SIEMENS

GSDML Getting Started V1.10

Creating a Device Description (GSD) file for PROFINET devices

Preface

Introduction 1

Using schema files for validation 2

Rules for generating the GSD filename 3

Basic structure of a GSD file 4

Generating a GSD file 5

Device Access Point list 6

Definition of the PDEV 7

Definition of Real-time Classes 8

Modules, submodules, slots and subslots 9

Diagnostic definitions 10

Graphic symbols 11

Texts in GSDML 12

02/2024
A5E51228963-AC

SIEMENS

GSDML Getting Started V1.10

Creating a Device Description (GSD) file for PROFINET devices

Using references 13

Catalog information in the GSD 14

Description of parameter record data objects 15

SNMP and MIB2 16

Fast startup 17

Media redundancy 18

IRT performance upgrade 19

IRT without Isochronous mode 20

PROFINET Security Class 1 21

PROFIsafe definitions in the GSD file 22

PROFIenergy definitions in the GSD file 23

Compatibility of GSDML versions 24

Tools 25

GSD File Signing Process 26

Signature Validation of GSDX File 27

Appendix 28

02/2024

A5E51228963-AC

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER

indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING

indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION

indicates that minor personal injury can result if proper precautions are not taken.

NOTICE

indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E51228963-AC
Ⓟ 02/2024 Subject to change

Copyright © Siemens AG 2024.
All rights reserved

Preface

Purpose of the manual

The objective of this document is to facilitate getting acquainted with the structure of GSDML and to introduce implementing GSD files for PROFINET devices that are being developed. This document demonstrates how the elements and attributes of a GSDML document work in principle. However, the syntax of the XML itself and the structure of schema files will not be explained here.

The standard document GSDML Technical Specification for PROFINET is available on the website of the PNO.

In case of any deviations compared with the GSDML specification, the GSDML specification takes precedence over this document.

This document is constantly adapted to the requirements of the market. For that reason, feedback is always welcome. Please report any kind of incompatibilities to profinet.devkits.industry@siemens.com.

Target group of this manual

Knowledge of the PROFINET standard is a prerequisite.

Guide

The manual contains various navigation aids that allow you to find specific information more quickly:

- A complete table of contents is provided at the beginning of the manual.
- In the appendix you will find a list of abbreviations, which defines important technical terms used in this manual.

Conventions

You should also pay particular attention to notes such as the one shown below:

Note: A note contains important information on the product, on handling of the product and on the section of the documentation to which you should pay particular attention.

Terms and definitions

- GSD General Station Description

  The device description (file) of a specific device (either PROFINET or PROFIBUS).

- GSDML GSD Markup Language

  The language in which GSDs are written (XML, elements and attributes carry information).

Cybersecurity information

Siemens provides products and solutions with industrial cybersecurity functions that support the secure operation of plants, systems, machines, and networks.

In order to protect plants, systems, machines, and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial cybersecurity concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e. g. firewalls and/or network segmentation) are in place.

For additional information on industrial cybersecurity measures that may be implemented, please visit https://www.siemens.com/industrialsecurity.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customers' exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Cybersecurity RSS Feed under https://www.siemens.com/industrialsecurity.

Table of contents

1 Introduction **1**

2 Using schema files for validation **3**
2.1 Using schema files in GSDML context 3
2.2 GSDML schema files 4

3 Rules for generating the GSD filename **6**

4 Basic structure of a GSD file **7**
4.1 Processing instructions for XML parser 7
4.2 Root element 7
4.3 ProfileHeader 8
4.4 ProfileBody 9
4.4.1 ApplicationProcess 9
4.4.1.1 DeviceAccessPointList 9
4.4.1.2 ModuleList 9
4.4.1.3 SubmoduleList 9
4.4.1.4 ValueList 9
4.4.1.5 ChannelDiagList 10
4.4.1.6 ChannelProcessAlarmList 10
4.4.1.7 UnitDiagTypeList 10
4.4.1.8 LogBookEntryList 10
4.4.1.9 GraphicsList 10
4.4.1.10 CategoryList 10
4.4.1.11 ExternalTextList 10

5 Generating a GSD file **11**

6 Device Access Point list **13**

7 Definition of the PDEV **15**

8 Definition of Real-time Classes **17**

9 Modules, submodules, slots and subslots **18**
9.1 Definition of modules with isochronous mode capability 21

10 Diagnostic definitions **22**

11 Graphic symbols **24**

12 Texts in GSDML **25**
12.1 General 25
12.2 Incorporating foreign languages 26

13 Using references **28**
13.1 Checks using the schema 28
13.2 Tips for naming IDs meaningfully 29

14 Catalog information in the GSD **31**
14.1 Category assignment 31

15 Description of parameter record data objects **32**
15.1 "Const" definition of a ParameterRecordDataItem 32
15.2 "Ref" definition of a ParameterRecordDataItem 33

16 SNMP and MIB2 . . . . . **36**

17 Fast startup **37**

18 Media redundancy **38**

19 IRT performance upgrade **39**

20 IRT without Isochronous mode **41**

21 PROFINET Security Class 1 . . . . . **42**

22 PROFIsafe definitions in the GSD file **43**

23 PROFIenergy definitions in the GSD file **44**

24 Compatibility of GSDML versions **46**

25 Tools **48**
25.1 PROFINET GSD Checker 48
25.2 Tools for GSD generation 49

26 GSD File Signing Process **50**

27 Signature Validation of GSDX File **51**
27.1 Validating signature of GSDX file with GSD Checker tool 51
27.2 Validating signature of GSDX file with TIA Portal 52

28 Appendix **55**
28.1 Abbreviations 55

Introduction 1

The GSDML is an XML-based description language for PROFINET devices. By means of the opportunities introduced by GSDML, XML-based documents shall be generated that are capable of representing accurately the hierarchical device model. The language can also describe the attributes of a device family in several hierarchical levels. Broadly defined, a device family is a grouping of devices with respect to their function classes. Examples of device families include I/O systems, HMI stations, PLCs, etc.

- **Overview of implementation of a GSDML document**
  Implementing a GSDML document "from scratch" is performed according to the steps listed below[1].

Table 1.1: Implementing procedure

Step	Procedure	Comments	See Chapter
1	Generating the GSD filename	Conform to the model for GSD filenames (mandatory)	3
2	Defining the processing instructions	Define the encoding (optional)	4.1
3	Defining the root element	Specify namespaces and the import instructions for the schema files (mandatory)	4.2
4	Defining the ProfileHeader	Conform to the fixed structure (mandatory)	4.3
5	Defining the ProfileBody	Contains actual data (mandatory). The following steps in this table, step 6 and onwards, contain information about definitions in the ProfileBody.	4.4
6	Defining the device access point list	Specify the essential features of a PROFINET device (mandatory)	4.4.1.1 and 6
7	Defining the physical device	Specify hardware-oriented Interface Submodule and Port Submodule parameters (mandatory)	7
8	Defining modules, submodules, slots and subslots	Assign the modules and submodules to DAPs for physical configuration of a base module (optional)	9
9	Defining the Real-time Classes	Define DAP-specific communication mode parameters within PDEV (optional)	8
10	Enabling Fast startup features	(optional)	17
11	Enabling media redundancy	(optional)	18

continues on next page

[1] Please refer to Chapter 5 for adapting an existing GSD file.

Table 1.1 – continued from previous page

Step	Procedure	Comments	See Chapter
12	Enabling IRT performance upgrade features	(optional)	19
13	Diagnostic definitions	Define system-defined, profile-specific or diagnostic information (at least one of the related child elements is mandatory)	10
14	Enabling SNMP	(optional)	16
15	PROFIsafe definitions	Mandatory for I/O modules that support this functionality	22
16	Defining graphic symbols	Required for configuration in engineering tools (mandatory)	11
17	Defining catalog information	Required for representation in engineering tools (mandatory)	14

References

[1] GSDML Technical Specification for PROFINET

[2] Popp, Manfred (2014) Industrial Communication with PROFINET

[3] ISO 15745-1:2003/Amd 1:2007

[4] Diagnosis for PROFINET Guideline

[5] ISO 639-1 Codes

[6] REC-xmldsig-core1-20130411, XML Signature Syntax and Processing Version 1.1 - W3C Recommendation 11 April 2013, available at http://www.w3.org/TR/xmldsig-core1/

[7] PROFIenergy Technology

Using schema files for validation 2

An XML schema defines a type of an XML document in terms of additional constraints on the structure above and beyond the basic constraints imposed by the XML language itself. In addition to this, user groups can agree on a standard[2] for interchanging data. In other words, the content of the defined document type governed by data types can be constrained by XML schema definitions. To this end, the structure of an XML document can be checked using that schema file. If the syntax of an XML document is correct, it is called "well-formed". Additionally, if an XML document is validated against an XML schema, it is called both "well-formed" and "valid", that is, the document has a correct syntax, and it is in compliance with the agreed standard.

The GSD file can be edited by an XML editor and can be validated via an XML parser. An overview of those tools is provided in Chapter 25.

It is also possible to check the structure of a GSD file with the tool PROFINET GSD Checker which is downloadable free of charge from PNO website. The versioning of the tool is of specific significance: Suppose that a user has downloaded version 2.4 of PROFINET GSD Checker. That version of the tool supports the GSDML versions 2.4 and earlier. If a GSD file based on a newer GSDML version, say V2.41, is used as input, only the elements and attributes defined in GSDML version 2.4 will be displayed in the HTML view of the tool[3].

If PROFINET GSD Checker is used instead of other tools, specialist knowledge regarding the functions and the application of schema files is not required. This is due to the fact that the tool contains a function to check the validity of the GSD files. A brief introduction can be found in Section 25.1 and more information is available in the Help menu item of the tool.

2.1 Using schema files in GSDML context

The GSD Working Group (GSD WG) generates, updates and makes available the GSDML schema files contained in the download of the latest GSDML Specification for PROFINET from the PNO website[4]. By using an XML parser for validation, the author of a GSD file can check it against the provided schema and make the necessary corrections if required.

Figure 2.1 shows the sequence diagram for schema handling.

[2] For instance, restrictions on data.

[3] Note: At the time of writing of this document, the current released version is 2.43.

[4] The schema files are also contained within the tool PROFINET GSD Checker.

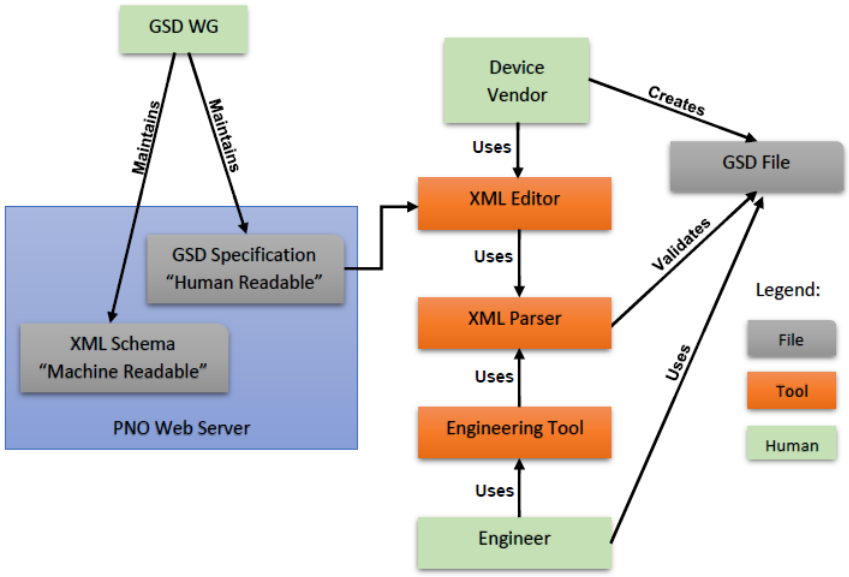

Fig. 2.1: **GSDML workflow**

2.2 GSDML schema files

Four schema files (filename extension **.xsd**) are required to validate a GSD file. These are arranged as follows:

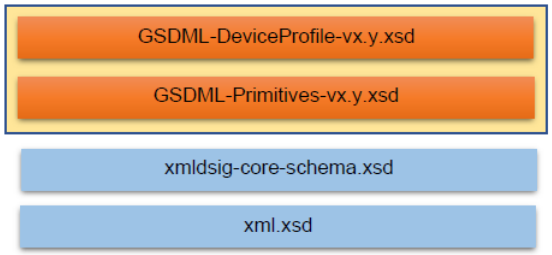

Fig. 2.2: **Schema files**

As mentioned above, the first two schema definitions whose filenames begin with "GSDML" in Figure 2.2 can be downloaded together with the GSDML specification from the related PNO website. They can be used for inspection and validation of the GSDs in an XML editor/parser. The rest can be downloaded from the W3C website, usually by using the namespace identifier as an internet address.

- **GSDML-DeviceProfile-vx.y.xsd** GSDML Device Profile schema

  Namespace > `http://www.profibus.com/GSDML/2003/11/DeviceProfile`

  This is the actual schema document for defining the GSDML. It uses the GSDML Primitives schema and can only be used in conjunction with it. *x.y* in the identifier is a substitute for the version of the schema.

- **GSDML-Primitives-vx.y.xsd** GSDML Primitives schema

  Namespace > `http://www.profibus.com/GSDML/2003/11/Primitives`

The following schema files are imported by the GSDML schemas:

- **xml.xsd** The "xml" namespace

  Namespace > http://www.w3.org/XML/1998/namespace

  Schema location > http://www.w3.org/2009/01/xml.xsd

  This schema defines attributes and attribute groups suitable for use by schemas wishing to allow "xml:lang" or "xml:space" attributes on elements they define.

- **xmldsig-core-schema.xsd** Schema for XML signatures

  Namespace and schema location > http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/xmldsig-core-schema.xsd

  This document specifies XML digital signature processing rules and syntax. XML signatures provide integrity, message authentication, and/or signer authentication services for data of any type.

Rules for generating the GSD filename

3

The length of the GSD filename is limited by the operating system.

The GSD filename must conform to the following model:

```
GSDML-[GSDML schema version]-[vendor name]-[device family
name]-[date](-[time]).xml
```

The following rules apply to the keywords in square brackets:

GSDML schema version: Version identifier of the referenced schema in Vx.y format, where *x* and *y* are unsigned integers. The version identifier must correspond to the one in the filename of the `GSDMLDeviceProfile-[GSDML schema version].xsd`.

