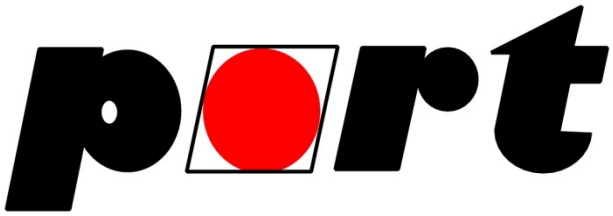
GOAL PROFINET
User Manual



GOAL PROFINET Version: 2.20.0

Contents

List of Tables

1 Introduction

The unified GOAL PROFINET layer provides a common interface to all adapted PROFINET stacks by using the same API in single core mode and also via GOALs Core To Core for multiple cores.

1.1 Application description

The GOAL PROFINET stack comes with ready to run examples in the folder appl. The matching GSDML file can be found in the path protos/pnio/gsdml.

1.1.1 appl/goal_pnio/00_rpc_cc (only available for Core To Core)

This application must be run on the communication core of a device. It provides the functionality for all other examples to the application core. It is not restricted to the examples but also provides the full GOAL PROFINET functionality to the user application

1.1.2 appl/goal_pnio/01_simple_io

The example 01_simple_io is setup with two modules. First module is a 64 byte input module, second module is 64 byte output. It copies the received data from the output module to the input module every time the integrated cnt variable reaches a multiple of 1000. The counting speed depends heavily on the used platform. The device setup can edit by changing the goal_pnioSubmodPlug lines in the function appl_setup in the file goal_appl.c.

1.1.3 appl/goal_pnio/02_io_demo

This example is an enhanced version of the 01_simple_io example that has two counters and includes LED and button mapping if they are available on the selected platform. Slot 1 contains a 64 byte input module, slot 2 contains a 64 byte output module. To match the configuration of the PLC the following tables show the inner values of the data stream. The LEDs, buttons and counters are in big endian format.

Table 1: Input data - 64 bytes

Bytes	Function
0-3	Button state.
4-7	Mirror output data bytes 4 - 7.
8 - 11	32-bit counter.
12 - 15	32-bit counter that only counts when at least 1 button is pressed.
16 - 63	Bit shift from bit 0 (at byte 16) to bit 7 (at byte 63).

Table 2: Output data - 64 bytes

Bytes	Function
0-3	Set LED state.
4-7	Data that is mirrored to input data bytes 4 - 7.
8 - 63	Unused.

1.1.4 appl/goal_pnio/03_record_write

This example is configured with the 64 byte input and the 64 byte output module. Whenever a record write request is received and can't be handled by the PROFINET Stack itself, the callback GOAL_PNIO_CB_ID_WRITE_RECORD is triggered. In this example it calls the function processWriteRequest which shows some information about the request and if the record data length defined in the define RECORD_DATA_LENGTH matches, it returns GOAL_OK_SUPPORTED. Otherwise the request will fail with GOAL_ERR_NOT_FOUND.

1.1.5 appl/goal_pnio/05_ioxs_states

This example is configured with the 64 byte input and the 64 byte output module. It demonstrates how to read the IOCS state and provides a function called show_io_state which shows the state as human readable string.

1.1.6 appl/goal_pnio/06_apdu_status

This example is configured with the 64 byte input and the 64 byte output module. It demonstrates how to read the overall frame status, called GOAL_PNIO_APDU_STATUS_T. This contains the cycle counter, the data status and the transfer status.

1.1.7 appl/goal_pnio/07_alarm_button

This example is configured with the 64 byte input and the 64 byte output module. It demonstrates how to send and receive AlarmNotification alarms to and from the PLC and also shows how to read AlarmNotification ACKs. In this example an alarm is triggered by pressing a button which only works if your platform supports it.

1.1.8 appl/goal_pnio/08_dynamic_modules

This example demonstrates how to handle the dynamic module configuration when a Connect Request from the PLC is received. It connects the two callbacks GOAL_PNIO_CB_ID_CONNECT_REQUEST_EXP_START and GOAL_PNIO_CB_ID_EXP_SUBM

The first callback GOAL_PNIO_CB_ID_CONNECT_REQUEST_EXP_START removes all modules from all slots but the DAP slot. The second callback GOAL_PNIO_CB_ID_EXP_SUBMOD then tries to plug all modules/submodules from the ExpectedSubmoduleBlock. Only the DAP is left out because in this example the DAP configuration is hardwired. If a module/submodule combination isn't found then the plug will fail and give an error message. After all ExpectedSubmoduleBlock requests are handled, the internal stack logic will check if all slots are match the requested setup from the PLC and if not, it will create a ModuleDiffBlock in the response, saying what modules/submodules aren't matching the configuration. Then the PLC can decide whether to accept this or to cancel the connection.

1.1.9 appl/goal_pnio/09_busy_records

This example shows how to handle a record read and write request if the result for the caller isn't available in the callback. For example when a write request needs some time to be stored to the non volatile storage and it is necessary to check if the write was ok then the busy handling functionality must be used. To demonstrate the functionality, configure the modules as following: * Slot 1: Module 0x33, Submodule 0x01 * Slot 2: Module 0x31, Submodule 0x01 During the Connect Request the PLC will write the parameters of module 0x33 which you should see in the debug output. Each time a write is requested the application will store it and wait for 10 seconds. After this time the write request is answered with a positive response status.

1.1.10 appl/goal_pnio/10_led_demo

This example shows how to take care about the connection and DCP signal LED. It works only for architectures that support the generic OAL_setLeds implementation. In the example the callbacks GOAL_PNIO_CB_ID_APPL_READY and GOAL_PNIO_CB_ID_RELEASE_AR toggle an LED showing the connection state and the callback GOAL_PNIO_CB_ID_BLINK toggles an LED for the DCP state.

1.1.11 appl/goal_pnio/11_multiple_write

This example shows how to handle a multiple write request (index 0xe040) with busy mode. For all sub- requests that are not handled within the stack, the callback GOAL_PNIO_CB_ID_WRITE_RECORD is called. It is possible that the successive calls of the callback are faster than their processing. Therefore, the callback data are stored and answered later. If the application returns the value GOAL_OK_SUPPORTED from the callback the multiple write response won't be sent until the function goal_pnioRecWriteFinish is called and processed.

1.1.12 appl/goal_pnio/12_diag_entry

This example shows how to set and clear a diagnostic entry. In the GSDML you can add a free text with up to four placeholders for dynamic values. The parameter errorNumber of the function goal_pnioDiagExtChanDiagAdd sets this values. The configured text is readable in the diagnosis buffer of the controller.

1.1.13 appl/goal_pnio/13_pnio_snmp

This example shows how to switch from PROFINET CC-A to PROFINET CC-B by adding SNMP support. Additionally, it periodically copies data from the output to the input module and supports the DCP blink command.

1.1.14 appl/goal_pnio/14_info_set

This example shows how to update the device identification. In the function appl_setup it is possible to set different device information, like e.g. vendor Id, device ID or vendor name.

1.1.15 appl/goal_pnio/15_config_set

This example is similar to the previous example. The difference is, that the values are written in variables of the goal configuration manager. The example application shows how to set the most common PROFINET configuration variables e.g. vendor id, vendor name, hardware revision, I&M tag function etc. In the function appl_setup it is possible to set these PROFINET configuration values before a new PROFINET instance is created.

1.1.16 appl/goal_pnio/16_device_name

This example shows how to update the device name from the application. Be aware that this is not specification conform and must only be used (if necessary) during development.

1.1.17 appl/goal_pnio/17_process_alert

This example shows how to send an process alarm cyclically. This transfers additional user specific data to the controller. In the appl_setup function, the total length of the alarm payload is adjusted before a new PROFINET instance is created.

1.1.18 appl/goal_pnio/18_dyn_mod_postpone

This example shows how to postpone the processing of a requested module configuration to a later time. The processing can be resumed when the final configuration is determined.

1.1.19 appl/goal_pnio/19_subst_val

This example shows the impact of setting a SubstituteValue by the PLC and the stack reaction for different APDU and IOPS states.

