

ICS 47.020.70

Maritime navigation and radiocommunication equipment and systems – Digital interfaces – Part 420: Multiple talkers and multiple listeners – Ship systems interconnection – Companion standard requirements and basic companion standards (IEC 61162-420:2001); German version EN 61162-420:2002, text in English

Matériels et systèmes de navigation et de radiocommunications maritimes – Interfaces numériques – Partie 420: Émetteurs multiples et récepteurs multiples – Interconnexion des systèmes embarqués – Exigences standard de la couche utilisateur et standard utilisateurs de base (CEI 61162-420:2001); Version allemande EN 61162-420:2002, texte en anglais

Die Europäische Norm EN 61162-420:2002 hat den Status einer Deutschen Norm.

Beginn der Gültigkeit

Die EN 61162-420 wurde am 2002-02-01 angenommen.

Nationales Vorwort

Für die vorliegende Norm ist das nationale Arbeitsgremium K 738 „Elektronische Navigationsinstrumente“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE zuständig.

Das Präsidium des DIN hat mit Präsidialbeschluss 1/78 festgelegt, dass Deutsche Normen, deren Inhalt sich auf internationale Arbeitsergebnisse der IEC gründet, unter bestimmten Bedingungen allein in englischer Sprache veröffentlicht werden dürfen. Die Bedingungen sind für diese Norm erfüllt. Sie steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen. Es kann davon ausgegangen werden, dass die Anwender über hinreichende englische Sprachkenntnisse verfügen. Für viele der verwendeten Benennungen existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender im Bereich der Seeschifffahrt in der Regel ebenfalls der englischen Benennungen bedienen.

Norm-Inhalt für die Teile 400, 401, 410 und 420 der Reihe DIN EN 61162-4 war veröffentlicht als E DIN IEC 80/175/CD:1998-10 und E DIN IEC 80/176/CD:1998-10.

Die enthaltene IEC-Publikation wurde vom TC 80 „Maritime navigation and radiocommunication equipment and systems“ erarbeitet.

Die Reihe DIN EN 61162 besteht aus 4 Teilen, wobei der Teil 4 in mehrere Einzelnormen unterteilt ist. Eine zusammenfassende Beschreibung der Normen dieser Reihe und eine Übersicht über deren unterschiedliche Anwendbarkeit ist in Anhang B der Norm DIN EN 61162-400 enthalten.

Das IEC-Komitee hat entschieden, dass der Inhalt dieser Publikation bis Juni 2005 unverändert bleiben soll. Zu diesem Zeitpunkt wird entsprechend der Entscheidung des Komitees die Publikation

- bestätigt,
- zurückgezogen,
- durch eine Folgeausgabe ersetzt oder
- geändert.

Fortsetzung Seite 2
und 103 Seiten EN

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE
Normenstelle Schiffs- und Meerestechnik (NSMT) im DIN Deutsches Institut für Normung e.V.

Nationaler Anhang NA (informativ)

Zusammenhang mit Europäischen und Internationalen Normen

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf eine Norm ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils neueste gültige Ausgabe der in Bezug genommenen Norm.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe der Norm.

Der Zusammenhang der zitierten Normen mit den entsprechenden Deutschen Normen ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation. Beispiel: IEC 60068 ist als EN 60068 als Europäische Norm durch CENELEC übernommen und als DIN EN 60068 ins Deutsche Normenwerk aufgenommen.

IEC hat 1997 die Benummerung der IEC-Publikationen geändert. Zu den bisher verwendeten Normnummern wird jeweils 60000 addiert. So ist zum Beispiel aus IEC 68 nun IEC 60068 geworden.

Navigations- und Funkkommunikationsgeräte und -systeme für die
Seeschifffahrt – Digitale Schnittstellen – Teil 420: Mehrere Datensender und
mehrere Datenempfänger – Schiffssystemzusammenschaltung
Anforderungen an Begleitnormen und Basisbegleitnormen
(IEC 61162-420:2001)

Maritime navigation and radiocommunication
equipment and systems – Digital interfaces –
Part 420: Multiple talkers and multiple listeners –
Ship systems interconnection – Companion
standard requirements and basic companion
standards
(IEC 61162-420:2001)

Matériels et systèmes de navigation et de
radiocommunications maritimes – Interfaces
numériques – Partie 420: Émetteurs multiples
et récepteurs multiples – Interconnexion des
systèmes embarqués – Exigences standard de
la couche utilisateur et standard utilisateurs de
base
(CEI 61162-420:2001)

Diese Europäische Norm wurde von CENELEC am 2002-02-01 angenommen. Die CENELEC-Mitglieder sind gehalten, die CEN/CENELEC-Geschäftsordnung zu erfüllen, in der die Bedingungen festgelegt sind, unter denen dieser Europäischen Norm ohne jede Änderung der Status einer nationalen Norm zu geben ist.

Auf dem letzten Stand befindliche Listen dieser nationalen Normen mit ihren bibliographischen Angaben sind beim Zentralsekretariat oder bei jedem CENELEC-Mitglied auf Anfrage erhältlich.

Diese Europäische Norm besteht in drei offiziellen Fassungen (Deutsch, Englisch, Französisch). Eine Fassung in einer anderen Sprache, die von einem CENELEC-Mitglied in eigener Verantwortung durch Übersetzung in seine Landessprache gemacht und dem Zentralsekretariat mitgeteilt worden ist, hat den gleichen Status wie die offiziellen Fassungen.