Vendor name: Name of the device manufacturer. Vendor name may contain the Vendor ID to ensure a unique filename.

Device family name: Name of the device family described in the GSD. The name should correspond to the content of the attribute "ProductFamily" of the element "DeviceFunction".

The Vendor name and device family name may contain any character allowed by the Microsoft Windows NT family of operating systems including space " " and dash "-".

Date: The release date of the GSD file in *yyyymmdd* format. Please note that the manufacturer must ensure that different GSD files for the same device family are not released with the same date.

Time: The release time of the GSD file in *hhmmss* format (optional).

Specifying date and time makes keeping track of the version of the GSD file itself unnecessary and ensures that different versions do not overwrite an existing file. In addition to this, due to the requirement that already released GSD files shall not be changed without changing the filename, the release date or time shall be changed when a new version of a GSD is generated.

If more than one version of a GSD is installed, the device catalog in TIA Portal allows choosing from the different GSDML files belonging to the same device.

The following is an example of a valid filename:

```
GSDML-V2.43-Vendor-Device-20220831.xml
```

Basic structure of a GSD file 4

4.1 Processing instructions for XML parser

As for any XML document, the processing instructions must be defined at the beginning of a GSD file.

```
<?xml version="1.0" encoding="utf-8"?>
```

Here, only the definition of the encoding can be changed. There are no restrictions for this in GSDML, that is, all encodings conforming to the standard can be specified. However, some editors might have issues with Unicode data, and hence they do not interpret the byte order mark correctly. Therefore, using "iso-8859-1" or "UTF-8" is recommended. Besides, using other encodings, such as UTF-16 or UTF-32, may lead to a significant increase in the file size.

4.2 Root element

The root element of a GSD file is "ISO15745Profile" according to the definitions of ISO 15745-1:2003/Amd 1:2007[5]. Namespaces and the import instructions for the schema files must be specified in the root element.

```
<ISO15745Profile xmlns=http://www.profibus.com/GSDML/2003/11/DeviceProfile
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.profibus.com/GSDML/2003/11/DeviceProfile .\xsd\GSDML-
↪DeviceProfile-V2.43.xsd">
```

As specified in ISO 15745, an ISO15745Profile consists of a

- "ProfileHeader", the header section of the device profile
- "ProfileBody", the body section of the device profile
- an optional "Signature". This element allows an integrity check of a GSD. The structure of this element is defined in *[6]*.

[5] Industrial automation systems and integration — Open systems application integration framework — Part 1: Generic reference description — Amendment 1. Please refer to *[3]*.

Figure 4.1 shows the structure of a GSD file.

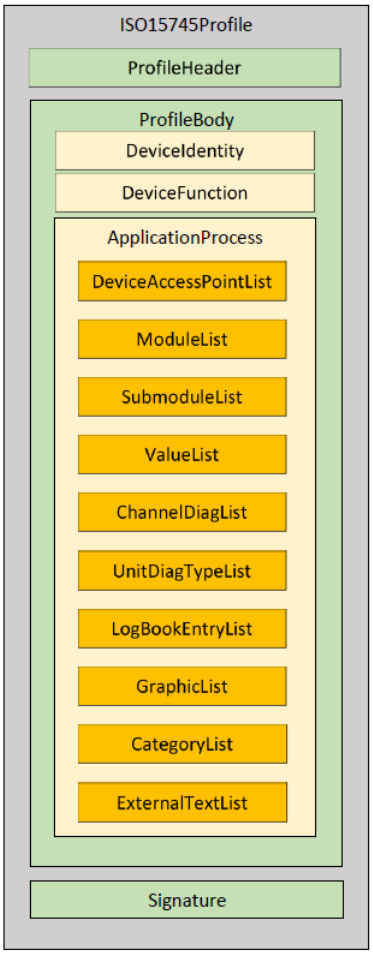

Fig. 4.1: **Structure of a GSD file**

4.3 ProfileHeader

The ProfileHeader shall have the following fixed content:

```
<ProfileHeader>
     <ProfileIdentification>PROFINET Device Profile</ProfileIdentification>
     <ProfileRevision>1.00</ProfileRevision>
     <ProfileName>Device Profile for PROFINET Devices</ProfileName>
     <ProfileSource>PROFIBUS Nutzerorganisation e. V. (PNO)</ProfileSource>
     <ProfileClassID>Device</ProfileClassID>
     <ISO15745Reference>
             <ISO15745Part>4</ISO15745Part>
             <ISO15745Edition>1</ISO15745Edition>
             <ProfileTechnology>GSDML</ProfileTechnology>
     </ISO15745Reference>
</ProfileHeader>
```

4.4 ProfileBody

The ProfileBody contains the actual data of the field device and consists of three parts:

- **DeviceIdentity** contains attributes which uniquely identify the device.
- **DeviceFunction** describes the intrinsic function of a device in terms of its technology.
- **ApplicationProcess** describes the behavior of the device in terms of the application, independent of the device technology.

4.4.1 ApplicationProcess

ApplicationProcess is the main part of the description data. It contains DAPs[6], modules and submodules which are the building blocks for configuring a device. Most important sections of the ApplicationProcess block are listed in the following subchapters.

4.4.1.1 DeviceAccessPointList

This section contains the description of the individual DAPs[7] which represent the interface / head module of the device, especially the possible configurations[8]. As mentioned above, a GSD file can contain a description for any number of interface modules. The sum of all DAPs forms a device family whose members all share the same VendorID and DeviceID. Different modules (base modules) can be assigned to each DAP.

4.4.1.2 ModuleList

This section includes the description of the individual modules of a field device. These can be plugged into the slots which are described by the DAP.

4.4.1.3 SubmoduleList

This section includes the description of submodules which can be plugged into the subslots of a DAP or a module by the engineering tool.

4.4.1.4 ValueList

This section contains the individual parameters of a field device in the form of an assignment between numerical values (an enumeration) and the associated text strings.

[6] Represented by the element "DeviceAccessPointItem".

[7] DAP is formally defined in the specification as a special module which connects the device to the network. "A DeviceAccessPointItem (DAP) represents the interface / head module of the device. It contains the network interface, and describes the general properties of the device, especially the possible configurations (plugging rules)" *[1]*.

[8] I.e., plugging rules

4.4.1.5 ChannelDiagList

This section contains the assignment between a channel error of a field device and the corresponding texts.

4.4.1.6 ChannelProcessAlarmList

This section contains the textual descriptions of channel process alarms.

4.4.1.7 UnitDiagTypeList

This section contains the assignment between diagnostic values and manufacturer specific status and error messages in **U**ser **S**tructure **I**dentifier (USI) format.

4.4.1.8 LogBookEntryList

This section contains the descriptions for LogBook entries which define a manufacturer specific PNIOStatus.

4.4.1.9 GraphicsList

This section includes the references to graphic representations of a field device.

4.4.1.10 CategoryList

This section includes certain categories (for example, Digital Input, Analog Output, etc.). It is used for the arrangement and better retrieval within the module catalog of an engineering tool.

4.4.1.11 ExternalTextList

This section contains all language-dependent text strings that can be referenced by the other sections. More detail is provided in Section 12.2.

Generating a GSD file

5

The GSDML schema files can be useful to generate a GSD file "from scratch". The rules contained regarding the syntax of a GSDML-based file shall suffice to generate a correct GSD file. Additionally, as mentioned in Chapter 2, those rules can be used to validate a GSD file.

However, it is easier to start with an existing GSD file and adapt it to the related field devices. A basis to start is provided, for example, by the sample files of the PNO that can be downloaded from the PNO website. See also Section 2.2.

If an existing GSD file is used, the following adaptations must be made:

- The filename must be structured according to Chapter 3.
- As VendorID (manufacturer identifier), enter the number that is defined for your company at the PNO. An overview is provided on page http://www.profibus.com/IM/Man_ID_Table.xml.

Please note that you must enter the number in the GSDML in hexadecimal format.

- Change the DeviceID to the ID of your device family.
- If needed, adapt the "MainFamily" attribute of the element "Family" to represent a rough technological classification of your field device. This attribute is used to structure the hardware catalog in the engineering tool.
- In the attribute "ProductFamily" of the element "Family", enter the name of your product family.

Figure 5.1 shows the locations that are to be modified in the GSD file.

```
<ProfileBody>
    <DeviceIdentity VendorID="0xFFFF" DeviceID="0x0096">
        <InfoText TextId="IDT_INFO_Device"/>
        <VendorName Value="PROFIBUS & PROFINET International"/>
    </DeviceIdentity>
    <DeviceFunction>
        <Family MainFamily="I/O" ProductFamily="PNO GSDML v2.43 Examples"/>
```

Fig. 5.1: **VendorID, DeviceID and product information in the GSD**

Configuration example: Configuring a device identification

The structure of a DeviceID can be specified within the company but must be uniform within the same company. The DeviceID must uniquely designate the hardware of a specific manufacturer. Therefore, coordination with the marketing and possibly sales departments might be required.

Example:

Vendor_ID		Device_ID	
		Device Class	Device Family
00	**2A**	Byte	Byte
Example for	Siemens	00= Development Kit	00= reserved 01= ERTEC 200 DevKit 02= ERTEC 400 DevKit 03= …
		01= Controllers	01= S7 300 02= S7 400 03= …

Fig. 5.2: **Device identification**

Device Access Point list

6

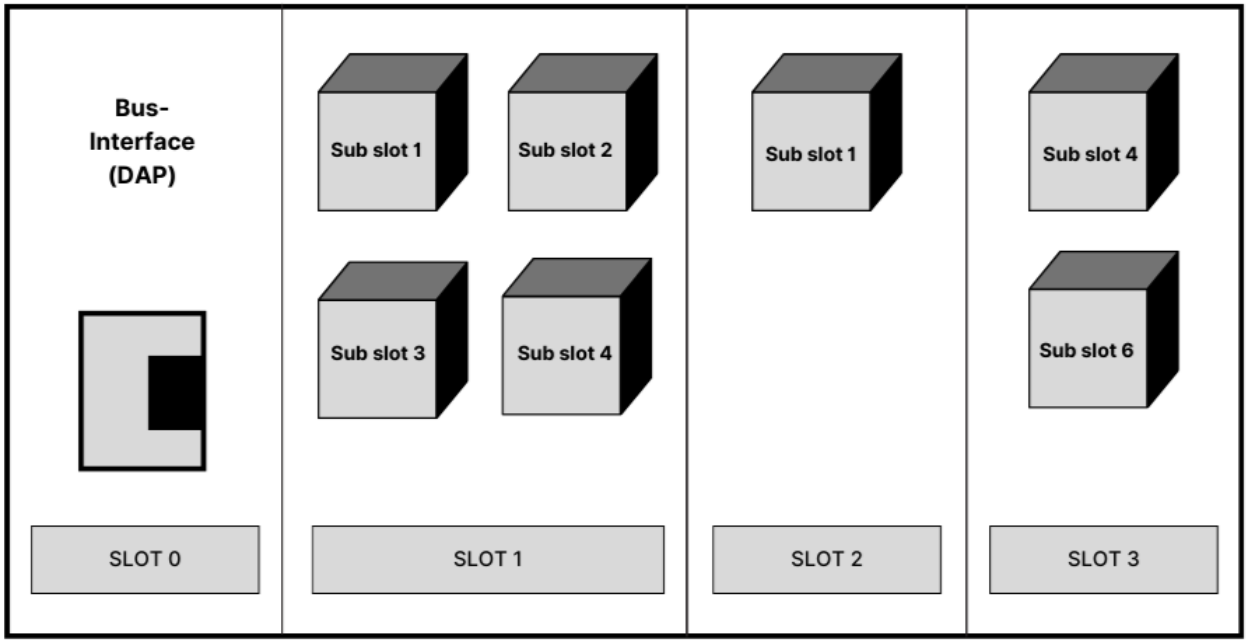

Fig. 6.1: **Structure of a modular field device with the associated modules**

The definitions related to the DAP start within the ApplicationProcess[9]. As mentioned in *Section DeviceAccessPointList*, a DAP represents a network interface and describes the essential features of an PROFINET device. Therefore, the technical capabilities of an PROFINET device and its modules together with their attributes must be described in the GSD file.

The following articles are always associated with a DAP:

- Physical Device Management (PDEV) contains all hardware-oriented definitions. See also Chapter 7.
- Supported Real-time classes
- Redundancy definitions
- Port definitions for each switch port

Examples of these definitions are provided in the respective chapters. For the sake of completeness, an example is given below.

Example: Below, a DAP based on the ERTEC 200P is defined. Since ERTEC 200P has two switch ports, they also must be defined within the DAP with the associated subslot numbers.

```
<DeviceAccessPointItem ID="DAP 1" PhysicalSlots="0..16" ModuleIdentNumber="0x00000001"
   MinDeviceInterval="4" ImplementationType="ERTEC200P" DNS_CompatibleName="ERTEC-DEVKit"
```

(continues on next page)

[9] I.e., the parent element of the "DeviceAccessPointList" is the element "ApplicationProcess".