- PLC Configuration:
  - Slot 1: “64 bytes Input”
  - Slot 2: “64 bytes Output”

Example output:

```
[I|RPC_handleReq:663] connect request: 192.168.0.100
[I|AR_createAR:339] new AR [0]: 83ebcc3de2a148f291141fc0e27dab6e
[I|CD_openEP:439] input EP period: 1 ms (scf: 32, rr: 1, ph: 1)
[I|PN_alarmOpenEp:317] Alarm EP: local: 0, remote: 44, timeout: 100 ms,
                       retries: 3, remote: da:72:53:61:3f:1d
[I|PN_recPDPortDataParamStart:2451] cleared PDPortData entries
[I|CD_activateEP:622] output EP period: 1 ms, timeout: 3 ms
[I|PN_recWrSubstVal:4778] SubstitutionValue for 0x0:0x2:0x1:
                          replacement value
[I|RPC_checkAutoAnswers:1663] Sending application ready
[I|appl_loop:225] APDU_Status.DataStatus.State: IOCR state is primary
[I|appl_loop:228] APDU_Status.DataStatus.Redundancy: (default) one
                  primary AR of a given AR-set is present
[I|appl_loop:235] APDU_Status.DataStatus.DataValid: DataItem valid
[I|appl_loop:239] APDU_Status.DataStatus.ProviderState: run
[I|appl_loop:243] APDU_Status.DataStatus.StationProblemIndicator:
                  normal operation
[I|appl_loop:247] APDU_Status.DataStatus.Ignore: evaluate the DataStatus
[I|appl_loop:266] output IOPS changed: 0x80
[I|appl_loop:272] hex data: 00 00 00 00 00 00 00 00
[I|appl_loop:266] output IOPS changed: 0x40
[I|appl_loop:272] hex data: 52 65 70 6c 61 63 65 6d
[I|appl_loop:239] APDU_Status.DataStatus.ProviderState: stop
[I|appl_loop:266] output IOPS changed: 0x80
[I|appl_loop:272] hex data: 52 65 70 6c 61 63 65 6d
[I|appl_loop:239] APDU_Status.DataStatus.ProviderState: run
[I|appl_loop:272] hex data: 00 00 00 00 00 00 00 00
[I|appl_loop:266] output IOPS changed: 0x40
[I|appl_loop:272] hex data: 52 65 70 6c 61 63 65 6d
[I|appl_loop:239] APDU_Status.DataStatus.ProviderState: stop
[I|appl_loop:266] output IOPS changed: 0x80
[I|appl_loop:272] hex data: 52 65 70 6c 61 63 65 6d
[I|appl_loop:239] APDU_Status.DataStatus.ProviderState: run
[I|appl_loop:272] hex data: 00 00 00 00 00 00 00 00
```

1.1.20 appl/goal_pnio/20_subst_mod

This example allows to plug two different modules into slot 2. If the “64 bytes Output” module is plugged everything is ok and the PLC receives no ModuleDiffBlock. As alternative the module “64 bytes Input/Output” can be plugged in slot 2 which the application will also accept but react with the submodule state “substitute” in the generated ModuleDiffBlock.

- PLC Configuration:
  - Slot 1: “64 bytes Input”
  - Slot 2: “64 bytes Output” or “64 bytes Input/Output”

Example output for module “64 bytes Output” in slot 2:

```
[I|main_modulePlug:232] plugged module (0x00000030, 0x00000001)
                        into slot (0, 1, 1)
[I|main_modulePlug:257] plugged module (0x00000031, 0x00000001)
                        into slot (0, 2, 1) - state: ok
```

Example output for module “64 bytes Input/Output” in slot 2:

```
[I|main_modulePlug:232] plugged module (0x00000030, 0x00000001)
                        into slot (0, 1, 1)
[I|main_modulePlug:271] plugged module (0x00000032, 0x00000001)
                        into slot (0, 2, 1) - state: subst
```

2 Initialization

GOAL PROFINET has two fixed steps to create a running instance. These are shown in the following example code. First goal_pnioInit must be called to register the GOAL PROFINET stack in GOAL. After that an instance can be created with goal_pnioNew. The GOAL PROFINET stack configuration is done by two configuration steps. The static configuration like how much slot memory should be reserved and which vendor id should be used must be done between goal_pnioInit and goal_pnioNew. This API starts with goal_pnioCfg. After goal_pnioNew all other APIs can be used like creating a slot or a module.

In GOAL this functionality matches the appl_* pattern. The function goal_pnioInit must be called from appl_init whereas the later init functionality like goal_pnioCfg, goal_pnioNew and setting up the device structure must be put into appl_setup. After the GOAL initialization has finished the data handling has to be done in appl_loop or by registering an own main loop handler in GOAL. Also using an external thread is possible as most of the GOAL PROFINET functionality is thread-safe.

```
static GOAL_PNIO_T *pPnio;                          /**< GOAL PROFINET handle */

GOAL_STATUS_T appl_init(
    void
)
{
    GOAL_STATUS_T res;                              /* GOAL result */

    /* initialize GOAL PROFINET */
    res = goal_pnioInit();
    if (GOAL_RES_ERR(res)) {
        goal_logErr("failed to initialize GOAL PROFINET");
    }

    return res;
}

GOAL_STATUS_T appl_setup(
    void
)
{
    GOAL_STATUS_T res;                              /* GOAL result */

    /* set default values for all GOAL PROFINET instances */
    /* (these can be overwritten as often as needed, as each instance
       creates a copy) */

    /* vendor id = 0x1234 */
```

```
    res = goal_pnioCfgVendorIdSet(0x1234);
    if (GOAL_RES_ERR(res)) {
        goal_logErr("failed to set the GOAL PROFINET vendor id");
        return res;
    }

    /* create a new GOAL PROFINET instance */
    res = goal_pnioNew(&pPnio,                  /* GOAL PROFINET instance */
                       0,                       /* GOAL PROFINET instance id */
                       appl_pnioCb              /* GOAL PROFINET callback */
                      );
    if (GOAL_RES_ERR(res)) {
        goal_logErr("failed to create a new PROFINET instance");
        return res;
    }

    /* create slots, subslots, modules, submodules, link them together ... */
}

void appl_loop(
    void
)
{
    /* check if a GOAL PROFINET connection is established (see callback API) */

    /* handle module data */
}
```

3 Application Programming Interface

This chapter lists the API functions that are provided by GOAL PROFINET.

3.1 goal_pnioInit - Register GOAL PROFINET in GOAL (appl_init)

This function registers GOAL PROFINET in GOAL and must be called in appl_init. Its main functionality is to register a set config manager variables before the NVS storage is initialized.

It returns a GOAL_STATUS_T status and has no parameters.

```
GOAL_STATUS_T appl_init(
    void
)
{
    GOAL_STATUS_T res;                              /**< GOAL result */

    /* initialize GOAL PROFINET */
    res = goal_pnioInit();
    if (GOAL_RES_ERR(res)) {
        goal_logErr("failed to initialize GOAL PROFINET");
    }

    return res;
}
```

3.2 goal_pnioNew - Create a GOAL PROFINET instance (appl_setup)

This function creates a new GOAL PROFINET instance by taking a snapshot of all pre-variables (goal_pnioCfg API) and allocating the necessary resources. The application has to provide a GOAL PROFINET instance id and a callback handler. The callback handler can also be NULL to only use GOAL PROFINET stack default behavior.

It returns a GOAL_STATUS_T status and a GOAL PROFINET instance handle.

Table 3: goal_pnioNew parameters

Parameter	Description
GOAL_PNIO_T **ppPnio	returned GOAL PROFINET instance handle
const uint32_t id	GOAL PROFINET instance id
GOAL_PNIO_FUNC_CB_T pFunc	GOAL PROFINET callback function

See example 01_simple_io for a demonstration.

3.3 goal_pnioCfgDcpFactoryResetDisableSet - Configure DCP Factory Reset

Controls if DCP factory reset is available. If set to GOAL_TRUE DCP factory reset will be denied. Default: GOAL_FALSE.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 4: goal_pnioCfgDcpFactoryResetDisableSet parameters

Parameter	Description
GOAL_BOOL_T flgDcpFactoryResetDisable	DCP Factory Reset Disable Flag

See example 15_config_set for a demonstration.

3.4 goal_pnioCfgDcpAcceptMixcaseStationSet - Configure DCP Mixcase Stationname Acceptance

Controls if DCP accepts station names with mixed case spelling. If set to GOAL_TRUE DCP accepts mixed case station names. Default: GOAL_FALSE.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 5: goal_pnioCfgDcpAcceptMixcaseStationSet parameters

Parameter	Description
GOAL_BOOL_T flgDcpAcceptMixcaseStation	DCP Mixcase Stationname Acceptance Flag

See example 15_config_set for a demonstration.

3.5 goal_pnioCfgVendorIdSet - Set Vendor Id

Configures the vendor id. Default: 0x028c (port GmbH)

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 6: goal_pnioCfgVendorIdSet parameters

Parameter	Description
uint16_t idVendor	Vendor ID

See example 15_config_set for a demonstration.

3.6 goal_pnioCfgDeviceIdSet - Set Device Id

Configures the device id. Default: 0x0001

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 7: goal_pnioCfgDeviceIdSet parameters

Parameter	Description
uint16_t idDevice	Device ID

See example 15_config_set for a demonstration.

3.7 goal_pnioCfgVendorNameSet - Set Vendor Name

Configures the vendor name. Default: “port GmbH”

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 8: goal_pnioCfgVendorIdSet parameters

Parameter	Description
const char *strVendor	Vendor Name

See example 15_config_set for a demonstration.

3.8 goal_pnioCfgPortDescSet - Set LLDP Port Description

Configures the LLDP port description. Default: “TestPort”

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 9: goal_pnioCfgPortDescSet parameters

Parameter	Description
const char *strDescPort	Port Description

See example 15_config_set for a demonstration.

3.9 goal_pnioCfgSystemDescSet - Set LLDP System Description

Configures the LLDP system description. Default: “PROFINET System”

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 10: goal_pnioCfgSystemDescSet parameters

Parameter	Description
const char *strSystem	System Description

See example 15_config_set for a demonstration.

3.10 goal_pnioCfgOrderIdSet - Set Order Id

Configures the order id. Default: “00210”

See document “Profile Guidelines Part 1: Identification & Maintenance Functions” for details.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 11: goal_pnioCfgOrderIdSet parameters

Parameter	Description
const char *strOrder	Order Id

See example 15_config_set for a demonstration.