CENELEC-Mitglieder sind die nationalen elektrotechnischen Komitees von Belgien, Dänemark, Deutschland, Finnland, Frankreich, Griechenland, Irland, Island, Italien, Luxemburg, Malta, Niederlande, Norwegen, Österreich, Portugal, Schweden, Schweiz, Spanien, der Tschechischen Republik und dem Vereinigten Königreich.

CENELEC

Europäisches Komitee für Elektrotechnische Normung
European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique

Zentralsekretariat: rue de Stassart 35, B-1050 Brüssel

Vorwort

Der Text des Schriftstücks 80/312/FDIS, zukünftige 1. Ausgabe von IEC 61162-420, ausgearbeitet von dem IEC TC 80 „Maritime navigation and radiocommunication equipment and systems“, wurde der IEC-CENELEC Parallelen Abstimmung unterworfen und von CENELEC am 2002-02-01 als EN 61162-420 angenommen.

Nachstehende Daten wurden festgelegt:

- spätestes Datum, zu dem die EN auf nationaler Ebene durch Veröffentlichung einer identischen nationalen Norm oder durch Anerkennung übernommen werden muss (dop): 2002-11-01
- spätestes Datum, zu dem nationale Normen, die der EN entgegenstehen, zurückgezogen werden müssen (dow): 2005-02-01

Anhänge, die als „normativ“ bezeichnet sind, gehören zum Norminhalt.
Anhänge, die als „informativ“ bezeichnet sind, enthalten nur Informationen.
In dieser Norm sind die Anhänge A, B, C, D, E, F und ZA normativ.
Der Anhang ZA wurde von CENELEC hinzugefügt.

Anerkennungsnotiz

Der Text der Internationalen Norm IEC 61162-420:2001 wurde von CENELEC ohne irgendeine Abänderung als Europäische Norm angenommen.

CONTENTS

INTRODUCTION	7
1 Scope and object	8
2 Normative references	9
3 Definitions	9
3.1 Terms and abbreviations	9
3.2 General typographical rules in this standard	11
4 General principles for the PCS	11
4.1 General structure	11
4.1.1 Purpose	11
4.1.2 Components of the PCSDL	11
4.1.3 Object based principles in the PCSDL	12
4.1.4 Generic and manufacturer specific companion standards	13
4.1.5 Generic companion standards (PFS)	14
4.1.6 Manufacturer specific companion standards	14
4.1.7 Guidelines for using PCS	15
4.2 Products of the PCS	16
4.2.1 MAU name	16
4.2.2 Interface name	16
4.2.3 Interface class name	16
4.2.4 Data object name	17
4.2.5 Data object function	17
4.2.6 Data object structure	17
4.2.7 Data object information contents	17
4.2.8 Run-time library information items	17
4.2.9 PFS information items	18
4.3 The PISCES foundation specifications (PFS)	18
4.4 Generic interfaces in the PFS	18
5 The companion standard reference specification	18
5.1 Introduction	18
5.2 Basic concepts of the PCS	19
5.3 Conventions for companion standard specification files	20
5.3.1 General principles	20
5.3.2 Tokens	20
5.3.3 Name structure	22
5.3.4 Name scope	22
5.3.5 Configurable identifiers and literal	23
5.4 General structure of PCS specifications	23
5.4.1 General	23
5.4.2 The specification header	24
5.4.3 The specification body	25
5.5 Application specifications	25
5.5.1 Overview	25
5.5.2 General layout	25
5.5.3 Header	26
5.5.4 Body	27
5.6 Interface component specifications	29
5.6.1 Overview	29
5.6.2 General layout	30
5.6.3 Header specification	30

5.6.4	Body specification	31
5.7	Information specifications	32
5.7.1	Overview	32
5.7.2	General layout	32
5.7.3	Header	33
5.7.4	Body	33
5.8	Data types	34
5.8.1	Overview	34
5.8.2	General layout	34
5.8.3	Header	35
5.8.4	Body	35
6	PISCES foundation specification (PFS)	37
6.1	Introduction	37
6.2	Naming conventions	38
6.3	Application classes	38
6.3.1	Introduction	38
6.3.2	Application base class: PACApplication	39
6.3.3	LNA MAU application: PACLNA	39
6.3.4	Managed applications: PACFullApplication	39
6.3.5	IEC 61162-1 and IEC 61162-2 interface application: PACNMEARelay	39
6.3.6	Console application: PACConsole	39
6.3.7	General alarm and monitoring application: PACServerApp	39
7	Specification requirements for PCS compliant applications	40
7.1	Introduction and general documentation format	40
7.2	Function block	40
7.2.1	Function block graphical view	40
7.2.2	Physical effects	40
7.2.3	Input variables	40
7.2.4	Output variables	41
7.2.5	Events	41
7.2.6	Commands	41
7.2.7	Status	41
7.2.8	Parameters	41
7.2.9	Indication of accept or connect functionality	41
7.3	Functional description	41
7.4	Companion standard descriptions	42