(continued from previous page)

```
   FixedInSlots="0" ObjectUUID_LocalIndex="1" MultipleWriteSupported="true"
   SharedDeviceSupported="true" DeviceAccessSupported="true" NumberOfDeviceAccessAR="1"
   MaxSupportedRecordSize="8192" NameOfStationNotTransferable="true"
   ParameterizationSpeedupSupported="false" LLDP_NoD_Supported="true" ResetToFactoryModes="2"
   CheckDeviceID_Allowed="true" PowerOnToCommReady="490" IOXS_Required="false"
   RequiredSchemaVersion="V2.4" PNIO_Version="V2.43" PrmBeginPrmEndSequenceSupported="true"
   CIR_Supported="true" AddressAssignment="LOCAL;DCP">
   <ModuleInfo CategoryRef="ID_ERTEC200PDEVKit">
      <Name TextId="TOK_Standard_MRP_S2_DR" />
      <InfoText TextId="TOK_ModInfo_InfoTextId_DAP1" />
      <VendorName Value="SIEMENS" />
      <OrderNumber Value="6ES7 195-3BE00-0YA0" />
   </ModuleInfo>
   <CertificationInfo ConformanceClass="C"
      ApplicationClass="Isochronous;HighPerformance;HighAvailability"
      NetloadClass="III" />
   <SubslotList>
      <SubslotItem SubslotNumber="32768" TextId="TOK_Subslot_8000" />
      <SubslotItem SubslotNumber="32769" TextId="TOK_Subslot_8001" />
      <SubslotItem SubslotNumber="32770" TextId="TOK_Subslot_8002" />
   </SubslotList>
   <IOConfigData MaxInputLength="1440" MaxOutputLength="1440" MaxDataLength="2880" />
   <UseableModules>
      <ModuleItemRef ModuleItemTarget="ID_Mod_01" AllowedInSlots="1..16" />
      <ModuleItemRef ModuleItemTarget="ID_Mod_02" AllowedInSlots="1..16" />
      <ModuleItemRef ModuleItemTarget="ID_Mod_03" AllowedInSlots="1..16" />
      <ModuleItemRef ModuleItemTarget="ID_Mod_10" AllowedInSlots="1..16" />
      <ModuleItemRef ModuleItemTarget="ID_Mod_11" AllowedInSlots="1..16" />
      <ModuleItemRef ModuleItemTarget="ID_Mod_12" AllowedInSlots="1..16" />
      <ModuleItemRef ModuleItemTarget="ID_Mod_16" AllowedInSlots="1..16" />
      <ModuleItemRef ModuleItemTarget="ID_Mod_17" AllowedInSlots="1..16" />
      <ModuleItemRef ModuleItemTarget="ID_Mod_18" AllowedInSlots="1..16" />
      <ModuleItemRef ModuleItemTarget="ID_Mod_50" AllowedInSlots="1..16" />
      <ModuleItemRef ModuleItemTarget="ID_Mod_51" AllowedInSlots="1..16" />
      <ModuleItemRef ModuleItemTarget="ID_Mod_52" AllowedInSlots="1..16" />
      <ModuleItemRef ModuleItemTarget="ID_Mod_53" AllowedInSlots="1..16" />
      <ModuleItemRef ModuleItemTarget="ID_Mod_60" AllowedInSlots="1..16" />
   </UseableModules>
   <VirtualSubmoduleList>
      <VirtualSubmoduleItem ID="DAP 1" SubmoduleIdentNumber="0x0001" Writeable_IM_Records="1 2 3
↪4" IM5_Supported="true" MayIssueProcessAlarm="false">
         <IOData />
         <ModuleInfo>
            <Name TextId="TOK_Standard_MRP_S2_DR" />
            <InfoText TextId="TOK_ModInfo_InfoTextId_DAP1" />
         </ModuleInfo>
      </VirtualSubmoduleItem>
   </VirtualSubmoduleList>
   <SystemDefinedSubmoduleList>
      <!-- Here, the PDEV definitions are located. Please refer to "Definition of the PDEV"
↪section. -->
   </SystemDefinedSubmoduleList>
</DeviceAccessPointItem>
```

Definition of the PDEV

7

Note: Substantial parts of this chapter are taken from *[2]*.

In the case of PROFINET, the physical interfaces of a field device and their switch ports constitute the PDEV. An Ethernet interface is uniquely identified by its name, IP and MAC address, and it can consist of several switch ports.

In a field device, all hardware-oriented port information and transmission parameters are stored retentively in the PDEV. These are organized as follows:

- **Interface Submodule** corresponds to an instance. For example, the planning data for IRT communication, clock synchronization, media redundancy, etc., are stored here.
- **Port Submodule** stores the data for the port settings, neighborhood information, fiber optic parameters, etc.

The subslot numbers are defined for the respective DAP according to the entry in the GSD file. These numbers start with 0x8000 for the interface subslot. The allocation is organized as follows:

Subslot number 0x8i00> i = number of the interface module (0 . . . 0x0F) and

Subslot number 0x8ipp> pp = number of the respective port (1 . . . 0x0FF)

The addressing of the PDEV is defined in the GSD file of the device. Below, you see an example of how the definition in the GSD file can appear.

Example

The PDEV definition is located within the SystemDefinedSubmoduleList[10] and belongs to the respective DAP. To address the PDEV, Subslot 0x8000 (32768 in decimal) is selected for the example below. This example also includes the supported Real-time Classes, the IRT definition and definitions for Fast Startup. These definitions will be discussed in greater detail in the corresponding chapters. Yet, for the sake of clarity, they are included in the example below.

```
<SystemDefinedSubmoduleList>
   <InterfaceSubmoduleItem ID="IDS_1I" SubslotNumber="32768" SubmoduleIdentNumber="0x0002"
      IsochroneModeSupported="true" IsochroneModeInRT_Classes="RT_CLASS_3"
      SupportedRT_Classes="RT_CLASS_1;RT_CLASS_3" TextId="TOK_DAP_InterfaceModule"
      SupportedProtocols="SNMP;LLDP" NetworkComponentDiagnosisSupported=\"true\""
      PTP_BoundarySupported="true" DCP_BoundarySupported="true"
      MulticastBoundarySupported="true" DCP_HelloSupported="true" MaxFrameStartTime="1600"
      MinNRT_Gap="960" DelayMeasurementSupported="true" Writeable_IM_Records="1 2 3 4"
      IM5_Supported="true" PDEV_CombinedObjectSupported="true"
      SupportedDelayMeasurements="PTCP">
      <RT_Class3Properties ForwardingMode="Absolute;Relative" MaxBridgeDelay="2088"
```

(continues on next page)

[10] The element "SystemDefinedSubmoduleList" contains all types of submodules whose structure is defined in PNO-2.712, Application Layer Services, "Physical Device Management ASE".

(continued from previous page)

```
        MaxBridgeDelayFFW="1024" MaxNumberIR_FrameData="1024" MaxRangeIR_FrameID="1024"
        StartupMode="Legacy;Advanced" MaxDFP_Frames="1" MaxDFP_Feed="340"
        AlignDFP_Subframes="false" FragmentationType="Dynamic" MaxRedPeriodLength="4000"
        MinFSO="1760" MinRTC3_Gap="1120" MinYellowTime="10240" YellowSafetyMargin="160"
        MaxRetentionTime="262000"/>
     <SynchronisationMode SupportedRole="SyncSlave" MaxLocalJitter="300" T_PLL_MAX="1000"
        SupportedSyncProtocols="PTCP" PeerToPeerJitter="250" />
     <ApplicationRelations NumberOfAR="4" StartupMode="Legacy;Advanced"
        NumberOfAdditionalInputCR="0" NumberOfAdditionalMulticastProviderCR="0"
        NumberOfAdditionalOutputCR=0 NumberOfMulticastConsumerCR=0
        PullModuleAlarmSupported="true">
        <TimingProperties ReductionRatio="1 2 4 8 16 32 64 128 256 512" SendClock="8 16 32 64␣
↪128">
        <RT_Class3TimingProperties ReductionRatioPow2="2 4 8 16" SendClock="4 6 8 12 16 20 24␣
↪28 32 36 40 44 48 52 56 60 64 68 72 76 80 84 88 92 96 100 104 108 112 116 120 124 128"␣
↪ReductionRatio="1" />
     </ApplicationRelations>
     <MediaRedundancy SupportedRole="Client" MRPD_Supported="true"
        AdditionalForwardingRulesSupported="true" />
  </InterfaceSubmoduleItem>
  <PortSubmoduleItem ID="IDS_1P1" SubslotNumber="32769" SubmoduleIdentNumber="0x0003"
     TextId="TOK_Port1" MaxPortRxDelay="198" MaxPortTxDelay="6" IsDefaultRingport="true"
     MAUType="100BASETXFD" MAUTypes="16" PortDeactivationSupported="true"
     LinkStateDiagnosisCapability="Up+Down" ShortPreamble100MBitSupported="true"
     CheckMAUTypeSupported="true" CheckMAUTypeDifferenceSupported="true"
     Writeable_IM_Records="1 2 3 4" IM5_Supported="true">
     <MAUTypeList>
        <MAUTypeItem Value="16" AdjustSupported="true" />
     </MAUTypeList>
  </PortSubmoduleItem>
  <PortSubmoduleItem ID="IDS_1P2" SubslotNumber="32770" SubmoduleIdentNumber="0x0003"
     TextId="TOK_Port2" MaxPortRxDelay="198" MaxPortTxDelay="6" IsDefaultRingport="true"
     MAUType="100BASETXFD" MAUTypes="16" PortDeactivationSupported="true"
     LinkStateDiagnosisCapability="Up+Down" ShortPreamble100MBitSupported="true"
     CheckMAUTypeSupported="true" CheckMAUTypeDifferenceSupported="true"
     Writeable_IM_Records="1        2 3 4" IM5_Supported="true">
     <MAUTypeList>
        <MAUTypeItem Value="16" AdjustSupported="true" />
     </MAUTypeList>
  </PortSubmoduleItem>
</SystemDefinedSubmoduleList>
```

Definition of Real-time Classes

8

In order to adapt the communication optimally to the needs of the automation plant, the Real-time Classes for data exchange have been defined in PROFINET. In the GSD files, Real-time Classes supported by the respective field devices can be specified. Definition of Real-time Classes are DAP-specific and must be specified in the PDEV. The attribute "SupportedRT_Classes"[11] of the element "InterfaceSubmoduleItem" defines all Real-time Classes supported by the interface.

PROFINET uses the PTCP protocol for synchronization. Since a synchronization frame is sent over the entire bus to all stations, the time delay must also be specified when a frame passes through a switchport in send and receive directions.

In the GSD file, the definition of the Real-time Classes and synchronization information shall look like the following:

```
<InterfaceSubmoduleItem ID="IDS_1I" SubslotNumber="32768" SubmoduleIdentNumber="0x0002"
   IsochroneModeSupported="true" IsochroneModeInRT_Classes="RT_CLASS_3"
   SupportedRT_Classes="RT_CLASS_1;RT_CLASS_3" TextId="TOK_DAP_InterfaceModule"
   SupportedProtocols="SNMP;LLDP" NetworkComponentDiagnosisSupported="true"
   PTP_BoundarySupported="true" DCP_BoundarySupported="true" MulticastBoundarySupported="true"
   DCP_HelloSupported="true" MaxFrameStartTime="1600" MinNRT_Gap="960"
   DelayMeasurementSupported="true" Writeable_IM_Records="1 2 3 4" IM5_Supported="true"
   PDEV_CombinedObjectSupported="true" SupportedDelayMeasurements="PTCP">
   <RT_Class3Properties ForwardingMode="Absolute;Relative" MaxBridgeDelay="2088"
      MaxBridgeDelayFFW="1024" MaxNumberIR_FrameData="1024" MaxRangeIR_FrameID="1024"
      StartupMode="Legacy;Advanced" MaxDFP_Frames="1" MaxDFP_Feed="340"
      AlignDFP_Subframes="false" FragmentationType="Dynamic" MaxRedPeriodLength="4000"
      MinFSO="1760" MinRTC3_Gap="1120" MinYellowTime="10240" YellowSafetyMargin="160"
      MaxRetentionTime="262000" />
   <SynchronisationMode SupportedRole="SyncSlave" MaxLocalJitter="300" T_PLL_MAX="1000"
      SupportedSyncProtocols="PTCP" PeerToPeerJitter="250" />
   <ApplicationRelations NumberOfAR="4" StartupMode="Legacy;Advanced"
      NumberOfAdditionalInputCR="0" NumberOfAdditionalMulticastProviderCR="0"
      NumberOfAdditionalOutputCR="0" NumberOfMulticastConsumerCR="0"
      PullModuleAlarmSupported="true">
      <TimingProperties ReductionRatio="1 2 4 8 16 32 64 128 256 512" SendClock="8 16 32 64 128"␣
↪/>
      <RT_Class3TimingProperties ReductionRatioPow2="2 4 8 16" SendClock="4 6 8 12 16 20 24 28
         32 36 40 44 48 52 56 60 64 68 72 76 80 84 88 92 96 100 104 108 112 116 120 124 128"
         ReductionRatio="1" />
   </ApplicationRelations>
   <MediaRedundancy SupportedRole="Client" MRPD_Supported="true"
      AdditionalForwardingRulesSupported="true" />
</InterfaceSubmoduleItem>
```

[11] For the GSDs of version <= V2.1 SupportedRT_Class is used.

Modules, submodules, slots and subslots 9

It is possible to define different modules and submodules in a GSD file. They are defined only once within a GSD file and if needed, they can be assigned to different DAPs. The modules/submodules define the physical configuration of an electronic base module. Each of them is uniquely referenced with an eight-digit ident number. For that reason, the module and submodule numbers must be unique, i.e., they must be assigned only once. For actual operation, the modules/submodules are assigned to the physical slots of a device. It is also possible to assign a module/submodule to several slots. The I/O data is always addressed by means of a slot-subslot combination.

The combination consisting of DeviceID, ModuleIdentNumber and SubmoduleIdentNumber must be unique for a vendor ID.

Since it must be possible to identify a hardware uniquely from the point of view of the engineering tool as well as of the process-oriented field device, the following rules are applied to the usage of ident numbers:

- Modules within a field device are identified by unique non-interchangeable ident numbers.
- Submodules within a module are identified by unique submodule ident numbers.

Note: If a module from the same or a different device family is used in several field devices, the ModuleIdentNumber shall be assigned uniquely across all these devices.

If a submodule is used in several modules, the SubmoduleIdentNumber shall be assigned uniquely across all these modules. The following applies to such a SubmoduleIdentNumber:

- The submodules identify not only the physical properties, but also the representation of the cyclical I/O data. If submodules of different length in data exist, different SubmoduleIdentNumbers must be used for different representations (length and data format). Only in this way it is possible to uniquely identify physically identical submodules within a module, i.e., via different representation of the data.
- The VendorID, DeviceID and the ModuleIdentNumber represent a one-to-one relationship, i.e., there can never be more than one ModuleIdentNumber for the same hardware. A submodule on the other hand can have several unique SubmoduleIdentNumbers for different representations of the I/O data.

Deviation from these rules is allowed only if,

- Existing modular field devices become PROFINET devices provided that for this transition only the head module must be exchanged.
- Gateways (links) to other networks are used. Usually, the field devices of the second network are mapped to the PROFINET address space.