3.11 goal_pnioCfgSerialNumSet - Set Serial Number

Configures the serial number. Default: “20074”

See document “Profile Guidelines Part 1: Identification & Maintenance Functions” for details.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 12: goal_pnioCfgSerialNumSet parameters

Parameter	Description
const char *strNumSerial	Serial Number

See example 15_config_set for a demonstration.

3.12 goal_pnioCfgHwRevSet - Set Hardware Revision

Configures the hardware revision. Default: 1

See document “Profile Guidelines Part 1: Identification & Maintenance Functions” for details.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 13: goal_pnioCfgHwRevSet parameters

Parameter	Description
uint16_t idRevHw	Hardware Revision

See example 15_config_set for a demonstration.

3.13 goal_pnioCfgSwRevPrefixSet - Set Software Revision Prefix

Configures the software revision prefix. Default: “P”

See document “Profile Guidelines Part 1: Identification & Maintenance Functions” for details.

Valid values are:

- “V” - official
- “R” - revision
- “P” - prototype
- “U” - under test
- “T” - test device

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 14: goal_pnioCfgSwRevPrefixSet parameters

Parameter	Description
const char chrRevSwPrefix	Software Revision Prefix

See example 15_config_set for a demonstration.

3.14 goal_pnioCfgSwRevFuncEnhSet - Set Software Revision Functional Enhancement

Configures the software revision functional enhancement. Default: 0x50

See document “Profile Guidelines Part 1: Identification & Maintenance Functions” for details.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 15: goal_pnioCfgSwRevFuncEnhSet parameters

Parameter	Description
uint8_t idRevSwFuncEnh	Software Revision Functional Enhancement

See example 15_config_set for a demonstration.

3.15 goal_pnioCfgSwRevBugfixSet - Set Software Revision Bugfix

Configures the software revision bugfix. Default: 3

See document “Profile Guidelines Part 1: Identification & Maintenance Functions” for details.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 16: goal_pnioCfgSwRevBugfixSet parameters

Parameter	Description
uint8_t idRevSwBugfix	Software Revision Bugfix

See example 15_config_set for a demonstration.

3.16 goal_pnioCfgSwRevIntChgSet - Set Software Revision Internal Change

Configures the software revision internal change. Default: 0x18

See document “Profile Guidelines Part 1: Identification & Maintenance Functions” for details.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 17: goal_pnioCfgSwRevIntChgSet parameters

Parameter	Description
uint8_t idRevSwIntChg	Software Revision Internal Change

See example 15_config_set for a demonstration.

3.17 goal_pnioCfgSwRevCntSet - Set Software Revision Counter

Configures the software revision counter. Default: 0x0000

See document “Profile Guidelines Part 1: Identification & Maintenance Functions” for details.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 18: goal_pnioCfgSwRevCntSet parameters

Parameter	Description
uint16_t idRevSwRevCnt	Software Revision Counter

See example 15_config_set for a demonstration.

3.18 goal_pnioCfgIm1TagFuncSet - Set I&M1 Tag Function

Configures the I&M1 tag function. Default: “”

See document “Profile Guidelines Part 1: Identification & Maintenance Functions” for details.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 19: goal_pnioCfgIm1TagFuncSet parameters

Parameter	Description
const char *strIm1TagFunc	I&M1 Tag Function

See example 15_config_set for a demonstration.

3.19 goal_pnioCfgIm1TagLocSet - Set I&M1 Tag Location

Configures the I&M1 tag location. Default: “”

See document “Profile Guidelines Part 1: Identification & Maintenance Functions” for details.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 20: goal_pnioCfgIm1TagLocSet parameters

Parameter	Description
const char *strIm1TagLoc	I&M1 Tag Location

See example 15_config_set for a demonstration.

3.20 goal_pnioCfgIm2DateSet - Set I&M2 Date

Configures the I&M2 date. Default: “”

See document “Profile Guidelines Part 1: Identification & Maintenance Functions” for details.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 21: goal_pnioCfgIm2DateSet parameters

Parameter	Description
const char *strIm2Date	I&M2 Date

See example 15_config_set for a demonstration.

3.21 goal_pnioCfgIm3DescSet - Set I&M3 Description

Configures the I&M3 description. Default: “”

See document “Profile Guidelines Part 1: Identification & Maintenance Functions” for details.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 22: goal_pnioCfgIm3DescSet parameters

Parameter	Description
const char *strIm3Desc	I&M3 Description

See example 15_config_set for a demonstration.

3.22 goal_pnioCfgIm4SigSet - Set I&M4 Signature (Functional Safety)

Configures the I&M4 signature. Default: “”

See document “Profile Guidelines Part 1: Identification & Maintenance Functions” for details.

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 23: goal_pnioCfgIm4SigSet parameters

Parameter	Description
const char *strIm4Sig	I&M4 Signature

See example 15_config_set for a demonstration.

3.23 goal_pnioCfgDevDapSimpleSet - Configure Device DAP Simple Mode

Configures the device DAP simple mode. If flgDevDapSimple is GOAL_TRUE the GOAL PROFINET stack automatically creates the DAP and port slots and modules. Default: GOAL_TRUE

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 24: goal_pnioCfgDevDapSimpleSet parameters

Parameter	Description
GOAL_BOOL_T flgDevDapSimple	Device DAP Simple Mode Flag

See example 15_config_set for a demonstration.

3.24 goal_pnioCfgDevDapApiSet - Set Device DAP API Number

Configures the device DAP API number. Default: 0

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 25: goal_pnioCfgDevDapApiSet parameters

Parameter	Description
uint32_t idDevDapApi	Device DAP API Number

See example 15_config_set for a demonstration.

3.25 goal_pnioCfgDevDapSlotSet - Set Device DAP Slot Number

Configures the device DAP slot number. Default: 0

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 26: goal_pnioCfgDevDapSlotSet parameters

Parameter	Description
uint16_t idDevDapSlot	Device DAP Slot Number

See example 15_config_set for a demonstration.

3.26 goal_pnioCfgDevDapSubslotSet - Set Device DAP Subslot Number

Configures the device DAP subslot number. Default: 1

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 27: goal_pnioCfgDevDapSubslotSet parameters

Parameter	Description
uint16_t idDevDapSubslot	Device DAP Subslot Number

See example 15_config_set for a demonstration.

3.27 goal_pnioCfgDevDapModuleSet - Set Device DAP Module Id

Configures the device DAP module id. Default: 1

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 28: goal_pnioCfgDevDapModuleSet parameters

Parameter	Description
uint32_t idDevDapMod	Device DAP Module Id

See example 15_config_set for a demonstration.

3.28 goal_pnioCfgDevDapSubmoduleSet - Set Device DAP Submodule Id

Configures the device DAP submodule id. Default: 1

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 29: goal_pnioCfgDevDapSubmoduleSet parameters

Parameter	Description
uint32_t idDevDapSubmod	Device DAP Submodule Id

See example 15_config_set for a demonstration.

3.29 goal_pnioCfgNetLinkSafetySet - Configure Device Port Disable Behavior

Configures the device port disable behavior. If flgNetLinkSafety is set to GOAL_TRUE the GOAL PROFINET stack doesn't allow the master to disable all Ethernet interfaces at the same time. Default: GOAL_TRUE

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 30: goal_pnioCfgNetLinkSafetySet parameters

Parameter	Description
GOAL_BOOL_T flgNetLinkSafety	Device Port Disable Behavior

See example 15_config_set for a demonstration.

3.30 goal_pnioCfgNewIoDataCbSet - Configure New IO Data Callback

Configures the new IO data callback. If flgCbNewIoData is set to GOAL_TRUE the registered callback function is also called for each arrived cyclic frame. It must return as fast as possible as this happens in real time context. Default: GOAL_FALSE

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: If enabled it produces a very high application load as the callback is called 1000 times a second if the cycle time is set to 1 ms.

Table 31: goal_pnioCfgNewIoDataCbSet parameters

Parameter	Description
GOAL_BOOL_T flgCbNewIoData	New IO Data Callback Flag

See example 15_config_set for a demonstration.

3.31 goal_pnioCfgLldpOrgExtSet - Configure LLDP Organizationally-specific Extension

Configures the LLDP organizationally-specific extension. Default: GOAL_TRUE

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 32: goal_pnioCfgLldpOrgExtSet parameters

Parameter	Description
GOAL_BOOL_T flgLldpOrgExt	LLDP Organizationally-specific Extension Flag

See example 15_config_set for a demonstration.

3.32 goal_pnioCfgLldpOptTlvSet - Configure LLDP Optional TLV Parameters

Configures the LLDP optional TLV parameters. Default: GOAL_TRUE

These parameters contain:

- port description
- system name
- system description
- system capabilities
- management address
- object ID

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 33: goal_pnioCfgLldpOptTlvSet parameters

Parameter	Description
GOAL_BOOL_T flgLldpOptTlv	LLDP Optional TLV Parameters Flag

See example 15_config_set for a demonstration.

3.33 goal_pnioCfgLldpGenMacSet - Configure LLDP Port MAC Address Generation

Configures the automatic LLDP port MAC address generation. If set to GOAL_TRUE the LLDP port-specific MAC addresses are automatically generated by adding the port id to the host port MAC address. Default: GOAL_TRUE

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: If this functionality is enabled (default) every device uses at least two MAC addresses (real plus one per port) in an ascending order. To use specific virtual port MAC addresses this feature must be disabled and the driver must provide different MAC addresses per request (GOAL_ETH_PORT_HOST and port specific requests).