Annex A (normative) Defined keywords.....	43
Annex B (normative) Basic IEC 61162-4 data types	45
Annex C (normative) General application companion standards	46
C.1 Introduction and general principles	46
C.2 Functionality overview	46
C.2.1 General data definitions.....	46
C.2.2 Version codes	46
C.2.3 Manufacturer and model identification.....	46
C.2.4 Interface and MCP information.....	46
C.2.5 Authentication	46
C.2.6 File overview	47
C.2.7 Data types General	47
C.2.8 Application PACSimpleApplication	51
C.2.9 Application PACFullApplication	51
C.2.10 Interface PCCVersionCodes	52
C.2.11 Interface PCCApplicationInfo	53
C.2.12 Data types UserAuth.....	54
C.2.13 Interface PCCUserAuth	55
Annex D (normative) LNA-MAU companion standard	58
D.1 General principles	58
D.2 Companion standards	59
D.2.1 Data types LnaMau.....	59
D.2.2 Interface PCCLNASTats.....	61
Annex E (normative) General alarm and monitoring companion standards.....	64
E.1 Introduction and general principles	64
E.2 Alarm and monitoring system identifiers	64
E.3 Functionality overview	64
E.3.1 Companion standard for tag based monitoring and alarm system.....	64
E.3.2 Client-server architecture	65
E.3.3 Tag number	65
E.3.4 Tag sets.....	65
E.3.5 Tag information.....	65
E.3.6 Tag attributes.....	65
E.3.7 Tag data.....	65
E.3.8 Alarms	66
E.4 Application classes	66
E.5 Companion standard structure	67
E.6 File structure.....	67
E.7 Standard tag names.....	68
E.7.1 General.....	68
E.7.2 Structure of P tag name class	69
E.7.3 General structural rules	69
E.7.4 Main process codes.....	70
E.7.5 Process sub-codes	70
E.7.6 General sub-groups.....	70
E.7.7 Automation related sub-group	71
E.7.8 Navigation sub-groups.....	71

E.7.9	Data type indication group	71
E.7.10	Use of engineering units	72
E.7.11	Sequence number	72
E.8	Structure of standard tags (S class)	73
E.9	Structure of yard tags (Y class)	73
E.10	Structure of internal tags (I class).....	73
E.11	New tag name classes	73
E.12	General quality indicators.....	73
E.13	Certification	73
E.14	Time stamp.....	73
E.15	Validity flag	74
E.16	Authentication	74
E.17	Companion standard specifications	74
E.17.1	DATA TYPES TagData	74
E.17.2	Application PACReadableServer.....	80
E.17.3	Application PACWritableServer.....	80
E.17.4	Application PACAlarmSystem	81
E.17.5	Interface PCCTagDatabase	82
E.17.6	Interface PCCTagText.....	84
E.17.7	Interface PCCTagStream	85
E.17.8	Interface PCCTagNetsearch	86
E.17.9	Interface PCCTagAttributes.....	87
E.17.10	Interface PCCTagSubscribe.....	88
E.17.11	Interface PCCTagWrite	89
E.17.12	Interface PCCTagAlarm	91
E.17.13	Interface PCCTagSet	92
E.17.14	Interface PCCTagAttributeWrite.....	93
Annex F (normative)	Navigational interfaces	95
F.1	IEC 61162-1 relay function	95
F.2	Interface PCCNMEAIn.....	95
F.2.1	READ NoOfPorts	95
F.2.2	FUNCTION GetPortDescription.....	95
F.2.3	FUNCTION NoOfSentences.....	95
F.2.4	FUNCTION GetListOfSentences.....	95
F.2.5	FUNCTION GetSentence	95
F.2.6	SUBSCRIBE Port_<nn>	96
F.2.7	SUBSCRIBE Port_<nn>_<fmt>	96
F.3	Interface PCCNMEAOut	96
F.3.1	READ NoOfPorts	96
F.3.2	FUNCTION GetPortDescription.....	96
F.3.3	NONACKED-WRITE Port_<nn>	96
F.4	The IEC 61162-1/2 related companion standard documents	97
F.4.1	The IEC 61162-1/2 data type description	97
F.4.2	Description of Interface PCCNMEAIn.....	98
F.4.3	Description of Interface PCCNMEAOut.....	101
F.4.4	Application Description	102
Anhang ZA (normativ)	Normative Verweisungen auf internationale Publikationen mit ihren entsprechenden europäischen Publikationen.....	103

INTRODUCTION

International Standard IEC 61162 is a four part standard which specifies four digital interfaces for applications in marine navigation, radiocommunication and system integration.

The four parts are:

- IEC 61162-1 Single talker and multiple listeners
- IEC 61162-2 Single talker and multiple listeners, high speed transmission
- IEC 61162-3 Multiple talkers and multiple listeners – Serial data instrument network
- IEC 61162-4 Multiple talkers and multiple listeners – Ship systems interconnection. Part 4 of this standard is sub-divided into a number of individual standards with part numbers in the 400 series.

IEC 61162-420 contains the specification of a description language for IEC 61162-4 series companion standards (user layer specifications), a framework for the organization of such companion standard descriptions and also the descriptions of basic components that can be used as a starting point to build IEC 61162-4 series components and networks.

Later standards in the companion standard series (IEC 61162-42x) are expected to address more concrete interface requirements for specific navigational equipment.

Relationship with the other parts of the IEC 61162 series of standards is defined in annex B to IEC 61162-400.

MARITIME NAVIGATION AND RADIOCOMMUNICATION EQUIPMENT AND SYSTEMS – DIGITAL INTERFACES –

Part 420: Multiple talkers and multiple listeners – Ship systems interconnection – Companion standard requirements and basic companion standards

1 Scope and object

International Standard IEC 61162-420 specifies the requirement for and basic components of the IEC 61162-4 series companion standards. These components are referred to as follows:

- a) **PCS** (PISCES companion standards) which contain the rules for creation of companion standards. The general principles underlying the PCS are described in clause 4.
- b) **PCSDL** (PCS description language). Part of the PCS is the definition of the syntax for the various types of companion standard documents that make them readable by computer tools. The PCSDL is described in clause 5.
- c) **function block** description. The function block description is a high level and graphical description of applications using the IEC 61162-4 series interface standard. The function block syntax is specified in clause 7.
- d) **PFS** (PISCES foundation specifications) which contain a framework for classification of applications adhering to the IEC 61162-4 standard. The PFS will also provide a minimum level of interoperability between different manufacturers' applications using this framework. The PFS is described in clause 6.