Submodule definitions can occur at different locations within a GSD file. Some of these locations are listed below:

- InterfaceSubmodules and PortSubmodules within a DAP in the SystemDefinedSubmoduleList

- PortSubmodules as so-called "normal" modules in the SystemDefinedSubmoduleList or SubmoduleList (pluggable[12] PortSubmodules)
- "Normal" modules within the VirtualSubmoduleList[13]
- "Normal" submodules within the global SubmoduleList

All these identification numbers can be read out as follows:

- DCP Identify and RPC EndPointMapper (VendorID and DeviceID)
- I&M functions (VendorID)
- Acyclical reading with RealIdentificationData (ModuleIdentNumber and SubmoduleIdentNumber)

Example:

In the example below, several modules are defined that can be read out based on their ID (for instance, in this example the ID is "ID_Mod_01"). They are defined in two steps.

1. Define the list of the modules within the "UseableModules" element. All modules are identified with their unique IDs.

```
<UseableModules>
   <ModuleItemRef ModuleItemTarget="ID_Mod_01" AllowedInSlots="1..16"/>
   <ModuleItemRef ModuleItemTarget="ID_Mod_02" AllowedInSlots="1..16"/>
   <ModuleItemRef ModuleItemTarget="ID_Mod_03" AllowedInSlots="1..16"/>
   <ModuleItemRef ModuleItemTarget="ID_Mod_10" AllowedInSlots="1..16"/>
   <ModuleItemRef ModuleItemTarget="ID_Mod_11" AllowedInSlots="1..16"/>
   <ModuleItemRef ModuleItemTarget="ID_Mod_12" AllowedInSlots="1..16"/>
   <ModuleItemRef ModuleItemTarget="ID_Mod_16" AllowedInSlots="1..16"/>
   <ModuleItemRef ModuleItemTarget="ID_Mod_17" AllowedInSlots="1..16"/>
   <ModuleItemRef ModuleItemTarget="ID_Mod_18" AllowedInSlots="1..16"/>
   <ModuleItemRef ModuleItemTarget="ID_Mod_50" AllowedInSlots="1..16"/>
   <ModuleItemRef ModuleItemTarget="ID_Mod_51" AllowedInSlots="1..16"/>
   <ModuleItemRef ModuleItemTarget="ID_Mod_52" AllowedInSlots="1..16"/>
   <ModuleItemRef ModuleItemTarget="ID_Mod_53" AllowedInSlots="1..16"/>
   <ModuleItemRef ModuleItemTarget="ID_Mod_60" AllowedInSlots="1..16"/>
</UseableModules>
```

2. Define the properties of the modules within the "ModuleList" element with reference to the ModuleItem ID. The actual data length is specified within the VirtualSubmodule definition. It is defined by the attribute "Length" of the child element "DataItem".

```
<ModuleList>
   <ModuleItem ID="ID_Mod_01" ModuleIdentNumber="0x00000020">
      <ModuleInfo>
         <Name TextId="TOK_TextId_Module_1IO"/>
         <InfoText TextId="TOK_InfoTextId_Module_1IO"/>
   </ModuleInfo>
   <VirtualSubmoduleList>
```

(continues on next page)

[12] I.e., modular.

[13] The term **virtual** implies that the submodule is not present physically; but it can be addressed logically within a real module in order to assign process data to the corresponding modules *[2]*.

(continued from previous page)

```
      <VirtualSubmoduleItem ID="1" SubmoduleIdentNumber="0x00000001" Writeable_IM_Records="1 2 3␣
↪4" MayIssueProcessAlarm="true">
         <IOData>
            <Input Consistency="All items consistency">
               <DataItem DataType="OctetString" TextId="TOK_Input_DataItem_1" Length="1"     ␣
↪UseAsBits="true"/>
            </Input>
         <Output Consistency="All items consistency">
            <DataItem DataType="OctetString" TextId="TOK_Output_DataItem_1" Length="1" UseAsBits=
↪"true"/>
         </Output>
         </IOData>
         <RecordDataList>
            <ParameterRecordDataItem Index="1" Length="4" TransferSequence="0">
               <Name TextId="T_general_parameter"/>
               <Const Data="0x20,0x01,0x56,0x78" ByteOffset="0"/>
            </ParameterRecordDataItem>
         </RecordDataList>
         <ModuleInfo>
            <Name TextId="TOK_TextId_Module_1IO"/>
            <InfoText TextId="TOK_InfoTextId_Module_1IO"/>
         </ModuleInfo>
         <PROFIenergy ProfileVersion="V1.3" EntityClass="Class3" EntitySubclass="Subclass1" >
            <EnergySavingModeList>
               <EnergySavingModeItem ID="1" TimeToPause="600" RTTO="600" TimeMinLengthOfStay="200
↪"/>
               <EnergySavingModeItem ID="2" TimeToPause="600" RTTO="600" TimeMinLengthOfStay="200
↪"/>
               <EnergySavingModeItem ID="255" TimeToPause="600" RTTO="600" TimeMinLengthOfStay=
↪"200"/>
            </EnergySavingModeList>
            <MeasurementList>
               <MeasurementItem Number="0">
                  <MeasurementValue ID="13" AccuracyDomain="1" AccuracyClass="3"/>
               </MeasurementItem>
            <MeasurementItem Number="1">
                  <MeasurementValue ID="14" AccuracyDomain="1" AccuracyClass="3"/>
               </MeasurementItem>
               <MeasurementItem Number="2">
                  <MeasurementValue ID="15" AccuracyDomain="1" AccuracyClass="3"/>
               </MeasurementItem>
            </MeasurementList>
         </PROFIenergy>
      </VirtualSubmoduleItem>
   </VirtualSubmoduleList>
   </ModuleItem>
      Here, additional module definitions follow.
</ModuleList>
```

9.1 Definition of modules with isochronous mode capability

For the example below, suppose that the module "ID_Mod_03" supports isochronous mode. In that case, the definition of this module must contain the XML element "IsochroneMode"[14].

Example:

```
<ModuleItem ID="ID_Mod_03" ModuleIdentNumber="0x00000003">
   <ModuleInfo>
      <Name TextId="TOK_TextId_Module_1I_Isochron"/>
      <InfoText TextId="TOK_InfoTextId_Module_1I_Isochron"/>
   </ModuleInfo>
   <VirtualSubmoduleList>
      <VirtualSubmoduleItem ID="03" SubmoduleIdentNumber="0x00000001" API="0">
         <IOData>
            <Input Consistency="All items consistency">
               <DataItem DataType="OctetString" TextId="TOK_Inp_DataItem_1_Isochron" Length="1"␣
↪UseAsBits="true"/>
            </Input>
         </IOData>
         <ModuleInfo>
            <Name TextId="TOK_TextId_Module_1I_Isochron"/>
            <InfoText TextId="TOK_InfoTextId_Module_1I_Isochron"/>
         </ModuleInfo>
         <IsochroneMode IsochroneModeRequired="true" T_DC_Base="8" T_DC_Max="128" T_DC_Min="8" T_
↪IO_Base="125000" T_IO_InputMin="1" T_IO_OutputMin="1"/>
      </VirtualSubmoduleItem>
   </VirtualSubmoduleList>
</ModuleItem>
```

[14] Note that the attribute "IsochroneModeRequired" indicates whether the local isochronous application requires the PROFINET controller to support and use isochronous mode. If the attribute is set to "true", the local isochronous application requires an PROFINET controller that supports and uses isochronous mode, otherwise it cannot be operated.

Diagnostic definitions

10

In PROFINET, alarms are used to signal the critical states of the process(es) or the field device(s) to the controller in the form of diagnosis. The diagnostic information can be classified into three groups:

- System-defined diagnosis
- Profile-specific diagnosis
- Manufacturer-specific diagnosis

In order to display diagnostic information in a human-readable form, an engineering tool needs to know the structure of the diagnosis and additional texts. The profile- and manufacturer-specific diagnoses shall be described in the GSD file of the device. The system-defined diagnoses are considered well-known to the engineering tool with regards to the structure as well as the additional texts. To put it differently, their structure and additional texts are to be implemented in the engineering tool[15]. Therefore, a change in system-defined diagnoses requires an update of the engineering tool.

The diagnosis types that are defined by different elements can be found in the GSDML Specification[16].

Example:

Manufacturer-specific channel diagnosis is described in the element "ChannelDiagItem" whose parent is the "ChannelDiagList" element. The "ErrorType" attribute of the "ChannelDiagItem" contains an integer in the range of 15 to 32767 to which the diagnostic text definitions are assigned. The "TextId" attribute of the "Name" element whose parent is "ChannelDiagItem" contains the ID which references the related language-dependent text information in the "ExternalTextList" element. Please note that "ErrorType" should be unique across all "ChannelDiagItem" and "SystemDefinedChannelDiagItem" elements. Please consider the excerpts below:

```
<ChannelDiagList>
   <ChannelDiagItem ErrorType="16">
      <Name TextId="TOK_Name_ErrorType16"/>
      <Help TextId="TOK_HelpName_ErrorType16"/>
   </ChannelDiagItem>
   <ChannelDiagItem ErrorType="17">
      <Name TextId="TOK_Name_ErrorType17"/>
      <Help TextId="TOK_HelpName_ErrorType17"/>
   </ChannelDiagItem>
   <ChannelDiagItem ErrorType="18">
      <Name TextId="TOK_Name_ErrorType18"/>
      <Help TextId="TOK_HelpName_ErrorType18"/>
   </ChannelDiagItem>
<ChannelDiagList>
```

Additionally, we recommend that the "Text" attribute be kept at less than 400 characters, so that the users can read the complete text without scrolling.

[15] A guideline for the texts for all system-defined and profile-specific diagnosis in several languages can be found in the Diagnosis for PROFINET Guideline *[4]*.

[16] See Annex "PROFINET diagnosis".

```
<ExternalTextList>
   <PrimaryLanguage>
      <Text TextId="TOK_Name_ErrorType16" Value="parameter assignment error"/>
      <Text TextId="TOK_Name_ErrorType17" Value="Power supply fault"/>
      <Text TextId="TOK_Name_ErrorType18" Value="fuse blown"/>
   <PrimaryLanguage>
<ExternalTextList>
```

Graphic symbols

11

When configuring PROFINET systems, the field devices are represented by graphic symbols. The element "Graphics" contains a list of "GraphicItemRef" elements that references graphical information about a device, module or submodule. Below is an example of the "Graphics" element and, right below the text, an example of the referenced graphics (for ERTEC 200P Development Kit).

```
<Graphics>
   <GraphicItemRef Type="DeviceSymbol" GraphicItemTarget="ID_Graph_1"/>
</Graphics>
```

Texts in GSDML

12

12.1 General

Most texts in the GSDs can be defined in different languages. To this end, only "Ids" are used in the description part (designated as attribute with the name "TextId") that refer to the actual texts in the "ExternalTextList". There are no length limitations for the texts themselves, since as a rule it is possible to respond in the engineering tools[17] to texts of any length with scroll bars, for example. Nevertheless, it is pointed out that if texts are overlong, it can happen that the entire text is not displayed.

For that reason, it is recommended to stay within the following length definitions.

Table 12.1: Recommended text lengths

Use	Element name	Recommended maximum length in octets
Category names	CategoryItem	30
Parameter names	ValueItem	30
Parameter texts	Assign	-
Designating cyclical IO data	DataItem	30
Help information	Help	1024
Manufacturer specific diagnosis: Designation	ChannelDiagItem/Name	30
Manufacturer specific diagnosis: Help text	ChannelDiagItem/Text	1024
Name of a parameter assignment data set	ParameterRecordDataItem	30
Name of a module	ModuleInfo/Name	30
General information about a module	ModuleInfo/InfoText	250
General information about a PROFINET device	DeviceIdentity/InfoText	250

[17] Including HMI/SCADA or display panel of a controller itself.

12.2 Incorporating foreign languages

All language-dependent texts are stored in the section "ExternalTextList" of a GSD file. The "ExternalTextList" includes at least the section "PrimaryLanguage" that must contain all texts in **English** (en).

In addition, any number of "Language" sections is possible where the texts of another language are stored respectively. The language itself is defined with the attribute "xml:lang" and follows the rules of "two-letter codes" of the ISO 639-1:2002, Codes for the representation of names of languages - Part 1: Alpha-2 code. Table 12.2 shows the encodings for some languages.[18]

Table 12.2: Language encodings in the GSDML

Abbreviation	Language
en	English
de	German
es	Spanish
fr	French
it	Italian

The sections "PrimaryLanguage" and "Language" have identical structures. They include any number of "Text" entries. The attribute "TextId" is used for identification and the attribute "Value" contains the actual text.

An "ExternalTextList" with two languages shall look like this:

```
<ExternalTextList>
   <PrimaryLanguage>
      <Text TextId="TOK_DevIdent_InfoText" Value="ERTEC 200P Evaluation Kit"/>
      <Text TextId="TOK_Category_ERTEC200PDEVKit" Value="ERTEC 200P Evaluation Kit"/>
      <Text TextId="TOK_Standard_no_PDEV" Value="Standard, no PDEV"/>
      <Text TextId="TOK_Standard_no_MRP" Value="Standard, no MRP"/>
      ...
      <Text TextId="TOK_Name_ErrorType16" Value="parameter assignment error"/>
      <Text TextId="TOK_Name_ErrorType17" Value="Power supply fault "/>
      ...
   </PrimaryLanguage>
   <Language xml:lang="de">
      <Text TextId="TOK_DevIdent_InfoText" Value="ERTEC 200P Evaluation Kit"/>
      <Text TextId="TOK_Category_ERTEC200PDEVKit" Value="ERTEC 200P Evaluation Kit"/>
      <Text TextId="TOK_Standard_no_PDEV" Value="Standard, kein PDEV"/>
      <Text TextId="TOK_Standard_no_MRP" Value="Standard, kein MRP"/>
      ...
      <Text TextId="TOK_Name_ErrorType16" Value="Parametrierfehler"/>
      <Text TextId="TOK_Name_ErrorType17" Value="Stromversorgungsfehler"/>
      ...
   </Language>
</ExternalTextList>
```

[18] For a complete set, refer to *[5]*.

In addition to this method, it is also possible to move texts to an external language file. This allows for adding a language without the GSD file having to be changed.