Table 34: goal_pnioCfgLldpGenMacSet parameters

Parameter	Description
GOAL_BOOL_T flgLldpGenMac	Automatic LLDP MAC Generation Flag

See example 15_config_set for a demonstration.

3.34 goal_pnioCfgIm14SupportSet - Configure I&M 1-4 Support

Configures the GOAL PROFINET stack I&M 1-4 support. If set to GOAL_FALSE the stack denies I&M 1-4 requests. Default: GOAL_TRUE

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 35: goal_pnioCfgIm14SupportSet parameters

Parameter	Description
GOAL_BOOL_T flgIm14Support	I&M 1-4 Support Flag

See example 15_config_set for a demonstration.

3.35 goal_pnioCfgIm14CbSet - Configure I&M 1-4 Callback

Configures the GOAL PROFINET stack I&M 1-4 callback. If set to GOAL_TRUE the application callback must handle the I&M 1-4 get and set requests. Default: GOAL_FALSE

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 36: goal_pnioCfgIm14CbSet parameters

Parameter	Description
GOAL_BOOL_T flgIm14Cb	I&M 1-4 Callback Flag

See example 15_config_set for a demonstration.

3.36 goal_pnioCfgIm0CbSet - Configure I&M 0 Callback

Configures the GOAL PROFINET stack I&M 0 callback. If set to GOAL_TRUE the application callback must handle the I&M 0 get and set requests. Default: GOAL_FALSE

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 37: goal_pnioCfgIm0CbSet parameters

Parameter	Description
GOAL_BOOL_T flgIm0Cb	I&M 0 Callback Flag

See example 15_config_set for a demonstration.

3.37 goal_pnioCfgIm0FilterDataCbSet - Configure I&M 0 Filter Data Callback

Configures the GOAL PROFINET stack I&M 0 filter data callback. If set to GOAL_TRUE the application callback must handle the I&M 0 filter data get and set requests. Default: GOAL_FALSE

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 38: goal_pnioCfgIm0FilterDataCbSet parameters

Parameter	Description
GOAL_BOOL_T flgIm0FilterCb	I&M 0 Filter Data Callback Flag

See example 15_config_set for a demonstration.

3.38 goal_pnioCfgRecDataBusyBufsizeSet - Configure Record Handle Storage Count

Configures the count of parallel record handles. Default: 2

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 39: goal_pnioCfgRecDataBusyBufsizeSet parameters

Parameter	Description
GOAL_BOOL_T cntRecDataBusyBufsize	Record Handle Storage Count

See example 15_config_set for a demonstration.

3.39 goal_pnioCfgRpcFragReqLenMaxSet - Configure Maximum Record Size

Configures the maximum size in bytes of a record request. This must match the MaxSupportedRecordSize attribute in the GSDML file. Default: 4068

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: Changing this value to a smaller number can break conformity.

Table 40: goal_pnioCfgRpcFragReqLenMaxSet parameters

Parameter	Description
unsigned int sizeRpcFragMaxReqLen	Maximum Record Size

See example 15_config_set for a demonstration.

3.40 goal_pnioCfgRpcFragMaxCntSet - Configure Maximum RPC Fragment Number

This function is obsolete. The GOAL PROFINET stack now allows 64 fragmented frames.

3.41 goal_pnioCfgRpcFragEnableSet - Configure RPC Fragmentation

Configures the RPC fragmentation feature. If set to GOAL_TRUE RPC fragmentation is enabled. Default: GOAL_TRUE

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 41: goal_pnioCfgRpcFragEnableSet parameters

Parameter	Description
GOAL_BOOL_T flgRpcFragSupport	RPC Fragmentation Enable Flag

See example 15_config_set for a demonstration.

3.42 goal_pnioCfgRpcSessionMaxCntSet - Configure Maximum RPC Session Count

Configures the maximum count of RPC sessions. Default: 8

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 42: goal_pnioCfgRpcSessionMaxCntSet parameters

Parameter	Description
unsigned int numRpcSessions	RPC Session Count

See example 15_config_set for a demonstration.

3.43 goal_pnioCfgDiagBufMaxCntSet - Configure Maximum Diagnosis Entries

Configures the maximum count of diagnosis entries. Default: 20

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: Using a too small value can break conformity.

Table 43: goal_pnioCfgDiagBufMaxCntSet parameters

Parameter	Description
unsigned int numDiagBufMax	Maximum Diagnosis Entries

See example 15_config_set for a demonstration.

3.44 goal_pnioCfgDiagBufMaxDataSizeSet - Configure Maximum Diagnosis Data Size

Configures the maximum size in bytes of each diagnosis entry. Default: 28

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: Using a value below 28 can break conformity.

Table 44: goal_pnioCfgDiagBufMaxDataSizeSet parameters

Parameter	Description
unsigned int numDiagDataSizeMax	Maximum Diagnosis Data Size

See example 15_config_set for a demonstration.

3.45 goal_pnioCfgIocrBlocksMaxSet - Configure Maximum IOCR Block Buffers

Configures the maximum count of IOCR block buffers. Default: 10

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 45: goal_pnioCfgIocrBlocksMaxSet parameters

Parameter	Description
unsigned int numIocrBlocksMax	Maximum IOCR Block Buffers

See example 15_config_set for a demonstration.

3.46 goal_pnioCfgCrMaxCntSet - Configure Maximum Communication Relation Count

Configures the maximum count of communication relations. Default: 10

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 46: goal_pnioCfgCrMaxCntSet parameters

Parameter	Description
unsigned int numCrMax	Maximum CR Count

See example 15_config_set for a demonstration.

3.47 goal_pnioCfgArMaxCntSet - Configure Maximum Application Relation Count

Configures the maximum count of application relations. Default: 2

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 47: goal_pnioCfgArMaxCntSet parameters

Parameter	Description
unsigned int numArMax	Maximum AR Count

See example 15_config_set for a demonstration.

3.48 goal_pnioCfgApiMaxCntSet - Configure Maximum API Count

Configures the maximum count of application process identifiers. Setting this value only affects the ModuleDiffBlock logic and needs to be big enough to hold the largest possible GSDML configuration. Default: 1

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity and the GOAL PROFINET stack only supports API 0..

Table 48: goal_pnioCfgApiMaxCntSet parameters

Parameter	Description
unsigned int numApiMax	Maximum API Count

See example 15_config_set for a demonstration.

3.49 goal_pnioCfgSlotMaxCntSet - Configure Maximum Slot Count

Configures the maximum count of slots. Setting this value only affects the ModuleDiffBlock logic and needs to be big enough to hold the largest possible GSDML configuration. Default: 3

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 49: goal_pnioCfgSlotMaxCntSet parameters

Parameter	Description
unsigned int numSlotMax	Maximum Slot Count

See example 15_config_set for a demonstration.

3.50 goal_pnioCfgSubslotMaxCntSet - Configure Maximum Subslot Count

Configures the maximum count of subslots per slot. Setting this value only affects the ModuleDiffBlock logic and needs to be big enough to hold the largest possible GSDML configuration. Default: 6

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 50: goal_pnioCfgSubslotMaxCntSet parameters

Parameter	Description
unsigned int numSubslotMax	Maximum Subslot Count Per Slot

See example 15_config_set for a demonstration.

3.51 goal_pnioCfgSubslotIfSet - Configure Interface Subslot

Configures the interface subslot id. Default: 0x8000

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 51: goal_pnioCfgSubslotIfSet parameters

Parameter	Description
unsigned int idSubslotIf	Interface Subslot Id

See example 15_config_set for a demonstration.

3.52 goal_pnioCfgSubslotPortSet - Configure Port Subslot

Configures the port subslot base id. Default: 0x8001

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Warning: This function should not be used as the behavior can break conformity.

Table 52: goal_pnioCfgSubslotPortSet parameters

Parameter	Description
unsigned int idSubslotPort	Port Subslot Id

See example 15_config_set for a demonstration.

3.53 goal_pnioCfgSnmpIdSet - Configure SNMP Instance Id

Configures the GOAL SNMP instance id that is used by the GOAL PROFINET stack. Default: GOAL_ID_INVALID

It returns a GOAL_STATUS_T status.

This function configures the GOAL PROFINET instance and must be called before goal_pnioNew to have an effect.

Table 53: goal_pnioCfgSnmpIdSet parameters

Parameter	Description
idSnmp	GOAL SNMP Instance Id

See example 13_pnio_snmp for a demonstration.

3.54 goal_pnioSlotNew - Create a new slot

Create a new slot. Returns a GOAL_STATUS_T status.

Table 54: goal_pnioSlotNew parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint32_t idApi	API number (always 0)
uint32_t idSlot	slot number
GOAL_BOOL_T flgAutoGen	generate API automatically if not exist

```
/* create API 0:Slot 1 even if API 0 doesn't exist */
res = goal_pnioSlotNew(pPnio, 0, 1, GOAL_TRUE);
```

3.55 goal_pnioSubslotNew - Create a new subslot

Create a new subslot. Returns a GOAL_STATUS_T status.

Table 55: goal_pnioSubslotNew parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint32_t idApi	API number (always 0)
uint32_t idSlot	slot number
uint32_t idSubslot	subslot number
GOAL_BOOL_T flgAutoGen	generate API and slot automatically if not exist

```
/* create API 0:Slot 1:Subslot 1 even if API 0 and/or
 * Slot 1 don't exist */
res = goal_pnioSubslotNew(pPnio, 0, 1, 1, GOAL_TRUE);
```

See example 01_simple_io for a demonstration.