Clause 5 contains the complete reference to the PCS description language. Subclause 5.2 explains the basic concept of the PCS which is given by the distinction between four types of specifications: applications, interfaces, information and data types. General conventions with respect to the syntax of the PCS can be found in 5.3. All PCS documents are based on a similar structure. This approach is intended to make it easier to become familiar with the syntax and semantics of the PCS which is defined in 5.3.1. The four subclauses thereafter explain in detail the syntax and semantics of the four different types of specifications generated by the PCS.

Clause 6 describes the relationship between the different classes of IEC 61162-4 applications and gives an overview of their functionality. The annexes contain the detailed PCS definitions for the classes.

The objective of companion standards is to provide definitions of the information that is transferred within an integrated ship control system and of how these information items can be accessed or provided. Furthermore, the standard shall allow the definition of the actual network interfaces which the applications use to connect to the system. The description format is machine-readable, allowing an automatic compilation of the description into interface software.

A companion standard allows the reader to, at will, shift the focus between a technical specification and a definition of interfaces and information items. The development team can determine information attributes like unit, power, accuracy and the structure of the system architecture and create a common interpretation basis for data before the system implementation. The formalisms underlying the specification language will at the same time provide an unambiguous and precise description of the equipment interfaces which allow the use of computer tools to automatically generate interface program codes or to inspect and manipulate interfaces on-line, for example for debugging and monitoring purposes.

2 Normative references

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this part of IEC 61162. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of IEC 61162 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

IEC 61162-1:2000, *Maritime navigation and radiocommunication equipment and systems – Digital interfaces – Part 1: Single talker and multiple listeners*

IEC 61162-2:1998, *Maritime navigation and radiocommunication equipment and systems – Digital interfaces – Part 2: Single talker and multiple listeners, high speed transmission*

IEC 61162-3, *Maritime navigation and radiocommunication equipment and systems – Digital interfaces – Part 3: Multiple talkers and multiple listeners – Serial data instrument network*¹

IEC 61162-400, *Maritime navigation and radiocommunication equipment and systems – Digital interfaces – Part 400: Multiple talkers and multiple listeners – Ship systems interconnection – Introduction and general principles*

IEC 61162-401, *Maritime navigation and radiocommunication equipment and systems – Digital interfaces – Part 401: Multiple talkers and multiple listeners – Ship systems interconnection – Application profile*

ISO/IEC 8859-1:1998, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

3 Definitions

For the purpose of this part of IEC 61162, the following definitions apply.

3.1 Terms and abbreviations

3.1.1

abstract specification

PCS specifications that are part of the PFS (defined in this standard) are not intended for direct implementation and are termed “abstract”

3.1.2

application interface

a collection of interface components instantiated in an application definition document as one protocol level interface

¹ To be published.

3.1.3

CP – connection point

an application interface consists of a number of individual “functions” that can be called into or out from

3.1.4

CS – companion standard

a protocol layer on top of the normal OSI application level (see definition of companion standard in IEC 61162-400), representing the definition of how the application layer functionality is used to implement a certain application’s interface functionality. Also called user layer

3.1.5

function block

a high level, partly graphical representation of an application’s place in an integrated system which presents all interfaces and relationships between these on an overview level

3.1.6

interface (component)

a collection of connection points (CP) in one INTERFACE definition document. It can be aggregated with others into an actual interface as defined in the A-profile. The actual interface (A-profile sense) is defined as the application interface in the APPLICATION document

3.1.7

NMEA

the National Marine Electronics Association (NMEA) maintains a protocol standard called NMEA 0183 that is almost identical to IEC 61162-1 and IEC 61162-2. For historical reasons, symbolic names in some of the PCS documents in this standard refers to NMEA instead of to IEC 61162-1 (annex F). Although references to NMEA are made, the PCS documents define an application that is intended for use together with the IEC 61162-1 and IEC 61162-2 standards. It can also be useful for transmittal of NMEA 0183 messages, but this is outside the scope of this standard.

3.1.8

PCS – PISCES companion standard

the complete concept, including description language (PCSDL), function blocks and the foundation classes (PFS)

3.1.9

PCSDL – PISCES companion standard description language

the formal interface description language for PCS

3.1.10

PFS – PISCES foundation specifications

the interface base classes for all applications created in the framework of the PCS

3.1.11

tangible

a specification of an entity that shall be implemented (instantiated) at some time

3.2 General typographical rules in this standard

The following typographical rules apply throughout this standard:

- a) fragments and complete pieces of PCSDL source code is written in `Courier`;
- b) tokens written in capitals, typeset in `COURIER` are reserved PCSDL keywords;
- c) words in angle brackets (elbows), '<' and '>', define place-holders that have to be filled with the appropriate token as described in the text;
- d) tokens in square brackets, '[' and ']', define tokens that are optional, for example parts of a statement that are only required under special circumstances;
- e) ellipses, . . . , show that the preceding item can be repeated.

Subclause 5.3 defines other typographical and lexical conventions that apply to PCSDL documents.

