Connect Parameter Error Faulty SRInfoBlock	0x00-0x05	Error in one of the Block Parameters
0x0B	Connect Parameter Error Faulty ARFSUBlock	0x00-0x05	Error in one of the Block Parameters
0x14	IODControl Parameter Error. Faulty ControlBlockConnect	0x00-0x09	Error in one of the Block Parameters
0x15	IODControl Parameter Error. Faulty ControlBlockPlug	0x00-0x09	Error in one of the Block Parameters
0x16	IOXControl Parameter Error. Faulty ControlBlock after a connection establishment	0x00-0x07	Error in one of the Block Parameters
0x17	IOXControl Parameter Error. Faulty ControlBlock a plug alarm	0x00-0x07	Error in one of the Block Parameters
0x18	IODControl Parameter Error Faulty ControlBlockPrmBegin	0x00-0x09	Error in one of the Block Parameters
0x19	IODControl Parameter Error Faulty SubmoduleListBlock	0x00-0x07	Error in one of the Block Parameters
0x28	Release Parameter Error. Faulty ReleaseBlock	0x00-0x07	Error in one of the Block Parameters
0x32	Response Parameter Error. Faulty ARBlockRes	0x00-0x08	Error in one of the Block Parameters
0x33	Response Parameter Error. Faulty IOCRBlockRes	0x00-0x06	Error in one of the Block Parameters
0x34	Response Parameter Error. Faulty AlarmCRBlockRes	0x00-0x06	Error in one of the Block Parameters
0x35	Response Parameter Error. Faulty ModuleDiffBlock	0x00-0x0D	Error in one of the Block Parameters

© Hilscher, 2015–2017

ErrCode1	Description	ErrCode2	Description / Use
0x36	Response Parameter Error. Faulty ARRPCBlockRes	0x00-0x04	Error in one of the Block Parameters
0x37	Response Parameter Error Faulty ARServerBlockRes	0x00-0x05	Error in one of the Block Parameters
0x3C	AlarmAck Error Codes	0x00	Alarm Type Not Supported
		0x01	Wrong Submodule State
0x3D	CMDEV	0x00	State conflict
		0x01	Resource
		0x02-0xFF	State machine specific
0x3E	CMCTL	0x00	State conflict
		0x01	Timeout
		0x02	No data send
0x3F	CTLDINA	0x00	No DCP active
		0x01	DNS Unknown_RealStationName
		0x02	DCP No_RealStationName
		0x03	DCP Multiple_RealStationName
		0x04	DCP No_StationName
		0x05	No_IP_Addr
		0x06	DCP_Set_Error
0x40	CMRPC	0x00	ArgsLength invalid
		0x01	Unknown Blocks
		0x02	IOCR Missing
		0x03	Wrong AlarmCRBlock count
		0x04	Out of AR Resources
		0x05	AR UUID Unknown
		0x06	State conflict
		0x07	Out of Provider, Consumer or Alarm Resources
		0x08	Out of memory
		0x09	PDev already owned
		0x0A	ARset State conflict during connection establishment
		0x0B	ARset Parameter conflict during connection establishment
0x41	ALPMI	0x00	Invalid state
		0x01	Wrong ACK-PDU
0x42	ALPMR	0x00	Invalid state
		0x01	Wrong Notification PDU
0x43	LMPM	0x00–0xFF	Ethernet Switch Errors
0x44	MAC	0x00-0xFF	
0x45	RPC	0x01	CLRPC_ERR_REJECTED: EPM or Server rejected the call.
		0x02	CLRPC_ERR_FAULTED: Server had fault while executing the call

© Hilscher, 2015–2017

ErrCode1	Description	ErrCode2	Description / Use
		0x03	CLRPC_ERR_TIMEOUT: EPM or Server did not respond
		0x04	CLRPC_ERR_IN_ARGS; Broadcast or maybe “ndr_data” too large
		0x05	CLRPC_ERR_OUT_ARGS: Server sent back more than “alloc_len”
		0x06	CLRPC_ERR_DECODE: Result of EPM Lookup could not be decoded
		0x07	CLRPC_ERR_PNIO_OUT_ARGS: Out-args not “PN IO signature”, too short or inconsistent
		0x08	CLRPC_ERR_PNIO_APP_TIMEOUT: RPC call was terminated after RPC application timeout
0x46	APMR	0x00	Invalid state
		0x01	LMPM signaled an error
0x47	APMS	0x00	Invalid state
		0x01	LMPM signaled an error
		0x02	Timeout
0x48	CPM	0x00	Invalid state
0x49	PPM	0x00	Invalid state
0x4A	DCPUCS	0x00	Invalid state
		0x01	LMPM signaled an error
		0x02	Timeout
0x4B	DCPUCR	0x00	Invalid state
		0x01	LMPM signaled an error
0x4C	DCPMCS	0x00	Invalid state
		0x01	LMPM signaled an error
0x4D	DCPMCR	0x00	Invalid state
		0x01	LMPM signaled an error
0x4E	FSPM	0x00-0xFF	FAL Service Protocol Machine error
0x64	CTLSM	0x00	Invalid state
		0x01	CTLSM signaled error
0x65	CTLRDI	0x00	Invalid state
		0x01	CTLRDI signaled an error
0x66	CTLRDR	0x00	Invalid state
		0x01	CTLRDR signaled an error
0x67	CTLWRI	0x00	Invalid state
		0x01	CTLWRI signaled an error
0x68	CTLWRR	0x00	Invalid State
		0x01	CTLWRR signaled an error
0x69	CTLIO	0x00	Invalid state
		0x01	CTLIO signaled an error
0x6A	CTLSU	0x00	Invalid state
		0x01	AR add provider or consumer failed