3.56 goal_pnioModNew - Create a new module

Create a new module. Returns a GOAL_STATUS_T status.

Table 56: goal_pnioModNew parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint32_t idMod	module id

```
/* create module with id 0x30 */
res = goal_pnioModNew(pPnio, 0x30);
```

3.57 goal_pnioSubmodNew - Create a new submodule

Create a new submodule. Returns a GOAL_STATUS_T status.

Table 57: goal_pnioSubmodNew parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint32_t idMod	module id
uint32_t idSubmod	submodule id
GOAL_PNIO_MOD_TYPE_T type	submodule type (input, output, both)
uint16_t sizeInput	size of input module
uint16_t sizeOutput	size of output module
GOAL_BOOL_T flgAutogen	generate module if not exist

```
/* create 4 byte input submodule with id 0x30:0x01 even if module
 * doesn't exist */
res = goal_pnioSubmodNew(pPnio, 0x30, 0x01, GOAL_PNIO_MOD_TYPE_INPUT,
                         4, 0, GOAL_TRUE);
```

See example 01_simple_io for a demonstration.

3.58 goal_pnioModPlug - Plug a module into a slot

Plug a module into a slot. Returns a GOAL_STATUS_T status.

Table 58: goal_pnioModPlug parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint32_t idApi	API (always 0)
uint16_t idSlot	slot number
uint32_t idMod	module id

```
/* plug module 0x30 into slot 1 */
res = goal_pnioModPlug(pPnio, 0, 1, 0x30);
```

3.59 goal_pnioSubmodPlug - Plug a submodule into a subslot

Plug a submodule into a subslot. Returns a GOAL_STATUS_T status.

Table 59: goal_pnioSubmodPlug parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint32_t idApi	API (always 0)
uint16_t idSlot	slot number
uint16_t idSubslot	subslot number
uint32_t idMod	module id
uint32_t idSubmod	submodule id

```
/* plug submodule 0x30:0x01 into subslot 1:1 */
res = goal_pnioSubmodPlug(pPnio, 0, 1, 1, 0x30, 0x01);
```

See example 01_simple_io for a demonstration.

3.60 goal_pnioModPull - Pull a module from a slot

Pull a module from a slot. Returns a GOAL_STATUS_T status.

Table 60: goal_pnioModPull parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint32_t idApi	API (always 0)
uint16_t idSlot	slot number

```
/* pull module from slot 1 */
res = goal_pnioModPull(pPnio, 0, 1);
```

See example 08_dynamic_modules for a demonstration.

3.61 goal_pnioSubmodPull - Pull a submodule from a subslot

Pull a submodule from a subslot. Returns a GOAL_STATUS_T status.

Table 61: goal_pnioSubmodPull parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint32_t idApi	API (always 0)
uint16_t idSlot	slot number
uint16_t idSubslot	subslot number

```
/* pull submodule from subslot 1:1 */
res = goal_pnioSubmodPull(pPnio, 0, 1, 1);
```

3.62 goal_pnioDataOutputGet - Get output data from a submodule

Get output data from a submodule. Returns a GOAL_STATUS_T status.

Table 62: goal_pnioDataOutputGet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint32_t idApi	API (always 0)
uint16_t idSlot	slot number
uint16_t idSubslot	subslot number
char *pBuf	data buffer
uint16_t len	data buffer length
GOAL_PNIO_IOXS_T *pStateIops	pointer for producer state

```
GOAL_PNIO_IOXS_T iops = GOAL_PNIO_IOXS_GOOD;

/* get 4 bytes of output data for slot 1:1 */
res = goal_pnioDataOutputGet(pPnio, 0, 1, 1, &buf, 4, &iops);
```

See example 01_simple_io for a demonstration.

3.63 goal_pnioDataInputSet - Set input data for a submodule

Set input data for a submodule. Returns a GOAL_STATUS_T status.

Table 63: goal_pnioDataInputSet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint32_t idApi	API (always 0)
uint16_t idSlot	slot number
uint16_t idSubslot	subslot number
char *pBuf	data buffer
uint16_t len	data buffer length
GOAL_PNIO_IOXS_T stateIops	producer state

```
GOAL_PNIO_IOXS_T iops = GOAL_PNIO_IOXS_GOOD;

/* set 4 bytes of input data for slot 2:1 */
res = goal_pnioDataInputSet(pPnio, 0, 2, 1, &buf, 4,
                            GOAL_PNIO_IOXS_GOOD);
```

See example 01_simple_io for a demonstration.

3.64 goal_pnioApduStatusGet - Get the application protocol data unit status

Get the application protocol data unit status. Returns a GOAL_STATUS_T status.

Table 64: goal_pnioApduStatusGet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
GOAL_PNIO_AR_ID_T idAr	application relation id
GOAL_PNIO_APDU_STATUS_T *pStatusApdu	APDU status pointer

See example 06_apdu_status for a demonstration.

3.65 goal_pnioAlarmNotifySend - Send an alarm notification

Send an alarm notification. Returns a GOAL_STATUS_T status.

Table 65: goal_pnioAlarmNotifySend parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data

Parameter	Description
uint16_t *pNrAlarmSeq	pointer to store alarm sequence number
GOAL_PNIO_AR_ID_T idAr	application relation id
uint32_t prio	priority
const GOAL_PNIO_ALARM_ NOTIFY_T *pAlarmNotify	pointer to alarm notification
uint16_t lenDataUser	user data length
void *pDataUser	pointer to user data

See example 07_alarm_button for a demonstration.

3.66 goal_pnioAlarmNotifySendAck - Send an alarm notification acknowledge

Send an alarm notification acknowledge. Returns a GOAL_STATUS_T status.

Table 66: goal_pnioAlarmNotifySendAck parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
GOAL_PNIO_AR_ID_T idAr	application relation id
uint32_t prio	priority
const GOAL_PNIO_ALARM_ NOTIFY_T *pAlarmNotify	pointer to alarm notification
GOAL_PNIO_STATUS_T *pStatus	PROFINET status
uint16_t lenDataUser	user data length
void *pDataUser	pointer to user data

See example 07_alarm_button for a demonstration.

3.67 goal_pnioAlarmProcessSend - Send a process alarm

Send a process alarm. Returns a GOAL_STATUS_T status.

Table 67: goal_pnioAlarmProcessSend parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint32_t api	API
uint16_t slot	slot number

Parameter	Description
uint16_t subslot	subslot number
uint16_t usi	user structure identifier
uint16_t lenData	data length
uint8_t *pData	pointer to user data

See example 17_process_alert for a demonstration.

3.68 goal_pnioRecReadFinish - Answer a record read request

Answer a record read request. The sequence number is used to detect if the request has expired. If that is the case the response will be silently dropped. Returns a GOAL_STATUS_T status.

Table 68: goal_pnioRecReadFinish parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
int32_t hdl	record handle
GOAL_PNIO_STATUS_T *pStatusPnio	PROFINET status pointer
const uint8_t *pData	record data
uint16_t len	record data length
uint32_t numSeq	sequence number

See example 09_busy_records for a demonstration.

3.69 goal_pnioRecWriteFinish - Answer a record write request

Answer a record write request. The sequence number is used to detect if the request has expired. If that is the case the response will be silently dropped. Returns a GOAL_STATUS_T status.

Table 69: goal_pnioRecWriteFinish parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
int32_t hdl	record handle
GOAL_PNIO_STATUS_T *pStatusPnio	PROFINET status pointer
uint32_t numSeq	sequence number

See example 09_busy_records for a demonstration.

3.70 goal_pnioDiagExtChanDiagAdd - Add an extended channel diagnosis entry

Add an extended channel diagnosis entry. Returns a GOAL_STATUS_T status.

Table 70: goal_pnioDiagExtChanDiagAdd parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
GOAL_PNIO_DIAG_HANDLE_T *pHdl	pointer to store diagnosis handle
uint32_t api	API
uint16_t slot	slot number
uint16_t subslot	subslot number
uint16_t chan	channel index
uint16_t numErr	error number
uint16_t typeExtChanErr	extended channel error type
uint32_t valExtChanAdd	extended channel error value
GOAL_BOOL_T flgMaintReq	maintenance required flag
GOAL_BOOL_T flgMaintDem	maintenance demanded flag
GOAL_BOOL_T flgSubmitAlarm	submit alarm flag
uint16_t typeAlarm	alarm type

See example 12_diag_entry for a demonstration.

3.71 goal_pnioDiagChanDiagRemove - Remove an channel diagnosis entry

Remove an channel diagnosis entry. Returns a GOAL_STATUS_T status.

Table 71: goal_pnioDiagChanDiagRemove parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
GOAL_PNIO_DIAG_HANDLE_T hdl	diagnosis handle
GOAL_BOOL_T flgSubmitAlarm	submit alarm flag

See example 12_diag_entry for a demonstration.

3.72 goal_pnioCyclicCtrl - Control cyclic data received callback

Control cyclic data received callback. If flgCyclic is set to GOAL_TRUE then the callback GOAL_PNIO_CB_ID_NEW_IO_DATA signalizes the reception of new I/O data. This will led to a much higher system load. For most applications it is sufficient to poll the cyclic data by goal_pnioDataOutputGet. Returns a GOAL_STATUS_T status.