An external language file is structured as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<ExternalTextDocument xml:lang="de" xmlns="http://www.profibus.com/GSDML/2003/11/Primitives"
   xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
   xsi:schemaLocation="http://www.profibus.com/GSDML/2003/11/Primitives ..\..\xsd\GSDML-
↪Primitives-V2.43.xsd">
   <Text TextId="IDT_INFO_Device" Value="PNO GSDML V2.43 Beispiele"/>
   <Text TextId="IDT_NAME_DAP1" Value="Sprachbeispiel DAP"/>
   <Text TextId="IDT_INFO_DAP1" Value="Beschreibung des Sprachbeispiels DAP"/>
   <Text TextId="IDT_SUBSLOT_1" Value="Subslot 1"/>
   <Text TextId="IDT_SUBSLOT_Interface" Value="Schnittstelle"/>
   <Text TextId="IDT_SUBSLOT_Port1" Value="Anschluss 1"/>
   <Text TextId="IDT_SUBSLOT_Port2" Value="Anschluss 2"/>
   <Text TextId="IDT_RECORD_NAME_Head" Value="Parameter der Kopfbaugruppe"/>
   ...
   <Text TextId="IDT_NAME_PS1" Value="P1"/>
   <Text TextId="IDT_NAME_PS2" Value="P2"/>
</ExternalTextDocument>
```

External filenames must conform to certain name conventions. The rule is that the corresponding GSD filename must be appended with the string "-Text-" and the ISO 639-1:2002 compliant two-letter code. For the two-letter codes, see Table 12.2.

Example:

If GSD filename is:

`GSDML-V1.0-Siemens-ET200X-20030818.xml`

External language filename in Italian must be:

`GSDML-V1.0-Siemens-ET200X-20030818-Text-it.xml`

Using references

13

In a GSD file, numerous references are used in order to avoid describing the same data multiple times. For example, suppose that two modules support the same parameter "MeasuringRange". In addition to the textual explanations, numerous parameters are stored in the GSD for this parameter (such as encoding). By using references, it is possible to describe the parameter "MeasuringRange" at one location in the GSD and to reference it from all modules that are using this parameter. Otherwise, all texts would have to be specified separately in all languages for each module. This would not only increase the effort to update the file but also would increase considerably the size of the GSD file.

13.1 Checks using the schema

IDs are used to reference elements in the GSD. In the example below, the ID of a module is shown:

```
<ModuleItem ID="IDM_1" ModuleIdentNumber="0x00000020">
```

To reference this module, it is enough to specify the ID of this module:

```
<ModuleItemRef ModuleItemTarget="IDM_1" AllowedInSlots="1..16"/>
```

Referencing only works if the IDs are kept unique. This is checked and ensured by means of schema validation. The example below shows the error message of PROFINET GSD Checker for duplicate values, i.e., there are two modules with the same ID "IDM_1":

Fig. 13.1: **Error messages if reference is invalid**

13.2 Tips for naming IDs meaningfully

A GSD file can become quite large. Therefore, the many IDs must be assigned in a consistent manner. When the IDs are assigned, the GSDML schema itself only checks whether the string representing the assigned ID has leading or trailing blanks. Those are detected through schema validation and rejected if the string representing the assigned ID has leading or trailing blanks. Likewise, line breaks are not permitted in a string representing the assigned ID.

Otherwise, any string can be used for IDs.

To avoid confusion, it is recommended to start the IDs with a type-specific prefix:

- The string indicates which object the ID references
- Unique ID assignment is facilitated
- In an editor, the IDs can be displayed sorted in a list

Table 13.1 provides recommendations for assigning ID prefixes.

Table 13.1: ID prefix recommendations

Object	**Prefix**
DeviceAccessPointItem	IDD_
Module	IDM_
Submodule	IDS_
ValueItem	IDV_
GraphicItem	IDG_
Category	IDC_
InfoText	IDT_INFO_
ChannelDiagItem/Name	IDT_DIAG_NAME_
RecordDataItem/Name	IDT_RECORD_NAME_
ModuleInfo/Name	IDT_MODULE_NAME_
Category	IDT_CATEGORY_
Help	IDT_HELP_
Assign	IDT_ASSIGN_
Ref	IDT_REF_
DataItem	IDT_DATAITEM_

Catalog information in the GSD 14

To facilitate the selection of the desired modules for field devices with many module variants, each module in the GSD can be assigned to a category and a subcategory. This assignment has no effect on the runtime properties of the related modules. The representation of the selected modules in the engineering tool is provided in the form of a catalog.

By using categories, technically related modules can be grouped. For example, all analog input modules can be assigned to the category "Analog Input".

14.1 Category assignment

Each module (including the DAP) in the GSD has a mandatory element "ModuleInfo" which contains textual information for a catalog entry in the engineering tool. The element "ModuleInfo" has two optional attributes "CategoryRef" and "SubCategory1Ref". With these attributes, the assignment to a main category or a subcategory is made. The contents of the attributes must include an ID of the element "CategoryItem" whose parent is "CategoryList".

The "CategoryItem" element has a simple structure. In addition to the ID to be referenced from the attribute "CategoryRef" or "SubCategory1Ref" of the "ModuleInfo" element, it contains a "TextId" which references a text in the "ExternalTextList".

The example below shows the relationships:

```
<ModuleList>
    <ModuleItem ID="IDM_1" ModuleIdentNumber="0x00000002">
        <ModuleInfo CategoryRef="IDC_AnalogModules" SubCategory1Ref="IDC_AnalogInput">
        ...
</ModuleList>
...
<CategoryList>
    <CategoryItem ID="IDC_AnalogModules" TextId="IDT_CATEGORY_AnalogModules"/>
    <CategoryItem ID="IDC_AnalogInput" TextId="IDT_CATEGORY_AnalogIn"/>
    ...
</CategoryList>
...
<ExternalTextList>
    <PrimaryLanguage>
        <Text TextId="IDT_CATEGORY_AnalogModules" Value="Analog Modules"/>
        <Text TextId="IDT_CATEGORY_AnalogIn" Value="Analog Input Modules"/>
        ...
    </PrimaryLanguage>
</ExternalTextList>
```

Description of parameter record data objects 15

GSDML allows describing manufacturer-specific parameters that the PROFINET controller transmits to a submodule of the PROFINET device at startup. For PROFINET, these parameters are stored in the form of record data objects.

To describe this data, the element "ParameterRecordDataItem" whose parent is "RecordDataList" is used in the GSDML.

The individual parameters of a submodule can be distributed over several record data objects. Each record data object (within a submodule) can be uniquely addressed by means of the attribute "Index" of the element "ParameterRecordDataItem" within an application process.

The attribute "TransferSequence" controls the sequence in which the individual ParameterRecordDataObjects, i.e., manufacturer- and profile-specific record data objects, are transferred.

Other attributes and elements of a "ParameterRecordDataItem" in the GSDML are

- Length of the record data object in octets (attribute "Length")
- Name of the record data object (child element "Name"). It is used only for display purposes in the engineering tool.
- Initialization of the content of the record data object (child element "Const")

15.1 "Const" definition of a ParameterRecordDataItem

The content of a "ParameterRecordDataItem" can be initialized by using the "Const" element. If the "Const" element is missing, the record data object is initialized with octets set to zero. A "ParameterRecordDataItem" can contain any number of "Const" definitions. No overlapping between the definitions is allowed.

A "Const" element has two attributes:

```
<Const Data="0x20,0x01,0x56,0x78" ByteOffset="0"/>
```

The attribute "Data" contains the byte values - separated by a comma - of the record data objects in hex representation (starting with "0x").

The optional attribute "ByteOffset" describes the initial byte position to insert the "Data" attribute. If this attribute is missing, byte offset "0" is assumed.

[ByteOffset] + [the number of described bytes of the data element] must not exceed the length of the record data object.

The example below shows the initialization of a 4-byte record data object by means of the "Const" definition.

```
<ParameterRecordDataItem Index="1" Length="3">
   <Name TextId="IDT_RECORD_NAME_GeneralParameter"/>
   <Const Data="0xFF,0x00"/>
   <Const Data="0xAA" ByteOffset="3"/>
</ParameterRecordDataItem>
```

Bit → Byte	7	6	5	4	3	2	1	0
0 [0xFF]	1	1	1	1	1	1	1	1
1 [0x00]	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3 [0xAA]	1	0	1	0	1	0	1	0

Fig. 15.1: **Effect of the "Const" definition in a record data object**

In this example, the value of the byte with offset "2" is not specified by the "Const" definition. In this case, the GSDML specification specifies that these values are filled with "0".

15.2 "Ref" definition of a ParameterRecordDataItem

The individual parameters of a "ParameterRecordDataItem" are described by using the "Ref" element. This element has many attributes that specify the properties of the parameter. The "Ref" definition shall not exceed the length of the record data object. That is, the "ByteOffset" (and "BitOffset") plus the length of the data according to the "DataType" (and "BitLength"/"Length") shall be less than or equal to the value of the "Length" attribute of the "ParameterRecordDataItem" element.

The example below illustrates the use of the "Ref" element.

The record data object described in Figure 15.1 is intended to include a parameter called "MeasuringRange". This parameter, encoded by using two bits, can accept the following values:

- 0...20 mA (encoding 00)
- 4...20 mA (encoding 01)
- 0...10 V (encoding 10)
- +-10 V (encoding 11)

The object shall look like this:

Bit → Byte	7	6	5	4	3	2	1	0
0	1	1	1	1	1	1	1	1
1	0	0	0	MR Bit 1	MR Bit 0	0	0	0
2	0	0	0	0	0	0	0	0
3	1	0	1	0	1	0	1	0

Fig. 15.2: **Effect of the "Ref" definition in a record data object**

In the GSD, this parameter would be described as follows:

```
<ParameterRecordDataItem Index="1"Length="4">
   <Name TextId="IDT_RECORD_NAME_GeneralParameter"/>
   <Const Data="0xFF,0x00"/>
   <Const Data="0xAA" BitOffset="3"/>
<Ref ValueItemTarget="IDV_MeasuringRange" ByteOffset="1" BitOffset="3" DataType="BitArea"␣
 ↪BitLength="2" TextId="IDT_REF_ MeasuringRange"/>
</ParameterRecordDataItem>
<ValueList>
   <ValueItem ID="IDV_ MeasuringRange">
      <Assignments>
         <Assign Content="0" TextId="IDT_ASSIGN_+-10V"/>
         <Assign Content="1" TextId="IDT_ASSIGN_0-20mA"/>
         ...
      </Assignments>
   </ValueItem>
   ...
<ValueList>
<ExternalTextList>
   <PrimaryLanguage>
      ...
      <Text TextId="IDT_RECORD_NAME_GeneralParameter" Value="General Parameter"/>
      <Text TextId="IDT_REF_ MeasuringRange" Value=" Measuring range"/>
      <Text TextId="IDT_ASSIGN_+-10V" Value="+-10 Volt"/>
      <Text TextId="IDT_ASSIGN_0-20mA" Value="0...20 mA"/>
      ...
   <PrimaryLanguage>
<ExternalTextList>
```

Note:

- By means of "ValueItemTarget", a concrete value can be assigned to a text. In the engineering tool, this text is displayed to the user as a selection criterion. If there is no "ValueItemTarget", the tool will provide this parameter as a pure input field; that means, in the case above, only the values "0" to "3"

can be entered.

- By using the attribute "AllowedValues" of the "Ref" element, the value range can be limited. A similar module that is merely capable of measuring current could be limited by assigning "0..1" to this attribute.

If neither "ValueItemTarget" nor "AllowedValues" is available, the value range is limited only by the data type.

Figure 15.3 shows a possible representation of the parameter in an engineering tool.

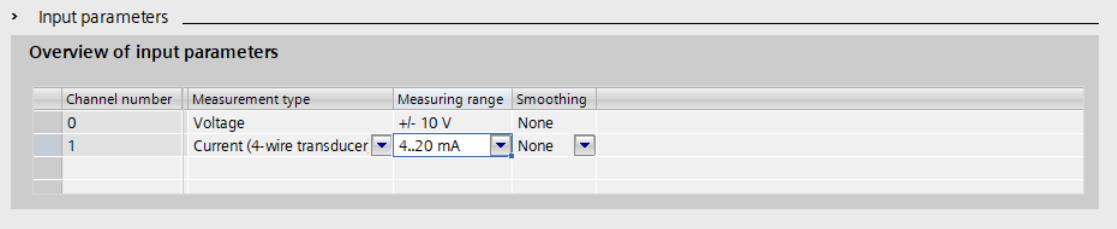

Fig. 15.3: **Representation of a parameter in TIA Portal**

SNMP and MIB2

16

If a field device supports SNMP, an engineering tool can, for example, read out the MIB2 and LLDP library by means of the SNMP protocol. This information is very useful when replacing field devices or displaying the plant topology. The following example shows what the definition shall look like:

```
<InterfaceSubmoduleItem ID="IDS_2I" SubslotNumber="32768" SubmoduleIdentNumber="0x00000002"
IsochroneModeSupported="true" IsochroneModeInRT_Classes="RT_CLASS_3"
SupportedRT_Classes="RT_CLASS_1;RT_CLASS_3" TextId="TOK_DAP_InterfaceModule"
SupportedProtocols="SNMP;LLDP"
NetworkComponentDiagnosisSupported="true"
PTP_BoundarySupported="true" DCP_BoundarySupported="true" MulticastBoundarySupported="true"
DCP_HelloSupported="true" MaxFrameStartTime="1600" MinNRT_Gap="960"
DelayMeasurementSupported="true" Writeable_IM_Records="1 2 3 4" IM5_Supported="true">
```

Fast startup

17

Usually, Ethernet stations need several seconds to access data exchange after power-on. However, in some areas of automation technology, a much faster startup of a field device is desired. This variant of power-on is called Fast Startup (FSU). In this case, the field device becomes active after voltage return and signals the DCP service with the DCP-Hello signal. The FSU definition is DAP-specific and therefore is located within the interface submodule. Definition of an FSU service is accomplished in three steps:

1. Define PowerOnToCommReady within the DAP.