© Hilscher, 2015–2017

ErrCode1	Description	ErrCode2	Description / Use
		0x02	AR alarm-open failed
		0x03	AR alarm-ack-send
		0x04	AR alarm-send
		0x05	AR alarm-ind
0x6B	CTLRPC	0x00	Invalid state
		0x01	CTLRPC signaled an error
0x6C	CTLPBE	0x00	Invalid state
		0x01	CTLPBE signaled an error
0xC8	CMSM	0x00	Invalid state
		0x01	CMSM signaled an error
0xCA	CMRDR	0x00	Invalid state
		0x01	CMRDR signaled an error
0xCC	CMWRR	0x00	Invalid state
		0x01	AR is not in state Primary. (Write not allowed)
		0x02	CMWRR signaled an error
0xCD	CMIO	0x00	Invalid state
		0x01	CMIO signaled an error
0xCE	CMSU	0x00	Invalid state
		0x01	AR add provider or consumer failed
		0x02	AR alarm-open failed
		0x03	AR alarm-send
		0x04	AR alam-ack-send
		0x05	AR alarm-ind
0xD0	CMINA	0x00	Invalid state
		0x01	CMINA signaled an error
0xD1	CMPBE	0x00	Invalid state
		0x01	CMPBE signaled an error
0xD2	CMDMC	0x00	Invalid state
		0x01	CMDMC signaled an error
0xFD	Used by RTA for protocol error (RTA_ERR_CLS_PROTOCOL)	0x00	Reserved
		0x01	error within the coordination of sequence numbers (RTA_ERR_CODE_SEQ)
		0x02	instance closed (RTA_ERR_ABORT)
		0x03	AR out of memory (RTA_ERR_ABORT)
		0x04	AR add provider or consumer failed (RTA_ERR_ABORT)
		0x05	AR consumer DHT / WDT expired (RTA_ERR_ABORT)
		0x06	AR cmi timeout (RTA_ERR_ABORT)
		0x07	AR alarm-open failed (RTA_ERR_ABORT)
		0x08	AR alarm-send.cnf(-) (RTA_ERR_ABORT)
		0x09	AR alarm-ack- send.cnf(-) (RTA_ERR_ABORT)

© Hilscher, 2015–2017

ErrCode1	Description	ErrCode2	Description / Use
		0x0A	AR alarm data too long (RTA_ERR_ABORT)
		0x0B	AR alarm.ind(err) (RTA_ERR_ABORT)
		0x0C	AR rpc-client call.cnf(-) (RTA_ERR_ABORT)
		0x0D	AR abort.req (RTA_ERR_ABORT)
		0x0E	AR re-run aborts existing (RTA_ERR_ABORT)
		0x0F	AR release.ind received (RTA_ERR_ABORT)
		0x10	AR device deactivated (RTA_ERR_ABORT)
		0x11	AR removed (RTA_ERR_ABORT)
		0x12	AR protocol violation (RTA_ERR_ABORT)
		0x13	AR name resolution error (RTA_ERR_ABORT)
		0x14	AR RPC-Bind error (RTA_ERR_ABORT)
		0x15	AR RPC-Connect error (RTA_ERR_ABORT)
		0x16	AR RPC-Read error (RTA_ERR_ABORT)
		0x17	AR RPC-Write error (RTA_ERR_ABORT)
		0x18	AR RPC-Control error (RTA_ERR_ABORT)
		0x19	AR forbidden pull or plug after check.rsp and before in- data.ind (RTA_ERR_ABORT)
		0x1A	AR AP removed (RTA_ERR_ABORT)
		0x1B	AR link down (RTA_ERR_ABORT)
		0x1C	AR could not register multicast-mac address (RTA_ERR_ABORT)
		0x1D	not synchronized (cannot start companion-ar) (RTA_ERR_ABORT)
		0x1E	wrong topology (cannot start companion-ar) (RTA_ERR_ABORT)
		0x1F	dcp, station-name changed (RTA_ERR_ABORT)
		0x20	dcp, reset to factory-settings (RTA_ERR_ABORT)
		0x21	cannot start companion-AR because a 0x8ipp submodule in the first AR... (RTA_ERR_ABORT)
		0x22	no irdata record yet (RTA_ERR_ABORT)
		0x23	PDEV (RTA_ERR0R_ABORT)
		0x24	PDEV, no port offers required speed / duplexity (RTA_ERR_ABORT)
		0x25	IP-suite [of the IOC] changed by means of DCP set(IPParameter) or local engineering (RTA_ERR_ABORT)
		0x26	IOCARSR RDHT expired (RTA_ERR0R_ABORT)
		0xC9	AR removed by reason of watchdog timeout in the Application task (RTA_ERR0R_ABORT)