Table 72: goal_pnioCyclicCtrl parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
GOAL_BOOL_T flgCyclic	cyclic data enable flag

3.73 goal_pnioVendorIdSet - Set the vendor id

Set the vendor id. Returns a GOAL_STATUS_T status.

Table 73: goal_pnioVendorIdSet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint16_t id	vendor id

See example 14_info_set for a demonstration.

3.74 goal_pnioDeviceIdSet - Set the device id

Set the device id. Returns a GOAL_STATUS_T status.

Table 74: goal_pnioDeviceIdSet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint16_t id	device id

See example 14_info_set for a demonstration.

3.75 goal_pnioHwRevSet - Set the hardware revision

Set the hardware revision. Returns a GOAL_STATUS_T status.

Table 75: goal_pnioHwRevSet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint16_t revHw	hardware revision

See example 14_info_set for a demonstration.

3.76 goal_pnioSwRevSet - Set the software revision

Set the software revision. Returns a GOAL_STATUS_T status.

Table 76: goal_pnioSwRevSet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint8_t chrPrefix	prefix
uint8_t idFuncEnh	functional enhancement
uint8_t idBugfix	bugfix
uint8_t idIntChange	internal change
uint16_t cntRev	revision counter

See example 14_info_set for a demonstration.

3.77 goal_pnioProfileIdSet - Set the profile id

Set the profile id. Returns a GOAL_STATUS_T status.

Table 77: goal_pnioProfileIdSet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint16_t id	profile id
uint16_t type	profile specific type

See example 14_info_set for a demonstration.

3.78 goal_pnioOrderIdSet - Set the order id

Set the order id. Returns a GOAL_STATUS_T status.

Table 78: goal_pnioOrderIdSet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
const char *strOrder	order id
uint32_t lenOrder	order id length

See example 14_info_set for a demonstration.

3.79 goal_pnioSerialNumSet - Set the serial number

Set the serial number. Returns a GOAL_STATUS_T status.

Table 79: goal_pnioSerialNumSet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
const char *strSerial	serial number
uint32_t lenSerial	serial number length

See example 14_info_set for a demonstration.

3.80 goal_pnioVendorNameSet - Set the vendor name

Set the vendor name. Returns a GOAL_STATUS_T status.

Table 80: goal_pnioVendorNameSet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
const char *strVendor	vendor name
uint32_t lenVendor	vendor name length

See example 14_info_set for a demonstration.

3.81 goal_pnioPortDescSet - Set the port description

Set the port description. Returns a GOAL_STATUS_T status.

Table 81: goal_pnioPortDescSet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
const char *strPort	port description
uint32_t lenPort	port description length

See example 14_info_set for a demonstration.

3.82 goal_pnioSystemDescSet - Set the system description

Set the system description. Returns a GOAL_STATUS_T status.

Table 82: goal_pnioSystemDescSet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
const char *strSys	system description
uint32_t lenSys	system description length

See example 14_info_set for a demonstration.

3.83 goal_pnioSubslotStateSet - Permit usage of wrong submodules (substitute)

If a not-matching submodule is plugged the application can use this function to mark the submodule as substitute. Possible values for state are:

- GOAL_PNIO_SUBSLOT_STATE_OK
- GOAL_PNIO_SUBSLOT_STATE_SUBST

The default value is GOAL_PNIO_SUBSLOT_STATE_OK and handles the subslot state internally. GOAL_PNIO_SUBSLOT_STATE_SUBST overrides the subslot state with **substitute** if a submodule is plugged and only has a non-matching module or submodule id.

Table 83: goal_pnioSubslotStateSet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET data
uint32_t idApi	API

Parameter	Description
uint16_t idSlot	slot id
uint16_t idSubslot	subslot id
uint16_t state	subslot state

See example 20_subst_mod for a demonstration.

3.84 goal_pnioConfClassGet - Get active conformance class

The GOAL PROFINET stack supports Conformance Class A and B. There is currently one special case where the GOAL PROFINET needs to know the used Conformance Class. The PdevData record must only be supported for Conformance Class B and higher. Valid return values for pIdConfClass are GOAL_PNIO_CC_A and GOAL_PNIO_CC_B.

Table 84: goal_pnioConfClassGet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET instance
GOAL_PNIO_CC_T *pIdConfClass	Conformance Class storage

3.85 goal_pnioConfClassSet - Set active conformance class

The GOAL PROFINET stack supports Conformance Class A and B. There is currently one special case where the GOAL PROFINET needs to know the used Conformance Class. The PdevData record must only be supported for Conformance Class B and higher. Valid values for idConfClass are GOAL_PNIO_CC_A and GOAL_PNIO_CC_B.

Table 85: goal_pnioConfClassSet parameters

Parameter	Description
GOAL_PNIO_T *pPnio	PROFINET instance
GOAL_PNIO_CC_T idConfClass	Conformance Class

3.86 goal_pnioConfTestGet - Get current PROFINET test environment

This delivers detailed information about the currently used test environment.

Table 86: goal_pnioConfTestGet parameters

Parameter	Description
const char **pStrPnioStd	PROFINET Standard
const char **pStrTestSpec	PROFINET Test Specification
const char **pStrTestcases	PROFINET Testcases
const char **pStrArt	PROFINET ART
const char **pStrTedCheck	PROFINET TED Check
const char **pStrPnio	PROFINET Version
const char **pStrGsd	PROFINET GSDML Version

4 Application Callbacks

To use the full potential of *PROFINET* the stack allows you to interact at several stages of the protocol. For example you can withhold the sending of the application ready signal to the PLC or handle record data reads and writes to specific slots.

The first thing to use the callback system is to provide a callback function that the stack can call every time it wants to communicate with the application. The callback function must have the following prototype:

```
GOAL_STATUS_T main_callback(
    GOAL_PNIO_T *pPnio,             /**< PROFINET handle */
    GOAL_PNIO_CB_ID_T id,           /**< callback id */
    GOAL_PNIO_CB_DATA_T *pCb        /**< callback parameters */
);
```

It is registered with the call to *goal_pnioInit* which takes the pointer to the callback function as second parameter.

```
res = goal\_pnioInit(&pPnio, main_callback);
```

Now every time the *main_callback* is called, the *GOAL_PNIO_CB_ID_T* value contains the reason of the call. The *GOAL_PNIO_CB_DATA_T* structure contains an array of multiple unions were each array element has a special meaning depending on the *GOAL_PNIO_CB_ID_T* value.

The following subsections will provide you with the details on the available callback ids.

4.1 GOAL_PNIO_CB_ID_ALARM_ACK_TIMEOUT - Timeout waiting for Alarm ACK

Indicate that a timeout occurred in APMS while waiting for an alarm acknowledge.

Parameter	Description
cb->data[0].idAr	application relation id
cb->data[1].u16	timed out alarm sequence number

4.2 GOAL_PNIO_CB_ID_ALARM_NOTIFY_ACK - Alarm notification ACK received

Indicates that an alarm notification ACK was received.

Parameter	Description
cb->data[0].idAr	application relation id
cb->data[1].u16	alarm priority (low, high)
cb->data[2].pAlarmNotifyAck	GOAL_PNIO_ALARM_NOTIFY_ACK_T structure pointer

4.3 GOAL_PNIO_CB_ID_ALARM_NOTIFY - Alarm notification received

Indicates that an alarm notification was received. If the callback isn't handled or GOAL_OK is returned the PROFINET stack will automatically acknowledge the received alarm notification.

Parameter	Description
cb->data[0].idAr	application relation id
cb->data[1].u16	alarm priority (low, high)
cb->data[2].pAlarmNotify	GOAL_PNIO_ALARM_NOTIFY_T structure pointer
cb->data[3].u32	user data length
cb->data[4].pCu8	user data pointer

4.4 GOAL_PNIO_CB_ID_APPL_READY - Application Ready Response Received

Indicate that an application ready response was received. The return value has to be *GOAL_OK*.

Parameter	Description
cb->data[0].idAr	application relation id

4.5 GOAL_PNIO_CB_ID_BLINK - Blink Request

Indicates that a DCP signal request was received and fills the callback data with the necessary information.

cb->data[0].stateDcpBlink tells the application the initial signal request (START), the correct time to toggle the signal (TOGGLE) and when the signal phase is over (FINISH). This information can be used if the application wants to fully represent the signal by itself.

The second possibility is to use the cb->data[1].stateDcpLight data which contains either GOAL_PNIO_DCP_LIGHT_OFF or GOAL_PNIO_DCP_LIGHT_ON which can be used to handover the signal indicator directly to a LED or blink function.

If variant 1 or 2 is used than the callback has to return GOAL_OK_SUPPORTED otherwise the stack handles the blinking by itself by calling goal_targetSetLeds with the GOAL_PNIO_LED_SIGNAL define. For stack internal handling the implementation of goal_targetGetLeds and goal_targetSetLeds is mandatory.