```
<DeviceAccessPointItem ID="DAP 2" PhysicalSlots="0..16" ModuleIdentNumber="0x00000002"
MinDeviceInterval="4" ImplementationType="ERTEC200P" DNS_CompatibleName="ERTEC-DEVKit"
FixedInSlots="0" ObjectUUID_LocalIndex="1" MultipleWriteSupported="true"
SharedDeviceSupported="true" DeviceAccessSupported="true" NumberOfDeviceAccessAR="1"
MaxSupportedRecordSize="8192" NameOfStationNotTransferable="true"
ParameterizationSpeedupSupported="true" LLDP_NoD_Supported="true" ResetToFactoryModes="2"
CheckDeviceID_Allowed="true"
PowerOnToCommReady="490"
```

Time is in ms until the first data exchange.

2. Carry out the PDEV definition (usually SubslotItem SubslotNumber= "32768") for the associated PDEV.

```
<SubslotList>
    <SubslotItem SubslotNumber="32768" TextId="TOK_Subslot_8000"/>
    <SubslotItem SubslotNumber="32769" TextId="TOK_Subslot_8001"/>
    <SubslotItem SubslotNumber="32770" TextId="TOK_Subslot_8002"/>
</SubslotList>
```

3. Specify DCP-Hello support (DCP_HelloSupported="true") for the corresponding subslot of the PDEV.

```
<InterfaceSubmoduleItem ID="IDS_2I" SubslotNumber="32768" SubmoduleIdentNumber="0x00000002"
IsochroneModeSupported="true" IsochroneModeInRT_Classes="RT_CLASS_3"
SupportedRT_Classes="RT_CLASS_1;RT_CLASS_3" TextId="TOK_DAP_InterfaceModule"
SupportedProtocols="SNMP;LLDP" NetworkComponentDiagnosisSupported="true"
PTP_BoundarySupported="true" DCP_BoundarySupported="true" MulticastBoundarySupported="true"
DCP_HelloSupported="true"
```

Media redundancy

18

Media Redundancy specifies a uniform method to increase the availability of the network. It is characterized by the presence of additional resources which are not required for a failure-free operation of the network. In case of a failure, these additional resources are used to prevent disintegration of the network. The media redundancy protocol consists of a **M**edia **R**edundancy **M**anager (MRM) and a **M**edia **R**edundancy **C**lient (MRC). Usually, the PROFINET controller performs the functionality of an MRM. Therefore, the following example focuses on an MRC definition within the InterfaceSubmodule of a PROFINET device.

MRPD

MRPD is an abbreviation of **M**edia **R**edundancy with **P**lanned **D**uplication of frames which specifies the use of redundancy for IRT communication. **MRPD_Supported** attribute must be set to "true" if the device supports MRPD with bumpless cyclic RT Class 3 data. MRPD always requires the activation of MRP in the ring.

Note: StartupMode "Advanced" must be supported by an interface submodule to support MRPD.

```
<InterfaceSubmoduleItem ID="IDS_8I" SubslotNumber="32768">
    <RT_Class3Properties ForwardingMode="Absolute;Relative">
    <SynchronisationMode SupportedRole="SyncSlave" MaxLocalJitter="300">
    <ApplicationRelations NumberOfAR="4" StartupMode="Legacy;Advanced">
    </ApplicationRelations>
    <MediaRedundancy SupportedRole="Client" MRPD_Supported="true"␣
↪AdditionalForwardingRulesSupported="true"/>
</InterfaceSubmoduleItem>
```

IRT performance upgrade

19

IRT performance upgrade is a set of improvements to IRT to be the best in its class within all Ethernet field busses. Those improvements offer the following functionalities, which are usable in the wide range of PROFINET:

- Send clocks less than 250 µs
- Fast Forwarding
- Fragmentation
- Preamble shortening
- Dynamic frame packing

The examples below allow supporting all functionalities mentioned above. Please keep in mind that these functionalities can also be enabled selectively. In addition to this, the definition of application class as High performance[19] is optional. The features mentioned above can be used and activated without the need of defining this application class.

```
<CertificationInfo ConformanceClass="C" ApplicationClass="Isochronous;HighPerformance"
↪NetloadClass="III"/>
```

RT_Class3Properties must include the following attributes[20]: ForwardingMode, MaxBridgeDelayFFW, MaxDFP_Frames, MaxDFP_Feed, AlignDFP_Subframes, FragmentationType, MaxRedPeriodLength, MinFSO, MinRTC3_Gap, MinYellowTime, YellowSafetyMargin, DFP_OutboundTruncationSupported and MaxRetentionTime.

```
<RT_Class3Properties ForwardingMode="Absolute;Relative" MaxBridgeDelay="2088"
MaxBridgeDelayFFW="1024" MaxRangeIR_FrameID="1024" MaxNumberIR_FrameData="1024"
StartupMode="Legacy;Advanced" MaxDFP_Frames="1" MaxDFP_Feed="340" AlignDFP_Subframes="false"
FragmentationType="Dynamic" MaxRedPeriodLength="4000" MinFSO="1760" MinRTC3_Gap="1120"
MinYellowTime="10240" YellowSafetyMargin="160" DFP_OutboundTruncationSupported="false"
MaxRetentionTime="30000"/>
```

InterfaceSubmoduleItem must include the following items: MaxFrameStartTime, MinNRT_Gap.

```
<InterfaceSubmoduleItem ID="IDS_2I" SubslotNumber="32768" SubmoduleIdentNumber="0x0002"
IsochroneModeSupported="true" IsochroneModeInRT_Classes="RT_CLASS_3"
SupportedRT_Classes="RT_CLASS_1;RT_CLASS_3" TextId="TOK_DAP_InterfaceModule"
SupportedProtocols="SNMP;LLDP" NetworkComponentDiagnosisSupported="true"
PTP_BoundarySupported="true" DCP_BoundarySupported="true"
MulticastBoundarySupported="true" DCP_HelloSupported="true"
MaxFrameStartTime="1600" MinNRT_Gap="960"
DelayMeasurementSupported="true" Writeable_IM_Records="1 2 3 4" IM5_Supported="true">
```

SynchronizationMode must support the following item: PeerToPeerJitter.

[19] According to IEC 61784-2.

[20] To check whether any of those attributes are mandatory/optional, please refer to chapter 8.13.5 of *[1]*

```
<SynchronisationMode SupportedRole="SyncSlave" MaxLocalJitter="300" T_PLL_MAX="1000"␣
 ↪SupportedSyncProtocols="PTCP"
PeerToPeerJitter="250"/>
```

PortSubmoduleItem for each port must support the following item: ShortPreamble100MBitSupported.

```
<PortSubmoduleItem ID="IDS_2P1" SubslotNumber="32769" SubmoduleIdentNumber="0x00000003"
TextId="TOK_Port1" MaxPortRxDelay="198" MaxPortTxDelay="6" IsDefaultRingport="true"
MAUType="100BASETXFD" MAUTypes="16" PortDeactivationSupported="true"
LinkStateDiagnosisCapability="Up+Down"
ShortPreamble100MBitSupported="true"
CheckMAUTypeSupported="true" CheckMAUTypeDifferenceSupported="true" Writeable_IM_Records="1 s2 3␣
 ↪4" IM5_Supported="true">
```

Send clocks less than 250 µs can be added to DeviceAccessPointItem and RT_Class3TimingProperties as values of "MinDeviceInterval" attribute and "SendClock" attribute[21] respectively. For a send clock less than 250 µs, those values are to be less than 8.

```
<DeviceAccessPointItem ID="DAP 2" PhysicalSlots="0..16" ModuleIdentNumber="0x00000002"
   MinDeviceInterval="4"
   ImplementationType="ERTEC200P" DNS_CompatibleName="ERTEC-DEVKit"
   FixedInSlots="0" ObjectUUID_LocalIndex="1" MultipleWriteSupported="true"
   SharedDeviceSupported="true" DeviceAccessSupported="true" NumberOfDeviceAccessAR="1"
   MaxSupportedRecordSize="8192" NameOfStationNotTransferable="true"
   ParameterizationSpeedupSupported="true" LLDP_NoD_Supported="true" ResetToFactoryModes="2"
   CheckDeviceID_Allowed="true" PowerOnToCommReady="490" IOXS_Required="false"
   RequiredSchemaVersion="V2.43" PNIO_Version="V2.43" AddressAssignment="LOCAL;DCP">
   <RT_Class3TimingProperties ReductionRatioPow2="2 4 8 16" SendClock="4 6 8 12 16 20 24 28 32␣
 ↪36 40 44 48 52 56 60 64 68 72 76 80 84 88 92 96 100 104 108 112 116 120 124 128"␣
 ↪ReductionRatio="1"/>
```

[21] The value of MinDeviceInterval contains the multiplier of the basic clock tick (i.e., 31.25 µs). For example, If MinDeviceInterval is 8, then 31.25x8 = 250 µs which implies that the device is capable of sending/receiving data every 250 µs. This claim is also valid for the "SendClock" attribute of RT_Class3TimingProperties.

IRT without Isochronous mode

20

Isochronous Mode means that the acquisition of peripheral inputs, their processing within an application program on the controller and the activation of peripheral outputs have a defined time relationship to each other. E.g., for motion control it is important to know how old the position values delivered by the inputs are and how long it will take to activate the new setpoints delivered by the outputs.

Isochronous Mode provides the basis for an isochronous application to be able to get **time-synchronized access** to the data of a group of submodules spread over multiple PROFINET devices.

Isochronous Mode locks the application's execution loop to the IRT update interval. Isochronous applications are possible with IRT.

IRT supporting devices without Isochronous Mode can:

- handle and pass on the IRT protocols
- be integrated in an IRT topology
- eliminate the variable data delays (jitter) at heavy network traffic.

PROFINET Security Class 1

21

Within the scope of PROFINET Security Class 1 feature, the following functionalities are supported by the PROFINET device with the introduction of the new attributes in the GSD file:

- Activate / deactivate the SNMP service by engineering.
- Enable read-only access for SNMP.
- Change the SNMP community names by engineering.
- Activate / deactivate DCP read-only mode by engineering.

To support SNMP-related features stated above, InterfaceSubmoduleItem must include the attribute SNMP_FeaturesSupported with the value SNMPAdjust.

To activate/deactivate DCP read-only mode, InterfaceSubmoduleItem must include the attribute “DCP_FeaturesSupported” with the value “RejectDCPSet”.

```
<InterfaceSubmoduleItem ID="IDS_2I" SubslotNumber="32768" SubmoduleIdentNumber="0x0002"
DCP_FeaturesSupported="RejectDCPSet" SNMP_FeaturesSupported="SNMPAdjust"
     IsochroneModeSupported="true" IsochroneModeInRT_Classes="RT_CLASS_3"
SupportedRT_Classes="RT_CLASS_1;RT_CLASS_3" TextId="TOK_DAP_InterfaceModule"
SupportedProtocols="SNMP;LLDP" NetworkComponentDiagnosisSupported="true"
PTP_BoundarySupported="true" DCP_BoundarySupported="true" MulticastBoundarySupported="true">
```

In addition to these functionalities explained above, GSD files and their referenced resources must be packaged in a GSDX file and protected against manipulation with a certificate-based signature issued by PROFIBUS & PROFINET International (PI). For details, please refer to Chapter 26 and Chapter 27.

To demonstrate the device’s compatibility with PROFINET Security Class 1 feature, specify the new attribute “SecurityClass” with a value of “1” within the CertificationInfo element of the GSD file, as illustrated below.

```
<CertificationInfo ConformanceClass="C" ApplicationClass="Isochronous;HighPerformance"
NetloadClass="III" SecurityClass="1"/>
```

PROFIsafe definitions in the GSD file

22

I/O modules that support PROFIsafe functionality must be defined in the GSD file within the "VirtualSubmoduleList" element. The following example shows what the definition shall look like:

```
<VirtualSubmoduleList>
    <VirtualSubmoduleItem ID="IDS_Safe261" SubmoduleIdentNumber="0x00000023"
    PROFIsafeSupported="true"
    MayIssueProcessAlarm="false">
            <IOData F_IO_StructureDescCRC="65432">
                <Input Consistency="All items consistency">
                    <DataItem DataType="Unsigned8" UseAsBits="true" TextId="IDT_Input_DataItem_
↪PS_DI0"/>
                    <DataItem DataType="Unsigned8" UseAsBits="true" TextId="IDT_Input_DataItem_
↪PS_DI1"/>
                    <DataItem DataType="Integer16" TextId="IDT_Input_DataItem_PS_AI"/>
                    <DataItem DataType="F_MessageTrailer5Byte" TextId="IDT_Input_DataItem_PS_
↪Safety"/>
                </Input>
                <Output Consistency="All items consistency">
                    <DataItem DataType="F_MessageTrailer5Byte" TextId="IDT_Output_DataItem_PS_
↪Safety"/>
                </Output>
            </IOData>
            <RecordDataList>
                <F_ParameterRecordDataItem Index="1" F_ParamDescCRC="56313">
                    <F_Check_iPar/>
                    <F_SIL DefaultValue="SIL3" AllowedValues="SIL1 SIL2 SIL3"/>
                    <F_CRC_Length DefaultValue="4-Byte-CRC" AllowedValues="4-Byte-CRC" Visible=
↪"true"/>
                    <F_CRC_Seed/>
                    <F_Passivation DefaultValue="Device/Module" AllowedValues="Device/Module"/>
                    <F_Block_ID DefaultValue="0" AllowedValues="0" Changeable="false"/>
                    <F_Par_Version/>
                    <F_Source_Add/>
                    <F_Dest_Add/>
                    <F_WD_Time DefaultValue="500" AllowedValues="10..2000"/>
                    <F_Par_CRC DefaultValue="21211"/>
                </F_ParameterRecordDataItem>
            </RecordDataList>
            <ModuleInfo>
                <Name TextId="IDT_NAME_Safe261"/>
                <InfoText TextId="IDT_INFO_Safe261"/>
            </ModuleInfo>
        </VirtualSubmoduleItem>
    </VirtualSubmoduleList>
```

PROFIenergy definitions in the GSD file 23

The PROFIenergy profile enables control devices (e.g. PLCs) to send commands to Energy Consuming Units (ECU), to signal pauses such as lunch breaks, holidays, random line stoppages or even peak load conditions. On receipt of the PROFIenergy commands, software agents in the ECU firmware initiate predefined energy-saving modes for the duration of the pause.[22]

I/O modules that support PROFIenergy functionality must be defined in the GSD file within the "VirtualSubmoduleList" element. The following example shows what the definition shall look like:

```
<VirtualSubmoduleList>
   <VirtualSubmoduleItem ID="1" SubmoduleIdentNumber="0x00000001" Writeable_IM_Records="1 2 3 4"␣
↪MayIssueProcessAlarm="true">
            <IOData>
               <Input Consistency="All items consistency">
                  <DataItem DataType="OctetString" TextId="TOK_Input_DataItem_1" Length="1" ␣
↪UseAsBits="true"/>
               </Input>
            <Output Consistency="All items consistency">
               <DataItem DataType="OctetString" TextId="TOK_Output_DataItem_1" Length="1"␣
↪UseAsBits="true"/>
            </Output>
            </IOData>
            <RecordDataList>
               <ParameterRecordDataItem Index="1" Length="4" TransferSequence="0">
                  <Name TextId="T_general_parameter"/>
                  <Const Data="0x20,0x01,0x56,0x78" ByteOffset="0"/>
               </ParameterRecordDataItem>
            </RecordDataList>
            <ModuleInfo>
               <Name TextId="TOK_TextId_Module_1IO"/>
               <InfoText TextId="TOK_InfoTextId_Module_1IO"/>
            </ModuleInfo>
            <PROFIenergy
            ProfileVersion="V1.3" EntityClass="Class3" EntitySubclass="Subclass1" >
               <EnergySavingModeList>
                  <EnergySavingModeItem ID="1" TimeToPause="600" RTTO="600" TimeMinLengthOfStay=
↪"200"/>
                  <EnergySavingModeItem ID="2" TimeToPause="600" RTTO="600" TimeMinLengthOfStay=
↪"200"/>
                  <EnergySavingModeItem ID="255" TimeToPause="600" RTTO="600"␣
↪TimeMinLengthOfStay="200"/>
               </EnergySavingModeList>
               <MeasurementList>
                  <MeasurementItem Number="0">
                     <MeasurementValue ID="13" AccuracyDomain="1" AccuracyClass="3"/>
                  </MeasurementItem>
                  <MeasurementItem Number="1">
                     <MeasurementValue ID="14" AccuracyDomain="1" AccuracyClass="3"/>
                  </MeasurementItem>
```

(continues on next page)

[22] This information is taken from and can be found in *[7]*

(continued from previous page)

```
                <MeasurementItem Number="2">
                    <MeasurementValue ID="15" AccuracyDomain="1" AccuracyClass="3"/>
                </MeasurementItem>
            </MeasurementList>
        </PROFIenergy>
    </VirtualSubmoduleItem>
</VirtualSubmoduleList>
```

Compatibility of GSDML versions

24

Further developments of the PROFINET technology usually influence the structure and/or content of the GSDs. This is reflected in an updated schema version of the GSDML. When developing the GSDML, it is ensured that existing GSD files are compatible with the more recent schema version (upward compatibility).

Downward compatibility, however (when a "more recent" GSD encounters an "older" GSDML schema) requires a more detailed examination.

If the structures used in the GSD have not yet been defined in the present schema, a schema validation will indicate error(s). In order to fix the error(s), always the latest engineering tools that "know" the new schema version would have to be used, or the field device manufacturer would have to develop and update several GSD versions. However, neither solution is desirable.

An alternative would be to dispense with the schema validation in the engineering tools, and to ignore all unknown GSD structures in the validation. But this would have the disadvantage that the correct function of the field device cannot be ensured due to the possibility that it might be necessary for the field device to receive the data that is newly described in the GSD.

As a remedy, the attribute "RequiredSchemaVersion" is added to the GSDML Version >= 2.0. By means of this attribute, the field device manufacturer can control which GSD structures can be ignored by an engineering tool without leading to errors. This attribute can be specified on the level of a single module or DAP.

The example below shows the entry in a GSD of schema version 2:

```
<ModuleItem ID="IDM_1" ModuleIdentNumber="0x00000000" RequiredSchemaVersion="V1.0">
...
    <VirtualSubmoduleList>
        <VirtualSubmoduleItem ID="IDS_2" SubmoduleIdentNumber="0x00000010">
            <IOData>
                ...
                <Output>
                    <DataItem DataType="Unsigned8" UseAsBits="true" TextId="IDT_DATAITEM_
↪DigitalOutputs">
                        <BitDataItem BitOffset="0" TextId="IDT_DATAITEM_CHANNEL0"/>
                        <BitDataItem BitOffset="1" TextId="IDT_DATAITEM_CHANNEL1"/>
                        <BitDataItem BitOffset="2" TextId="IDT_DATAITEM_CHANNEL2"/>
                        <BitDataItem BitOffset="3" TextId="IDT_DATAITEM_CHANNEL3"/>
                    </DataItem>
                </Output>
                ...
            </IOData>
        ...
    ...
...
</ModuleItem>
```

Here, it is specified for the module that it can be operated also with Version 1.0 of the GSDML. In the example, the expansion in comparison to Version 1.0 consists of the description of the bit structure of the

output data (element "BitDataItem"). Since this description has no effect on the function of the module itself, these elements can be ignored. An engineering tool that can only handle a GSDML Version 1.0 will read the data as follows:

```
<ModuleItem ID="IDM_1" ModuleIdentNumber="0x00000000">
    <VirtualSubmoduleList>
            <VirtualSubmoduleItem ID="IDS_2" SubmoduleIdentNumber="0x00000010">
                <IOData>
                        ...
                        <Output>
            <DataItem DataType="Unsigned8" UseAsBits="true"
            TextId="IDT_DATAITEM_DigitalOutputs">
                        </DataItem>
                        </Output>
                        ...
                    </IOData>
                    ...
                ...
        ...
</ModuleItem>
```

However, in other cases, it might be necessary for the field device to receive the parameters described in the GSD. An example is a field device that can be used only in a synchronous environment. Since in the GSDML V1.0 no description resources are available for this, it does not make sense to configure this field device with a tool that is only able to interpret a Version 1.0 of the GSDML.

In the engineering tool, modules or DAPs are not offered for configuration that have a "RequiredSchemaVersion" which is not supported.

Tools

25

25.1 PROFINET GSD Checker

This tool[23] is offered free of charge by the PNO and main functions are as follows:

- Displaying the contents of a GSD file
- Validation of a GSD file against the schema files.
- Importing and validating signatures of GSDX packages (starting from v2.44.1)

Moreover, the tool makes the GSD documentation available and allows the integration of any XML editor for generating GSD files. Since a few sample files are installed at the same time, the tool also provides a good basis for generating new GSD files.

Figure 25.1 shows the main screen of the tool.

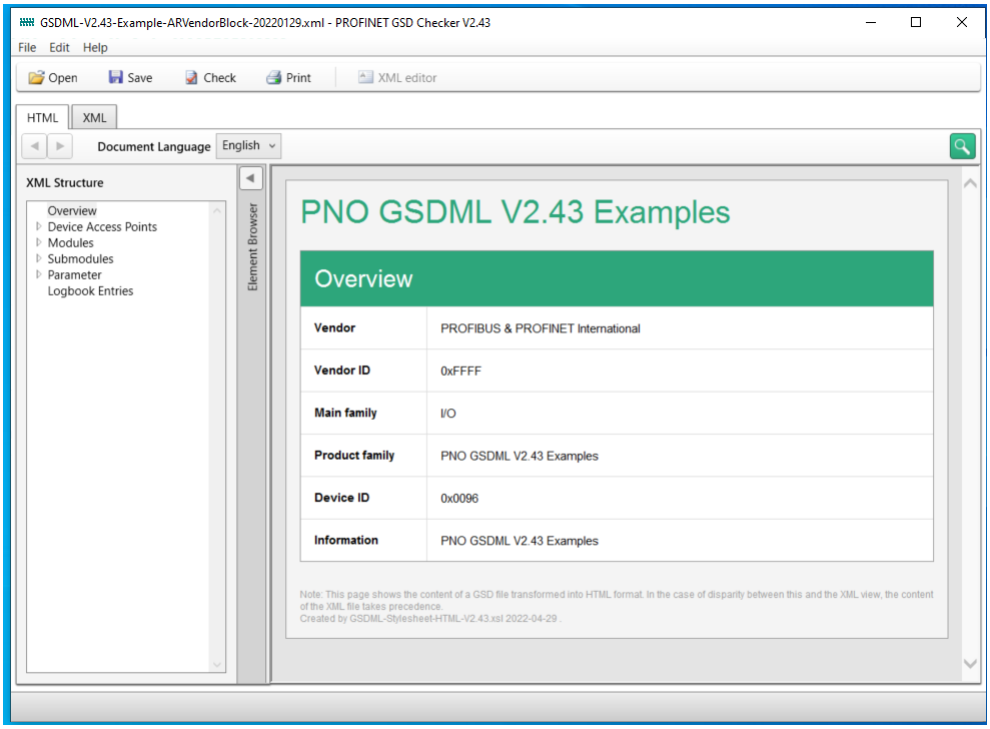

Fig. 25.1: **PROFINET GSD Checker**

[23] This tool is produced by the collaboration of a 3rd party company and Siemens.

25.2 Tools for GSD generation

Since a GSD file is an XML file, it can be created with a simple text editor. However, no support is provided when the input is made. Therefore, it might easily be the case that the generated file is not well-formed, or not valid according to a schema.

When using XML, numerous tools can be employed to create a GSD file. Two groups must be essentially differentiated:

- Validating XML parsers check whether the present XML document is structured according to the rules of a schema.
- XML editors provide support when editing an XML document and as a rule possess an integrated XML parser, or an interface for an external XML parser.

The XSD files for GSDML use many features of the XML schema language. An XML parser must be able to interpret the constructs that are used.

There is a huge number of editors for generating XML documents. For the creation of the GSDML schema files themselves the tool "XMLSpy" has been used. It can also be used for editing GSD files. XMLSpy has a "built-in" XML parser that can be used for validating GSD files.

The XMLSpy can be obtained from the following webpage:

- https://www.altova.com/xmlspy-xml-editor

Other XML editors:

- XML Notepad
- XMLFox
- XML Copy Editor
- Oxygen XML Editor
- Easy XML Editor

Additionally, there are extensions and/or tools developed for commonly used development environments, such as Notepad++, Eclipse, etc. For these, community pages or marketplaces can be referred to.

GSD File Signing Process

26

The GSDX Specification defines a container file format based on ISO/IEC 29500-2:2021 that allows to package any files and their referenced resources in a single package. These so-called GSDX files contain one or more GSD files, their attachments, and an optional digital signature. For more details regarding the GSDX Specification please refer to the document “GSDX Specification for PROFINET” (Order No.: 2.752) from the address (https://www.profibus.com/download/gsdml-gsdx-specification-for-profinet). A certificate-based digital signature is used to provide the integrity and authenticity of the GSDX files. By using the signed GSDX container format, any operator can verify whether the GSD files and resources are exactly the same as those delivered by the manufacturer. PROFIBUS & PROFINET International (PI) has established a Public Key Infrastructure (PKI). It allows to issue certificates to vendors which shall be used for signing. The PI web page provides a portal (https://www.profibus.com/products/gsd-files/gsd-signing-portal-1) with access to accounts for vendors to request a certificate.

PI offers two GSDX build options to their members:

- PI creates a manufacturer-specific certificate and delivers it to the manufacturer together with a license for the signing tool and a smart card per user. The manufacturer can sign the GSD files themselves using the provided tooling.
- PI creates a manufacturer-specific certificate, but the certificate remains at the PI Certification Office. The manufacturer can upload the GSD file to be signed via the GSD Signing Portal and will then receive the signed file in the GSDX container format.

The signing certificate needs to be renewed every three years for each of the two options described above. The vendor can request revocation if the credentials are compromised. Both activities must be initiated from the vendor via the web portal (https://www.profibus.com/products/gsd-files/gsd-signing-portal-1).

For more details, please refer to the document “PROFINET Security Class 1” (Order No.: 7.312) from (https://www.profibus.com/download/profinet-security-guideline).

Signature Validation of GSDX File

27

Engineering tools which support PN Security Class 1 features should be able to work with GSDX packages and validate the GSDX signatures as a preliminary step before importing the package. The primary function of GSDX validation is to verify that the content of the GSDX file originates from a reliable source and remains unaltered. In case errors are detected during this validation, they should be presented to the user with utmost clarity and detail. In the following sections you can find information about how to validate a GSDX package with TIA Portal or GSD Checker.

27.1 Validating signature of GSDX file with GSD Checker tool

- Start the GSD Checker tool which can be downloaded from (https://www.profibus.com/download/profinet-gsd-checker).
- Open the GSDX file in the same manner as a standard GSDML file (use the File – Open menu, click the Open button in the toolbar, or simply drag and drop the file into the tool).
- Following the loading of the GSDX file, the GSDX Explorer displays the result of the signature validation along with the associated success or error message.

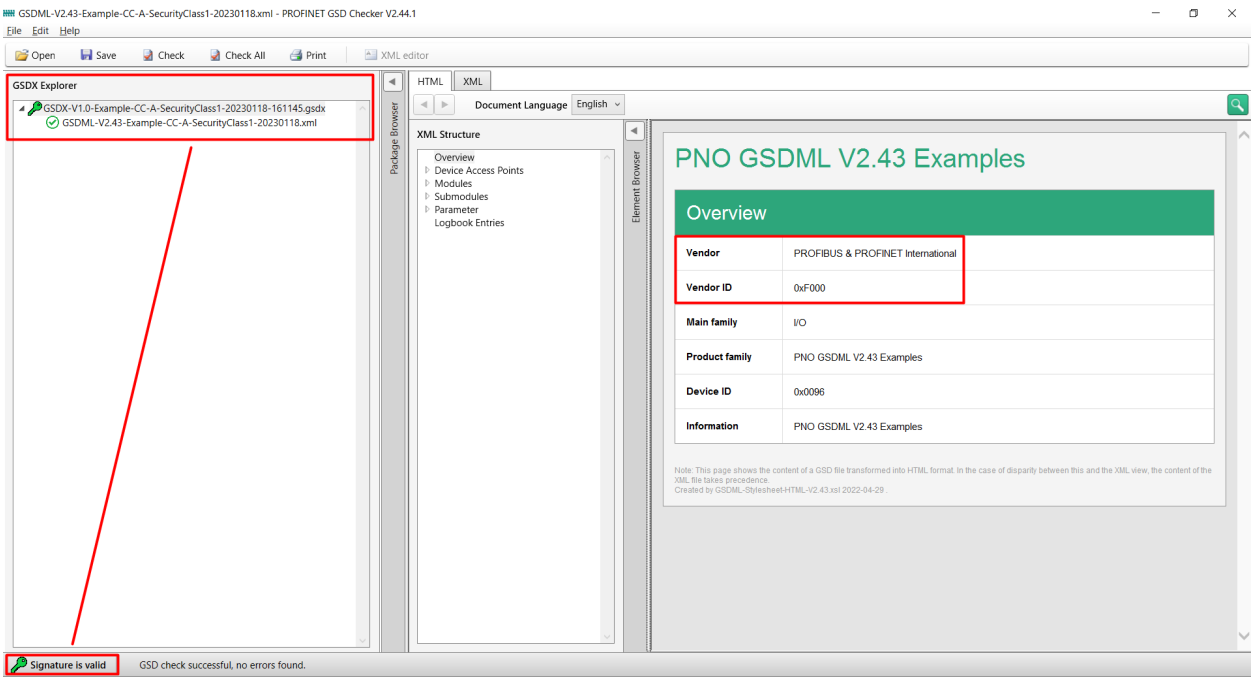

Fig. 27.1: **Result for the valid signature**

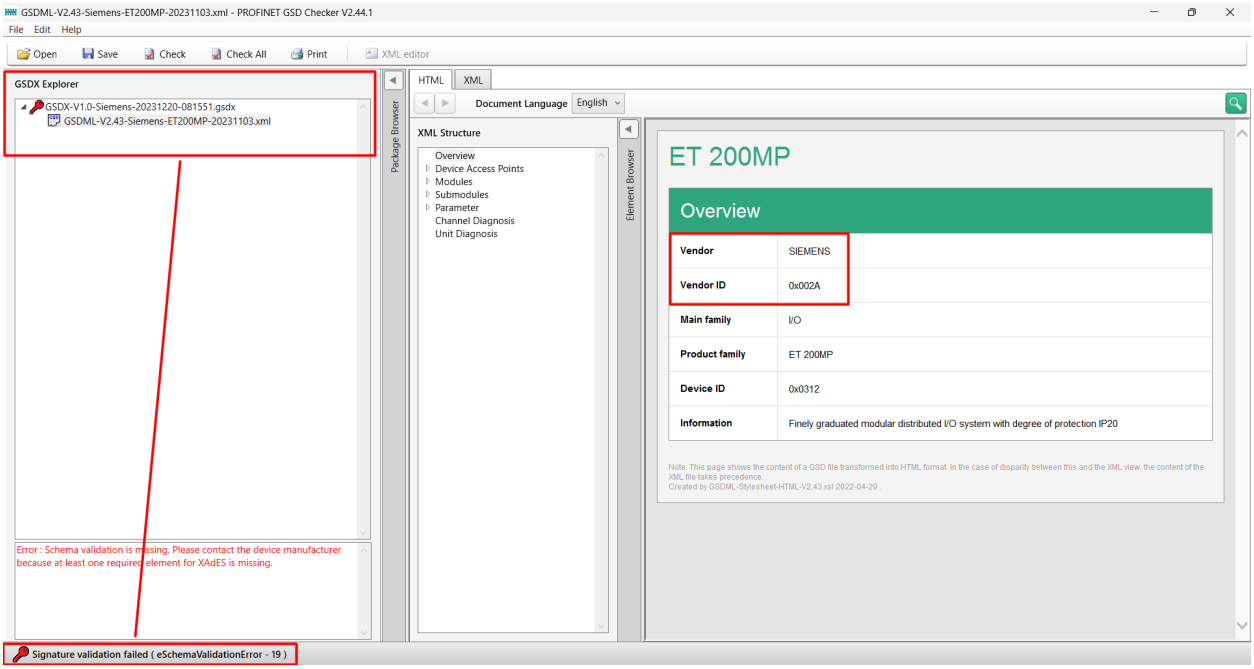

Fig. 27.2: **Result for the invalid signature**

27.2 Validating signature of GSDX file with TIA Portal

- Open “Options -> Manage general station description files (GSD)” to import a GSD file into TIA Portal.

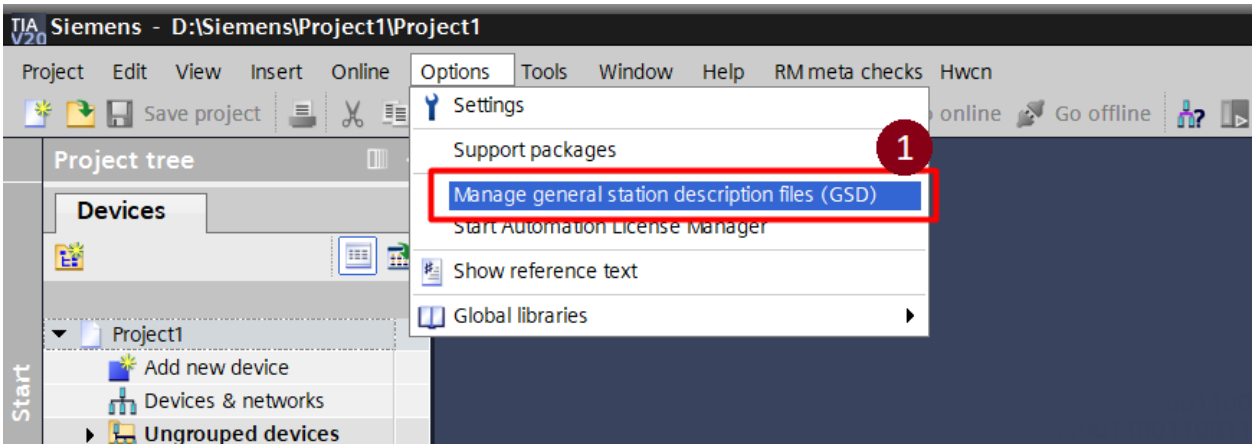

Fig. 27.3: **Manage general station description files (GSD)**

- Select the folder where the GSDX files are located. TIA Portal displays the list of all GSD files inside of each GSDX file.
- Select the desired GSD files and press the “Install” button. Then, GSDX file is validated.

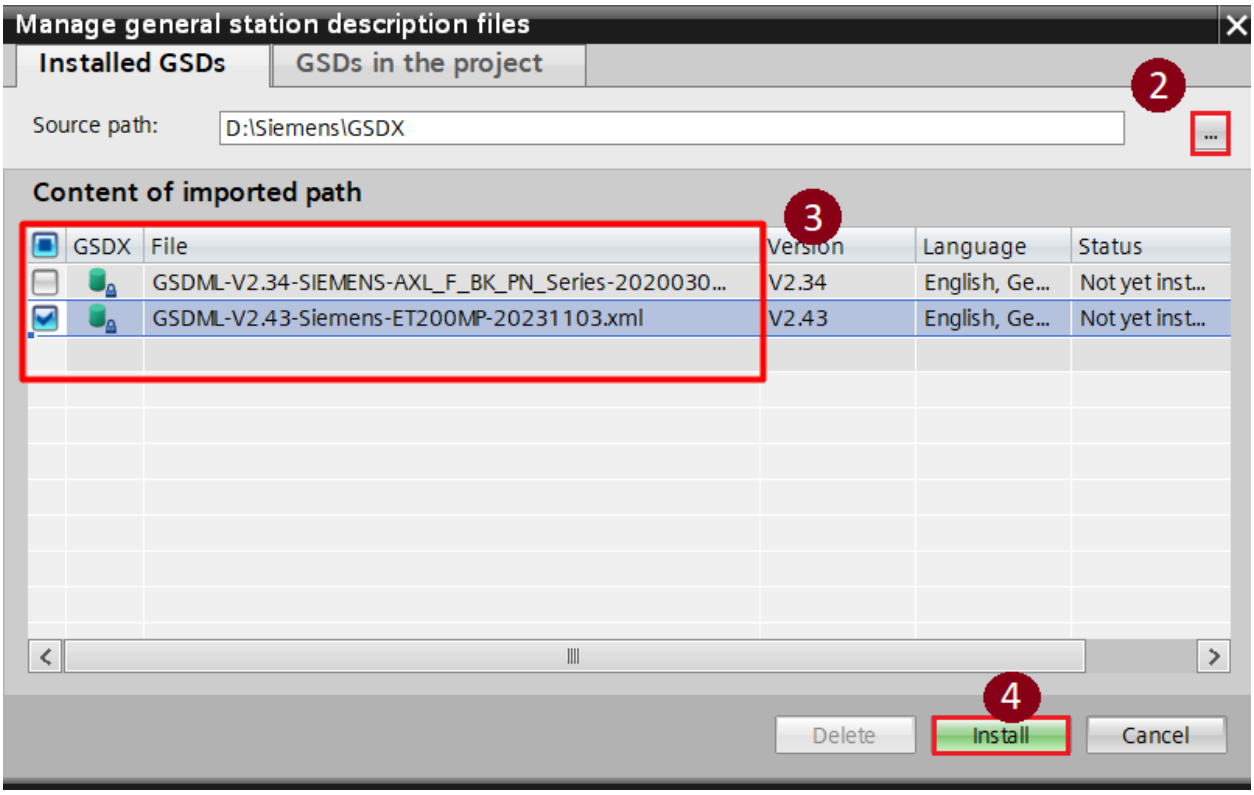

Fig. 27.4: **Install the GSDX file**

- If the selected file is valid, it is imported to the hardware catalogue. If the validation is not successful, an error message is displayed.

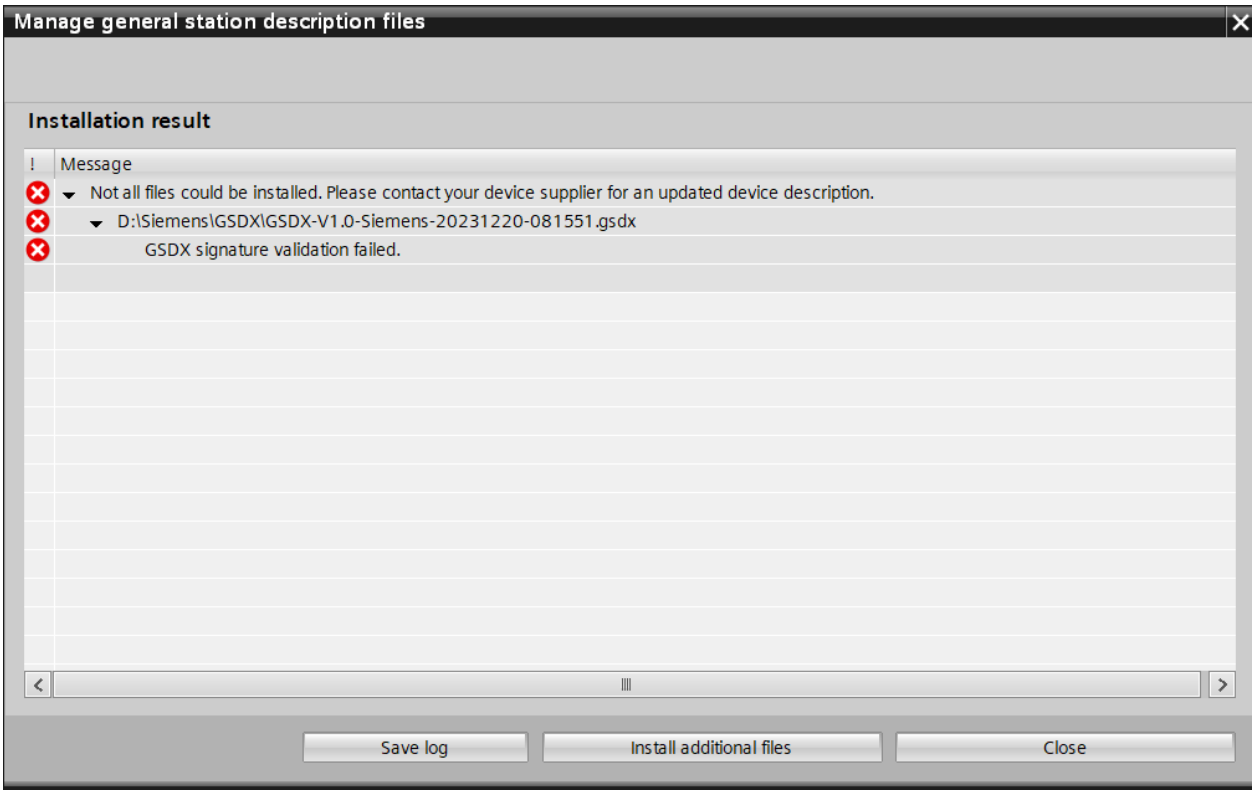

Fig. 27.5: **Error message after the validation**

Appendix 28

28.1 Abbreviations

Table 28.1: Abbreviations

Abbreviation	Meaning
DAP	Device Access Point
DCP	Discovery and Basic Configuration Protocol
FSU	Fast Startup Utility
GSD	General Station Description
GSDML	GSD Markup Language
GSD WG	GSD Working Group
GSDX	GSD Container based on ISO/IEC 29500-2:2021
ID	Identification
IEC	International Electrotechnical Commission (http://www.iec.ch/)
IRT	Isochronous Real-time
ISO	International Organization for Standardization (http://www.iso.org/)
LLDP	Link Layer Discovery Protocol
MIB	Management Information Base
MRC	Media Redundancy Client
MRM	Media Redundancy Manager
MRP	Media Redundancy Protocol
MRPD	Media Redundancy with Planned Duplication of frames
PDEV	Physical Device
PKI	Public Key Infrastructure
PNO	PROFIBUS Nutzerorganisation e.V. (http://www.profibus.com/)
PTCP	Precision Transparent Clock Protocol
RT	Real-time
SNMP	Simple Network Management Protocol

continues on next page

Table 28.1 – continued from previous page

Abbreviation	Meaning
USI	User Structure Identifier
W3C	W3C World Wide Web Consortium (http://www.w3.org/)
XML	Extensible Markup Language
XSD	XML Schema Definition recommended by W3C
XSL	Extensible Stylesheet Language Family