© Hilscher, 2015–2017

ErrCode1	Description	ErrCode2	Description / Use
		0xCA	AR removed by reason of pool underflow in the Application task (RTA_ERROR_ABORT)
		0xCB	AR removed by reason of unsuccessful packet sending (Queue) inside OS in the Application task (RTA_ERROR_ABORT)
		0xCC	AR removed by reason of unsuccessful memory allocation in the Application task (RTA_ERROR_ABORT)
0xFF	User specific	0x00-0xFE	User specific
		0xFF	Recommended for “User abort” without further detail

Table 53: Coding of ErrorCode1 for ErrorDecode = PNIO (Excluding reserved Values)

7.3.5 ErrorCode1 and ErrorCode2 for ErrorDecode is Manufacturer Specific

If ErrorDecode is set to Manufacturer Specific, the values of the fields ErrorCode1 and ErrorCode2 can be freely chosen.

© Hilscher, 2015–2017

8 Appendix

8.1 List of tables

Table 1: List of revisions ... 4
Table 2: Technical data ... 7
Table 3: Valid values of the device handle ... 14
Table 4: Valid values of the IOCR handle ... 14
Table 5: Internally generated frame id values ... 14
Table 6: Valid values for a submodule handle ... 15
Table 7: Packet configuration sequence ... 16
Table 8: Definiton of OEM parameter types ... 28
Table 9: Definition of alarm handling flags ... 38
Table 10: Parameter usPrmType ... 44
Table 11: Known values for fiber optic cable type ... 106
Table 12: Known values for fiber optic type ... 107
Table 13: RCX_GET_DEVICE_INFO_CNF_T - Get User Parameter Data Request ... 173
Table 14: Device Handle for Device Access AR ... 174
Table 15: Device Access AR Services ... 174
Table 16: Get configuration services ... 190
Table 17: PNM_AP_CFG_GET_IOC_REQ_T request ... 192
Table 18: PNM_AP_CFG_GET_IOC_CNF_T confirmation ... 195
Table 19: Definition of alarm handling flags ... 196
Table 20: PNM_AP_CFG_GET_IOC_PRM_LIST_REQ_T request ... 199
Table 21. PNM_AP_CFG_GET_IOC_PRM_LIST_CNF_T confirmation ... 200
Table 22: PNM_AP_CFG_GET_IOC_PRM_REQ_T request ... 201
Table 23: PNM_AP_CFG_GET_IOC_PRM_CNF_T confirmation ... 203
Table 24: Parameter usPrmType ... 205
Table 25: PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_REQ_T request ... 206
Table 26: PNM_AP_CFG_GET_NUM_CONFIGURED_OBJECTS_CNF_T confirmation ... 207
Table 27: PNM_AP_CFG_GET_IOD_REQ_DATA_T request ... 209
Table 28: PNM_AP_CFG_GET_IOD_CNF_T confirmation ... 213
Table 29: Parameter bARType - Type of the AR ... 213
Table 30: Parameter bAddressMode - Modes of IP address resolution ... 213
Table 31: Parameter ulFlags - Flag bits specifying additional properties of the AR ... 214
Table 32: Parameter ulArProperties - Bitmask describing PROFINET Properties of the AR ... 214
Table 33: PNM_AP_CFG_GET_AR_PRM_LIST_REQ_T request ... 216
Table 34: PNM_AP_CFG_GET_AR_PRM_LIST_CNF_T confirmation ... 217
Table 35: PNM_AP_CFG_GET_AR_PRM_REQ_T request ... 218
Table 36: PNM_AP_CFG_GET_AR_PRM_CNF_T confirmation ... 221
Table 37: Parameters usIdentifier, uData ... 222
Table 38: Structure PNM_AP_CFG_AR_PRM_ARVENDORBLOCK_T ... 222
Table 39: PNM_AP_CFG_GET_IOCR_REQ_T request ... 223
Table 40: PNM_AP_CFG_GET_IOCR_CNF_T confirmation ... 227
Table 41: PNM_AP_CFG_GET_SUBMODULE_REQ_T request ... 231
Table 42: PNM_AP_CFG_GET_SUBMODULE_CNF_T confirmation ... 235
Table 43: PNM_AP_CFG_GET_RECORD_DATA_REQ_T request ... 238
Table 44: PNM_AP_CFG_GET_RECORD_DATA_CNF_T confirmation ... 240
Table 45: Services using packet fragmentation ... 268
Table 46 MRP parameter summary ... 271
Table 47: PROFINET IO Controller LED status (Loadable firmware) ... 273
Table 48: Status codes / Error codes: PROFINET IO Controller ... 278
Table 49: Status codes / Error codes: Sockets ... 279
Table 50: Coding of PNIO Status ErrorCode (Excluding reserved Values) ... 280
Table 51: Coding of PNIO status ErrorDecode (Excluding reserved Values) ... 280
Table 52: Coding of ErrorCode1 for ErrorDecode = PNIORW (Excluding reserved Values) ... 281
Table 53: Coding of ErrorCode1 for ErrorDecode = PNIO (Excluding reserved Values) ... 287