Parameter	Description
cb->data[0].stateDcpBlink	DCP blink state (start, toggle, finish)
cb->data[1].stateDcpLight	DCP light state (on, off)

Return Values	Description
GOAL_OK	signal handled stack internal
GOAL_OK_SUPPORTED	application handles signal

```
    case GOAL_PNIO_CB_ID_BLINK:

#if FIRST_SCENARIO
        /*************************************/
        /* first scenario: act on blink state */
        /*************************************/
        switch (cb->data[0].stateDcpBlink) {

            case GOAL_PNIO_DCP_BLINK_START:
                /* start blinking */
                break;

            case GOAL_PNIO_DCP_BLINK_TOGGLE:
                /* toggle signal */
                break;

            case GOAL_PNIO_DCP_BLINK_FINISH:
                /* switch signal off */
                break;
        }

        res = GOAL_OK_SUPPORTED;
        break;
#endif /* FIRST_SCENARIO */

#if SECOND_SCENARIO
        /*************************************/
        /* second scenario: act on light state */
        /*************************************/
```

```
        vendorBlinkFunction(cb->data[1].stateDcpLight);
        res = GOAL_OK_SUPPORTED;
        break;
#endif /* SECOND_SCENARIO */

#if THIRD_SCENARIO
        /*************************************************************/
        /* third scenario: do nothing / PROFINET stack handles signal */
        /*************************************************************/
#endif /* THIRD_SCENARIO */
```

4.6 GOAL_PNIO_CB_ID_CONNECT_FINISH - Connect Request Done

Indicates that a connect request was fully processed and allows the application to cancel it by setting the error status and return a value that is not *GOAL_OK*.

Parameter	Description
cb->data[0].idAr	application relation id
cb->data[1].pStatus	pointer to PROFINET status

4.7 GOAL_PNIO_CB_ID_CONNECT_REQUEST - Connect Request

Indicates that the processing of a connect request has been started. The parameter pointer is set to *NULL*.

4.8 GOAL_PNIO_CB_ID_CONNECT_REQUEST_EXP_START - Expected Submodule Block Start

Indicates that an expected submodule block starts. This can be used to unplug all modules before the request callback for each module comes in.

Parameter	Description
cb->data[0].idAr	application relation id

4.9 GOAL_PNIO_CB_ID_END_OF_PARAM - Param End Received

Indicates that the parameter end information was received.

Parameter	Description
cb->data[0].idAr	application relation id

4.10 GOAL_PNIO_CB_ID_END_OF_PARAM_PLUG - Plug Param End Received

Indicates the the plug parameter end information was received.

Parameter	Description
cb->data[0].idAr	application relation id
cb->data[1].u16	plug handle

4.11 GOAL_PNIO_CB_ID_EXP_SUBMOD - Expected Submodule

Indicates an expected submodule which can be plugged immediately.

Parameter	Description
cb->data[0].idAr	application relation id
cb->data[1].u32	API
cb->data[2].u16	slot number
cb->data[3].u16	subslot number
cb->data[4].u32	module ident number
cb->data[5].u32	submodule ident number
cb->data[6].valBool	module flag (true = module, false = submodule)

4.12 GOAL_PNIO_CB_ID_FACTORY_RESET - Factory Reset

Indicates a factory reset. The parameter pointer is set to *NULL*.

4.13 GOAL_PNIO_CB_ID_IO_DATA_TIMEOUT - Cyclic Timeout

Indicates that an output endpoint timed out.

4.14 GOAL_PNIO_CB_ID_NET_IP_SET - IP Configuration Update

Indicate that the IP configuration was updated. The internal change flag indicates if the change was triggered by PROFINET DCP (PN_TRUE) or by an external caller like DHCP (PN_FALSE).

Parameter	Description
cb->data[0].u32	ip address
cb->data[1].u32	netmask
cb->data[2].u32	gateway
cb->data[3].valBool	temporary setting flag
cb->data[4].valBool	internal update flag

4.15 GOAL_PNIO_CB_ID_NEW_AR - New Application Relation

Indicates a new application relation. A return value different from *GOAL_OK* drops the AR and replies with an error.

Parameter	Description
cb->data[0].idAr	application relation id

4.16 GOAL_PNIO_CB_ID_NEW_IO_DATA - New IO Data

Indicates reception of new IO data.

Parameter	Description
cb->data[0].u16	cyclic frame id
cb->data[1].u16	data length
cb->data[2].pCu8	data pointer

4.17 GOAL_PNIO_CB_ID_PLUG_READY - Plug Ready Response Received

Indicates that a plug ready response was received.

Parameter	Description
cb->data[0].idAr	application relation id

4.18 GOAL_PNIO_CB_ID_READ_RECORD - Read Record Data

Indicates that record data that couldn't be handled by the stack itself should be read. Before this callback is called, the stack checks if the API, slot and subslot combination is valid.

If the function returns GOAL_OK the request will be denied with "invalid index".

Otherwise, if the application can handle the request, it needs to return GOAL_OK_SUPPORTED and can provide the answer either directly in the callback by calling the API goal_pnioRecReadFinish or store the callback information and respond outside the callback, also by calling the API goal_pnioRecReadFinish. The API goal_pnioRecReadFinish also allows to overwrite the PROFINET status if the matching parameter is set.

If the response isn't GOAL_OK or GOAL_OK_SUPPORTED the stack will automatically respond with application read error.

The parameter "sequence number handle" is used later when the response is send to detect if the request has expired.

Parameter	Description
cb->data[0].pStatus	unused (~~pointer to PROFINET status~~)
cb->data[1].idAr	application relation id
cb->data[2].u32	API
cb->data[3].u16	slot
cb->data[4].u16	subslot
cb->data[5].u16	record index
cb->data[6].pU8	unused (~~pointer to store record data~~)
cb->data[7].u32	maximum record read data length
cb->data[8].i32	internal record handle
cb->data[9].u32	sequence number handle

Example callback code:

```
case GOAL_PNIO_CB_ID_READ_RECORD:
```

```
goal_logInfo("read request received");
goal_logInfo("address: %"FMT_u32":%u:%u, idx: %u",
             pCb->data[2].u32, /* API */
             pCb->data[3].u16, /* slot */
             pCb->data[4].u16, /* subslot */
             pCb->data[5].u16  /* record index */);

if (GOAL_TRUE == flgAnswerInCallback) {

    /* send answer direct in callback */
    res = goal_pnioRecReadFinish(
            pPnio,                          /* PROFINET handle */
            pCb->data[8].i32,               /* internal record handle */
            NULL,                           /* PROFINET Status or NULL */
            DATA_BUFFER,                    /* answer data buffer */
            DATA_LENGTH,                    /* answer data length */
            pCb->data[9].u32                /* sequence number handle */
          );
    if (GOAL_RES_ERR(res)) {
        goal_logErr("failed to answer record read request");
    }

} else {

    /* postpone answer by storing the request */
    GOAL_MEMCPY(REQUEST_STORAGE, pCb, sizeof(GOAL_PNIO_CB_DATA_T));

}

/* tell the PROFINET stack that the record was or will be handled */
return GOAL_OK_SUPPORTED;
```

4.19 GOAL_PNIO_CB_ID_RELEASE_AR - Release Application Relation

Indicates that an application relation was released.

Parameter	Description
cb->data[0].idAr	application relation id

4.20 GOAL_PNIO_CB_ID_RESET_TO_FACTORY - Reset To Factory

Indicates a reset to factory request. If there is a separate request handling outside of the stack the application must return GOAL_OK_SUPPORTED.

If the application doesn't return from the callback with GOAL_OK or GOAL_OK_SUPPORTED the DCP error "0x04 suboption not set" is sent back to the DCP set request sender.

Parameter	Description
cb->data[0].u16	reset to factory action

4.21 GOAL_PNIO_CB_ID_STATION_NAME - Station Name Changed

Indicates a station name change. If the permanent flag is set to GOAL_TRUE, the station name is stored in NVS, otherwise the station name is only stored temporary and the NVS value is cleared.

Parameter	Description
cb->data[0].pCu8	name of station
cb->data[1].u32	name of station length
cb->data[2].valBool	permanent flag

4.22 GOAL_PNIO_CB_ID_WRITE_RECORD - Write Record Data

Indicates that record data that couldn't be handled by the stack itself should be written. Before this callback is called, the stack checks if the API, slot and subslot combination is valid.

If the function returns GOAL_OK the request will be denied with "invalid index".

Otherwise, if the application can handle the request, it needs to return GOAL_OK_SUPPORTED and can provide the answer either directly in the callback by calling the API goal_pnioRecWriteFinish or store the callback information and respond outside the callback, also by calling the API goal_pnioRecWriteFinish. The API goal_pnioRecReadFinish also allows to overwrite the PROFINET status if the matching parameter is set.

If the response isn't GOAL_OK or GOAL_OK_SUPPORTED the stack will automatically respond with application read error.

The parameter "sequence number handle" is used later when the response is send to detect if the request has expired.