4 General principles for the PCS

4.1 General structure

4.1.1 Purpose

The main purpose of the PCS is to give an unambiguous way to interpret data transmitted via the IEC 61162-401 A-profile protocol. In this sense, the companion standard adds meaning to the data transmitted via the protocol, converts it to information and makes it usable for application modules connected to the network. To serve this purpose, the PCS shall provide the following:

- a) establish a language to define information types, application interfaces and applications. This language has to be human readable as well as interpretable by a computer. this language is the **PCSDL**;
- b) provide a standardized set of information types and interfaces which can be used as a basis to create customized (i.e. vendor specific) application and interface descriptions. This set of specifications is the **PFS**, see 4.4;
- c) provide a general framework for a high level description of applications that use the IEC 61162-4 standard for communication. This is the function block specification format.

4.1.2 Components of the PCSDL

The PCSDL supports the generation of four different types of specifications as outlined in the following subclauses.

The three first document types can be used to generate protocol (A-profile) related entities, for example data object names, MAU names and format strings. The information specification can be used to add more application related meaning to the information entities. The information specifications can also use an extended format string syntax to implement higher level functionality based on the A-profile specification.

4.1.2.1 Application

Representation of application units within the PCS. An application is defined by application interfaces specifying the respective inputs and/or outputs. The application specification can be looked at as the specification of how one particular piece of equipment is connected to the system. Applications will normally consist of a number of *interfaces* configured as either providing data to or using data from the system.

4.1.2.2 Interface (component)

Specification of connection mechanisms used between applications. Each interface has one or more connection points (CP). Each connection point specifies one or more information type received or provided by the respective interface. The connection point specification consists of the data structures transferred, the information the data structures carry and how the information can be accessed (read, write, etc.).

4.1.2.3 Data type

Data type definition specifies the structure and to some degree the content of data transmitted via the IEC 61162-4 network. This document can be optionally exchanged or complemented with *information specifications*.

4.1.2.4 Information

An information specification represents and defines the content and interpretation of data transmitted via the IEC 61162-4 network. The purpose of these specifications is to define the format of exchanged data and specify the usage of the respective information. This type of document extends and replaces the function of the data type definition by making it possible to give more meaning to the data structures. Any information specification can be converted to a data type definition, but at a loss of information meaning.

NOTE The information type specification is not currently used in the PFS. At this time, only the data type specification is used to define structure and contents of transmitted data blocks. However, it is recommended that the information type is considered for new specifications.

4.1.3 Object based principles in the PCSDL

4.1.3.1 General

The PCSDL supports a simplified object based view on companion standard specifications. To create companion standards, the following object-based principles can be applied.

All specifications used within PCS are seen as encapsulated structures defining their own scope and domain. Each specification can be used as a “building block” to create new specifications following the rules given in clause 5. Two different mechanisms are available to compose new specifications from existing ones: specialization and aggregation.

4.1.3.2 Specialization

The PCSDL gives the opportunity to derive new specifications from existing ones. The new specification (i.e. application, interface or information) inherits all properties from the old specification called the base specification). This mechanism forms a generalized-specialization relation between these two specifications.

4.1.3.3 Aggregation

It is possible to compose new specifications by aggregating two or more existing specifications. A complex information type can, for example be formed by including several information specifications into a new specification. The same applies for composing complex application interfaces from one or more interfaces (see 5.6).

Specifications for applications, interfaces and information are included in three separate hierarchies, see figure 1. Each hierarchy stems from a base specification covering properties common to each specification of the respective type (i.e. information, application or interface). An overview of the PFS hierarchy for applications is given in annex E.

4.1.3.4 Data types versus information

The current version of the PFS does not make use of interface specifications and neither the object-oriented principles inherent in that. Instead, the less advanced data type definitions are used. This is due to legacy companion standards from previous versions of this document (see MiTS in part 400).

NOTE This part of the standard emphasizes the information type documents as it is believed that this will be the description method of choice in the future. However, the reader may want to concentrate on data type documents which are the most commonly used.

4.1.4 Generic and manufacturer specific companion standards

The general approach recommended by PCS is to divide the set of companion standards into two different parts (figure 1): Generic CS and manufacturer specific CS. Generic CS will be defined by the PFS. The PFS contains generic specifications for data types, information, interfaces and applications. These specifications can be used by manufacturers as a starting point to define own specifications to describe their respective equipment. This means that PFS specifications can be used as templates to create equipment specific specifications. This mechanism forms a generalized-specialization relation between the newly created specification and the respective specification of the PFS. It has to be noted that specifications which are part of the PFS are abstract in the sense that they cannot be used directly to specify equipment. A tangible PCS specification of a component has always to be derived from a PFS specification.

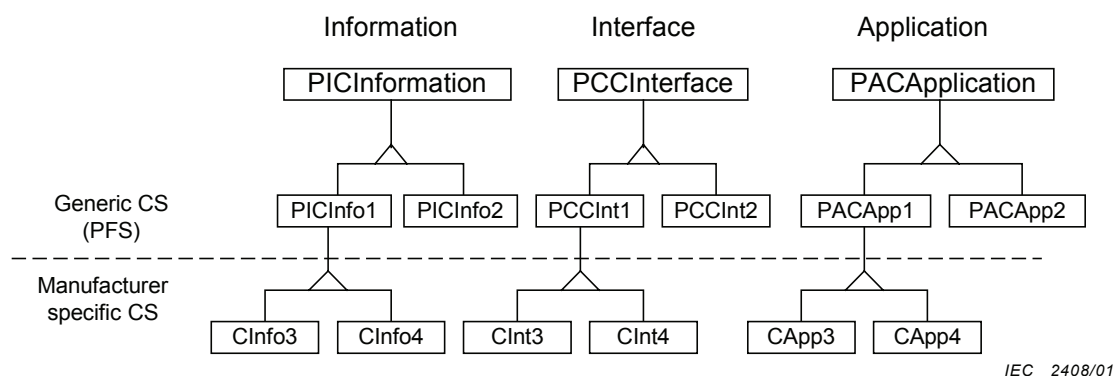


Figure 1 – General structure of PCS hierarchy of specifications

To form specifications of information, interfaces or applications, direct inheritance mechanisms can be used. The new specification inherits all properties from the specification from which it is derived (the base specification). When deriving new specifications from existing ones, attributes only may be added (see also 4.2.5). Thus, the following mechanisms are the only ones permitted by the PCS description language:

- generation of a new application specification: derivation of the new specification from an existing one (abstract or tangible specification) and addition of new application interfaces (see 5.5);
- generation of a new interface specification: derivation from an existing interface specification (abstract or tangible specification) and addition of connection points (see 5.6);
- generation of a new information specification: derivation from an existing specification and addition of attributes, i.e. data types or information (see 5.7);
- creation of new data type specifications (see 5.8).

4.1.5 Generic companion standards (PFS)

IEC 61162-420 defines generic information or data types to be transmitted via the A-profile protocol and standardized interface or application descriptions. With respect to physical components, these generic descriptions define the minimum required functionality (including the information they shall provide) of the component concerned. Generic companion standards, including application, interface, data type or information specifications, are part of this standard and shall not be modified by manufacturers.

Thus, applications, interfaces and information specifications contained in the generic companion standard form abstract specifications and normally it makes little or no sense to use these directly as full specifications for actual applications. Normally, it is necessary to create a more concrete specification (a tangible specification), by deriving it from one of the abstract specifications.

NOTE Generic companion standards provide the least common denominator for information interchange via the IEC 61162-4 protocol. For instance, a physical position sensor will at least own the interfaces defined for a generic position sensor. In this sense, these generic application and interface descriptions have to be in line with the respective IMO performance standards.

The collection of generic specifications defined by the PCS is the **PFS**.

4.1.6 Manufacturer specific companion standards

To satisfy the needs of specific (i.e. physical) components, a manufacturer creates specific companion standards for information type, data types, interfaces or applications. These companion standards are derived from generic companion standards (see 4.2.7). In general, a manufacturer specific companion standard for, for example an interface will contain the content (information types, attributes, etc.) of the respective generic companion standard as a sub-set.

NOTE The main advantage of this approach is that there is no need for a manufacturer to develop specific components from scratch. This accelerates the process of specification production and implementation significantly. The second advantage is that any interface derived from a generic PCS interface description provides as a minimum the information types defined within the respective specification of the PFS: even if a manufacturer specific companion standard is unknown, the services of the respective template can be used. For example if manufacturer B wants to connect his ECDIS system with a GPS receiver of vendor A, the manufacturer can at least receive the information types of the generic GPS interface (i.e. position, satellites in view, valid flag and time), even if he does not know the specification of the customized interface. Only the generic interface used as the template has to be known in this case.

Companion standards for information types, interfaces and applications are defined within hierarchies (figure 1). Information types and interfaces will be derived from the respective base specification defining the properties (attributes) common to all specifications derived from this base. There is also a base specification for applications.

The limit between the generic (normative) and the customized (manufacturer specific) part of the PCS is formed by a horizontal cut within the respective specification hierarchy. This is indicated by the dashed line in figure 1.

Attributes and methods necessary to implement services that are needed from the A-profile will also be defined within the highest-level base specifications. They are implemented by several specialized generic interfaces. These interfaces are described within 4.4.

To provide a high level description of a new application, the manufacturer should also provide a function block specification of the application (clause 7).

4.1.7 Guidelines for using PCS

This subclause gives an overview of how to use the PCSDL to create specific companion standards. The starting point is the need to connect a new application to an IEC 61162-4 compliant network. To implement such a connection, the following steps have to be carried out:

- identify the necessary interaction between the application concerned and other applications in the system. Add possible interactions to possible future applications where appropriate. Group these interactions roughly into draft PCS application interfaces. If possible, the draft should be based on an already existing similar application which is known to conform to the PFS;
- construct a function block that can be used to describe the new application. The function block should identify input interfaces, output interfaces, physical relationships between application and environment and the application's functionality;
- if an appropriate application specification does not already exist, derive a new application specification from a PFS template (i.e. an application base specification contained in the PFS) and adapt it by adding new application interfaces;
- determine the number and type of each connection point in each of the application interfaces. Add new connection points to existing interfaces where appropriate. Sub-divide each application interface into separate interfaces where appropriate (see below) and create the new interface specification documents, either based on existing interface specifications, from scratch or by inheritance from interfaces in the PFS;
- determine input/output formats for each new connection point in the interfaces. Describe this in information and/or data type documents, where possible from existing PCS specifications;
- create the final application specification document by using specialization from the appropriate component in the PFS library and adding the newly developed or modified interfaces.

Note that **application interfaces** are defined only in application specifications. Each application interface consists of one or more interface components, grouped together to form one interface module as seen from the application. Each of the interface components is defined by one interface specification. This approach ensures high flexibility for adapting application interfaces to specific needs of applications by creating a building block system for interfaces.