8.2 List of figures

Figure 1: Structure of configuration database ... 13
Figure 2: Timing diagram for output data ... 17
Figure 3: Sequence when using xChannelIOWrite ... 19
Figure 4: Timing diagram for input data ... 20
Figure 5: Sequence when using xChannelIORead ... 22
Figure 6: Process data timing ... 23

© Hilscher, 2015–2017

Figure 7 Controller Global/Network state 24
Figure 8: Definition of Send Clock Factor, Reduction Ration and Phase 62
Figure 9: Memory mapping of IOCR data (CSDU) to DPM input/output area 63
Figure 10: Mapping of submodule data to IOCRs and DPM areas 69
Figure 11: Network scan sequence 171
Figure 12: Device Access AR Sequence 175
Figure 13: Definition of Send Clock Factor, Reduction Ration and Phase 229
Figure 14: Memory mapping of IOCR data (CSDU) to DPM input/output area 230
Figure 15: Mapping of submodule data to IOCRs and DPM areas 237
Figure 16: Fragmented request/Confirmation sequence 267
Figure 17: Fragmented indication/response sequence 268
Figure 18: MRP single ring topology 270
Figure 19: Ring topology with two rings 270
Figure 20: Structure of the PROFINET Status Code 279

© Hilscher, 2015–2017

8.3 Contacts

Headquarters

Germany
Hilscher Gesellschaft für
Systemautomation mbH
Rheinstrasse 15
65795 Hattersheim
Phone: +49 (0) 6190 9907-0
Fax: +49 (0) 6190 9907-50
E-Mail: info@hilscher.com

Support
Phone: +49 (0) 6190 9907-99
E-Mail: de.support@hilscher.com

Subsidiaries

China
Hilscher Systemautomation (Shanghai) Co. Ltd.
200010 Shanghai
Phone: +86 (0) 21-6355-5161
E-Mail: info@hilscher.cn

Support
Phone: +86 (0) 21-6355-5161
E-Mail: cn.support@hilscher.com

France
Hilscher France S.a.r.l.
69500 Bron
Phone: +33 (0) 4 72 37 98 40
E-Mail: info@hilscher.fr

Support
Phone: +33 (0) 4 72 37 98 40
E-Mail: fr.support@hilscher.com

India
Hilscher India Pvt. Ltd.
Pune, Delhi, Mumbai
Phone: +91 8888 750 777
E-Mail: info@hilscher.in

Italy
Hilscher Italia S.r.l.
20090 Vimodrone (MI)
Phone: +39 02 25007068
E-Mail: info@hilscher.it

Support
Phone: +39 02 25007068
E-Mail: it.support@hilscher.com

Japan
Hilscher Japan KK
Tokyo, 160-0022
Phone: +81 (0) 3-5362-0521
E-Mail: info@hilscher.jp

Support
Phone: +81 (0) 3-5362-0521
E-Mail: jp.support@hilscher.com

Korea
Hilscher Korea Inc.
Seongnam, Gyeonggi, 463-400
Phone: +82 (0) 31-789-3715
E-Mail: info@hilscher.kr

Switzerland
Hilscher Swiss GmbH
4500 Solothurn
Phone: +41 (0) 32 623 6633
E-Mail: info@hilscher.ch

Support
Phone: +49 (0) 6190 9907-99
E-Mail: ch.support@hilscher.com

USA
Hilscher North America, Inc.
Lisle, IL 60532
Phone: +1 630-505-5301
E-Mail: info@hilscher.us

Support
Phone: +1 630-505-5301
E-Mail: us.support@hilscher.com

© Hilscher, 2015–2017