Parameter	Description
cb->data[0].pStatus	unused (~~pointer to PROFINET status~~)
cb->data[1].idAr	application relation id
cb->data[2].u32	API
cb->data[3].u16	slot
cb->data[4].u16	subslot
cb->data[5].u16	record index
cb->data[6].pCu8	pointer to read record data from
cb->data[7].u32	record data length
cb->data[8].i32	internal record handle
cb->data[9].valBool	subslot locked status flag
cb->data[10].u32	sequence number handle

Example callback code:

```
case GOAL_PNIO_CB_ID_WRITE_RECORD:

    goal_logInfo("write request received");
    goal_logInfo("address: %"FMT_u32":%u:%u, idx: %u",
                 pCb->data[2].u32, /* API */
                 pCb->data[3].u16, /* slot */
                 pCb->data[4].u16, /* subslot */
                 pCb->data[5].u16  /* record index */);

    if (GOAL_TRUE == flgAnswerInCallback) {

        /* send answer direct in callback */
        res = goal_pnioRecWriteFinish(
                pPnio,                          /* PROFINET handle */
                pCb->data[8].i32,               /* internal record handle */
                NULL,                           /* PROFINET Status or NULL */
                pCb->data[10].u32               /* sequence number handle */
              );
        if (GOAL_RES_ERR(res)) {
            goal_logErr("failed to answer record write request");
        }

    } else {

        /* postpone answer by storing the request */
        GOAL_MEMCPY(REQUEST_STORAGE, pCb, sizeof(GOAL_PNIO_CB_DATA_T));

    }
```

```
/* tell the PROFINET stack that the record was or will be handled */
return GOAL_OK_SUPPORTED;
```

4.23 GOAL_PNIO_CB_ID_INIT - Stack initialized

Indicates that the PROFINET stack is fully initialized. The parameter pointer is set to *NULL*. At this point it is safe to create the device configuration.

4.24 GOAL_PNIO_CB_ID_LLDP_UPDATE - LLDP Update

Signalizes that the device on the partner port changed.

Parameter	Description
cb->data[0].u32	GOAL Ethernet Port Id

4.25 GOAL_PNIO_CB_ID_CONN_REQ_EXP_FINISH - Connect Request Expected Submodule Block Finish

This callback is called after the Expected Submodule Block in the Connect Request has been parsed.

Parameter	Description
cb->data[0].idAr	application relation id

4.26 GOAL_PNIO_CB_ID_STATION_NAME_VERIFY - DCP Station Name Verification

Let the application verify a DCP Station Name Set Request.

If the function returns a GOAL error status the set request will be denied.

Parameter	Description
cb->data[0].pCu8	pointer to station name (unterminated)
cb->data[1].u32	length of station name
cb->data[2].valBool	permanent flag (permanent = true)

4.27 GOAL_PNIO_CB_ID_NET_IP_SET_VERIFY - DCP IP Configuration Verification

Let the application verify a DCP IP Configuration Set Request.

If the function returns a GOAL error status the set request will be denied.

Parameter	Description
cb->data[0].u32	IP address
cb->data[1].u32	netmask
cb->data[2].u32	gateway
cb->data[3].valBool	temporary flag (temporary = true)

5 GSDML

5.1 Known restrictions

- IOXS_Required = false is not supported

5.2 Settings

- MaxSupportedRecordSize must match the size that is configured with goal_pnioCfgRpcFragReqLenMaxSet.

6 Conformance

Identifier	Release
PROFINET Standard	V2.3Ed.2 MU5
Test Specification	V2.35b May 2018
Testcase Version	Bundle Sept 2018
ART Version	V2.35.2.2_RC
TED Check	V2.35.2.0004
PROFINET Version	2.35
GSD Version	2.35

7 PROFINET Features

7.1 Conformance Class A

- Supported Features
  - PROFINET Version: 2.35
  - Asynchronous data exchange (read and write record data)
  - Cyclic data exchange up to 1 ms (realtime data)
  - Link Layer Discovery Protocol (LLDP, Topology)
  - Topology detection and monitoring
  - Tool-less device replacement
  - Discovery and Basic Configuration Protocol (DCP)
  - Multiple Ethernet Interfaces (single port und switch support)
  - Virtual LAN (VLAN)
  - dynamic module pull and plug
  - Netload Class 1 (depending on environment)
  - MultipleWrite support
  - Alarms and diagnosis (system and application)
  - Processalarms
  - Direct access of IOxS and APDU information
  - Support for multiple MAC address (LLDP)
  - Device-Access
  - Ethernet: 100 Mbit/s, 1000 Mbit/s
  - Support for non-standard Interface- and Port-Submodule placement
  - Identification & Maintenance records 0-4 (system and application)
  - Manufacturer Specific Alarms
- Unsupported Features / Limitations
  - Only API 0 is supported
  - Profiles
  - Multicast-Relation Support (MCR)
  - Multiple-ARs
  - DHCP
  - Redundancy
  - Shared-Device
  - Shared-Inputs
  - Fast start-up (FSU)
  - PROFIenergy
  - PROFIsafe (possible with external software)
  - Media Redundancy Protocol (MRP)
  - Realtime Data via UDP
  - Configuration In Run (CIR)
  - Isochronous Real Time (IRT)

7.2 Conformance Class B

- Additional features to Conformance Class A
  - SNMP (Simple Network Management Protocol)
  - MIB2

8 Integration hints

8.1 General Integration Hints

8.1.1 MAC Address

- when the configuration option for LLDP MAC address generation is enabled, MAC addresses for ports are calculated from the base MAC address
- this option is enabled by default and can be disabled by (goal_pnioCfgLldpGenMacSet)

```
calculation of MAC addresses
============================

MAC (device)   = 0x02:0x00:0x00:0x00:0x00:0x01
MAC (port-001) = 0x02:0x00:0x00:0x00:0x00:0x02
MAC (port-002) = 0x02:0x00:0x00:0x00:0x00:0x03
```

8.1.2 NVS Storage

- via PROFINET (DCP) values can be set in the device which optionally need to be stored permanently
- to be conform to the PROFINET standard, this process needs to be executed timely and should not block normal device operation for more than 200ms
- BACKGROUND: in the conformance test it is checked if a DCP set request is processed and answered within a time window of 500ms

8.1.3 Firewall Settings

GOAL PROFINET Ethertypes:

Table 112: GOAL PROFINET Ethertypes

Ethertype	Description	Note
0x0800	Internet Protocol version 4 (IPv4)	only for GOAL managed TCP/IP Stacks
0x0806	Address Resolution Protocol (ARP)	only for GOAL managed TCP/IP Stacks
0x8100	VLAN-tagged frame (IEEE 802.1Q)	only if not removed by receive logic
0x8892	PROFINET Protocol	
0x88cc	Link Layer Discovery Protocol (LLDP)	

GOAL PROFINET UDP ports:

Table 113: GOAL PROFINET UDP ports

UDP Port	Description
0x8894	RPC port (34964, according to IANA)
0xc000	RPC responder port (49152, according to IANA)

GOAL PROFINET Multicast MAC addresses:

Table 114: GOAL PROFINET Multicast MAC addresses

Multicast MAC	Description
01:0e:cf:00:00:00	Discovery and Configuration Protocol (DCP
01:80:c2:00:00:0e	Link Layer Discovery Protocol (LLDP)

8.2 Platform specific Integration Hints

8.2.1 Example ARM Cortex-M3 Platform with GOAL 2.12

8.2.1.1 Memory Footprint (01_simple_io)

```
220 058 bytes of readonly code memory
69 079 bytes of readonly data memory
375 422 bytes of readwrite data memory (contains GOAL HEAP, thus not used 100%)
```

8.2.1.2 Memory Footprint (13_pnio_snmp)

```
275 816 bytes of readonly  code memory
78 228 bytes of readonly  data memory
420 413 bytes of readwrite data memory (contains GOAL HEAP, thus not used 100%)
```

8.2.1.2.1 GOAL heap (01_simple_io / 13_pnio_snmp)

```
heap usage id(GOAL_ID_TASK): 252 Bytes
heap usage id(GOAL_ID_TGT): 17004 Bytes
heap usage id(GOAL_ID_INSTANCE): 148 Bytes
heap usage id(GOAL_ID_MA_NVS): 88 Bytes
heap usage id(GOAL_ID_MI_NVS): 16867 Bytes
heap usage id(GOAL_ID_MAIN): 124 Bytes
heap usage id(GOAL_ID_DRV_NVS): 12 Bytes
```

```
heap usage id(GOAL_ID_MA_SECTION): 64 Bytes
heap usage id(GOAL_ID_DRV_SPI): 12 Bytes
heap usage id(GOAL_ID_MA_SPI): 100 Bytes
heap usage id(GOAL_ID_DRV_WD): 4 Bytes
heap usage id(GOAL_ID_LOCK): 624 Bytes
heap usage id(GOAL_ID_LOG): 5164 Bytes
heap usage id(GOAL_ID_CM): 8284 Bytes
heap usage id(GOAL_ID_QUEUE): 49960 Bytes
heap usage id(GOAL_ID_LM): 60 Bytes
heap usage id(GOAL_ID_STAT): 2388 Bytes
heap usage id(GOAL_ID_ETH): 120 Bytes
heap usage id(GOAL_ID_NET): 16 Bytes
heap usage id(GOAL_ID_TMR): 1280 Bytes
heap usage id(GOAL_ID_PNIO): 49368 Bytes
heap usage id(GOAL_ID_DD): 4 Bytes
memory allocation number: 568
memory allocation alignment overhead: 12
memory usage: 156964/259072 bytes
memory usage: (61%)
```

8.3 Certification Notes

8.3.1 Documentation

During the Conformance Test the behavior of the cyclic data is checked in the following situations:

- PLC enters state ‘stop’
- AR is closed

This behavior must be defined in the device manual. If not changed, the GOAL PROFINET stack handles the situations in this way:

- PLC enters state ‘stop’
  - Output data values switch to the configured SubstituteValue. Default: Zero
- AR is closed
  - Output data values are not longer available in memory, zeroed values are reported.