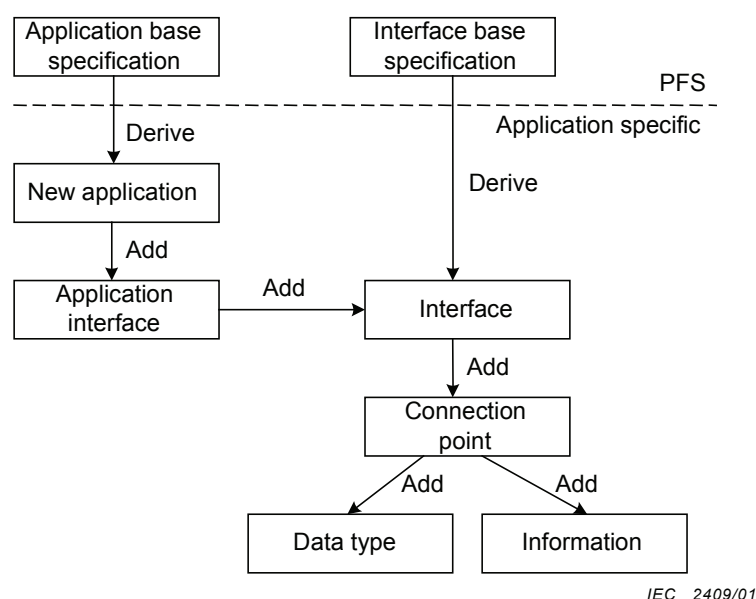


Figure 2 – Generating applications

Figure 2 shows the way to create an application specification derived from a template (base specification, i.e. the PFS).

As stated above, a new application specification must be created by deriving it from an existing PFS compliant specification. The newly created application will inherit all application interfaces from the respective base specification. The only adaptive mechanism allowed is application interfaces. A new application interface is created by grouping one or more existing or newly created interface specifications. As with application specifications, specifications of new interfaces have to be derived from existing specifications (i.e. abstract interface specifications of the PFS or other existing interface specifications). Newly created interfaces will inherit all connection points from the respective base specification.

Newly created interfaces can be adapted to specific needs by addition of connection points. Connection points already defined for that interface (i.e. those specified in base specifications of the interface in view) may not be changed. As shown in figure 2, connection points form collections of data types or information specifications. They may contain zero or an arbitrary number of attributes (data types or information specifications). It is also allowed to have empty connection points. Data types and information specifications shall not be mixed in one connection point.

4.2 Products of the PCS

The PCS is basically a formalized way to describe the system interfaces of applications that use this standard. This means that the PCS must specify certain protocol entities that are used in the establishment of connections between applications and in the exchange of information. These entities are described in the following subclauses.

There are also other entities produced from the PCSDL documents. These are either used in special protocol entities (priorities, load limitations) or in various PFS defined interfaces (authentication and application management). These entities are defined in the last two subclauses.

Note that there is a difference in whether applications are clients or servers of a particular data object. These differences are based on the fact that the server will generally be less configurable than the client.

4.2.1 MAU name

The server MAU name is one of the connection attributes of the data objects used for communication. The *application* document defines the MAU name. Note that the document may specify that the MAU name shall be configurable, in which case the MAU name is defined during system integration by the help of some configuration tool.

4.2.2 Interface name

The interface name is common to a group of data objects that are connected to *en block* during connection establishment. The interface name is one of the data object attributes. The interface name is determined in the *application* document. The interface name can in some cases be configurable. In this case, the ability to configure the name will be made explicit in the *application* document.

4.2.3 Interface class name

If an interface name is changed during configuration, the old interface name as specified in the application document shall be saved in the application and made available to configuration tools as the interface class name.

The interface class name is not used during connection establishment and can in principle be disregarded by applications not conform to the PFS.

4.2.4 Data object name

The data object name is another data object attribute that is used during connection establishment. The data object name is defined by the *interface* document. It is not possible to configure this name.

4.2.5 Data object function

The data object may represent one of several function types, for example read, write or subscribe. The functional capabilities of a data object is determined in the *interface* document. One data object can have only one function, but several data objects with the same name in the same interface can be distinguished between by their function (or data structures). This is similar to name overloading in object-oriented languages.

4.2.6 Data object structure

In addition to function, each data object is also recognized by the input and/or output data structures. The data structures are defined in the *interface* document itself, in a *data type* document or in an *information* document. On the protocol level, it is only the structure of the data element that is important for establishing contact. However, the protocol has provisions for embedding information contents requirements in the data structure (see 4.2.7).

4.2.7 Data object information contents

For meaningful exchange of information, the participating applications need to know more about the sent and received data than just their structure. This aspect is partly covered in the *data type* document, where meaningful interpretations of data structures usually are defined together with the definition of the structure itself. It is also possible to give meaning to the data structures in the *interface* documents in conjunction with defining the functional scope of an interface. A third method is to require additional documentation from a provider of a server MAU. This requirement can be specified in the *application* document.

However, the preferred way to give meaning to data structures is through the *information* document type. This document can either be an addition to the data type document or replace the data type document altogether.

4.2.8 Run-time library information items

Some parts of the companion standard documents generate attribute values for the run-time system that implements a MAU. These items are:

- a) load related attribute values, i.e. number of clients for an accept type interface or transaction queue length for all types;
- b) priority for various connection points or interfaces;
- c) password for interfaces;
- d) watchdog timer for the MAU.

These parameters are set in the application definition header and in conjunction with the definition of accept and connect interfaces in the same document.

Some of the attribute values are most commonly specified as configurable, for example password.

4.2.9 PFS information items

Parts of the companion standard documents do also generate various information items that are used by the PFS. These items are used by software libraries and parts of the PFS to generate various configuration tables. These items are:

- a) version codes for PFS base class;
- b) authentication parameters in application and interface component documents for use in user authentication interface classes;
- c) manufacturer and model information for application management classes;
- d) original classes of interface classes that are renamed in application documents.

4.3 The PISCES foundation specifications (PFS)

The PFS contains the generic part of the PCS. The PFS consists of the following parts:

- a) application foundation specifications;
- b) definition of generic interfaces (see 4.4);
- c) data type specifications.

Application foundation specifications form aggregations of standard interfaces according to the functionality to be covered by the respective generic application.

4.4 Generic interfaces in the PFS

The purpose of generic interfaces specified in the PCS is to give the developer access to the services provided by the A-profile. Annexes in this part of IEC 61162 contain companion standards for the following interfaces:

- a) retrieval of general information about applications and interfaces such as version codes, manufacturer and type of equipment;
- b) interfaces to change/assign control to a specific console combined with authentication of user and workstation, for example to change control, acknowledge alarms or change system parameters;
- c) specialized interfaces of a system MAU associated with each LNA for system management on the application level, for example retrieve network statistics or report attributes of local MAUs;
- d) general mechanisms for retrieving and manipulating data based on tagged information entities. These mechanisms include search, read, write, subscribe and alarm manipulation. The mechanisms can be used for general data access as well as for implementation of alarm systems. This includes an interface to transmit or receive stream based data;
- e) general interface for transmission of IEC 61162-1 telegrams over a system network.

5 The companion standard reference specification

5.1 Introduction

This clause contains the complete reference for the PISCES companion standard description language (PCSDL). It contains all information necessary to understand the companion standards contained in the PFS and to write own companion standards based on the PFS.

Subclause 5.2 explains the basic concepts of the PCS. Subclause 5.3 defines the general conventions with respect to the syntax and semantic of the PCS description language. It explains the use of tokens (identifiers, keywords and literal constants) and the use of white-space elements (delimiters, indentations, comments, etc.). Subclause 5.3.3 concludes with the explanation of naming rules with a particular focus on the scope rules for the identifiers.

As explained in 4.2, the PCS can be used to specify four types of entities: applications, interfaces, information and data types. These four different document types have a common structure, formed by a header specifying general properties for all definitions in the document and a body containing the individual definitions. The body of the definition document normally consists of several blocks, each identified by a keyword. This general structure will be elaborated on in 5.4.

Subclauses 5.5 to 5.8 contain the specification of the description language for each documentation type mentioned above. Each of these sections have the following layout:

- a short overview explaining the purpose and the properties of the document in question;
- a description of the general layout of the respective document. This description includes the general properties than can be set within the header of the document and an overview of the blocks allowed in the body of the document;
- a detailed reference to the syntax of the respective blocks.

5.2 Basic concepts of the PCS

The PCS supports a simplified object-based principle for specifying protocol entities. To create new companion standards, the mechanisms of aggregation and specialization can be used. Specialization means that new companion standards can be derived from existing specifications. The new specification inherits all properties from the base specification from which it is derived (see figure 3).

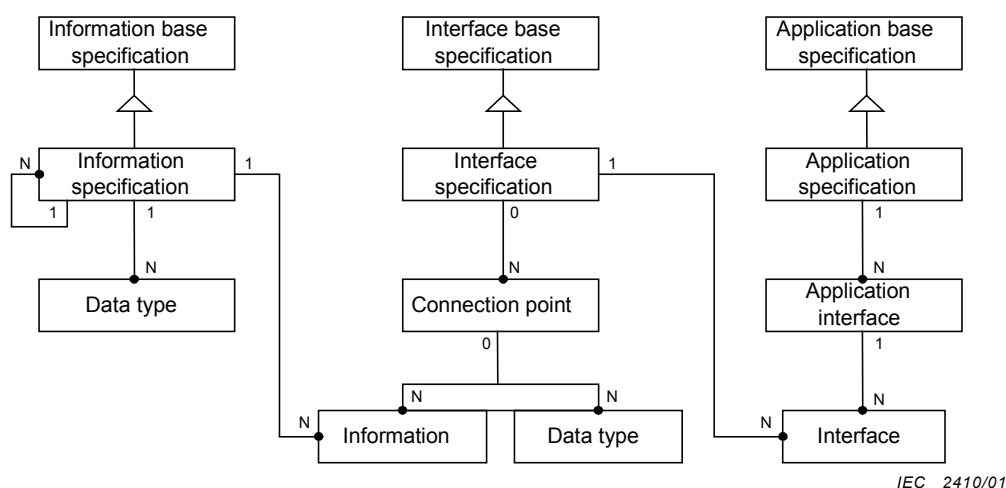


Figure 3 – Relationships between specifications of the PCS

The following types of specialization are possible for PCS:

- derived applications will inherit all application interfaces from the respective base specification;
- derived interfaces will inherit all connection points from the respective base specification;
- derived information specifications will inherit all attributes from their base specification. Attributes may be of the type data or information.

Newly created specifications can be adapted to specific needs by adding properties to them. This mechanism is called aggregation. The following aggregation mechanisms are allowed for companion standards:

- application specifications can be adapted by adding application interfaces to them. Each application interface consists of a set of interfaces. For every interface used, a corresponding interface specification must exist. Application interfaces already defined in base specifications may not be changed;
- interfaces can be extended by addition of connection points. Connection points use one or more information or data types. For each information or data type used a corresponding specification must exist. Connection points already defined in base specifications cannot be changed. Data types and information specifications shall not be mixed in one connection point;
- information specifications can be adapted by adding new attributes to them (data type or information type).

The mechanisms of specialization and aggregation form relationships between the different specification types as elucidated in figure 3.

5.3 Conventions for companion standard specification files

This subclause covers general typographic conventions that apply to the PCS description files. It covers the use of tokens (see 5.3.2) and general naming rules including specification of the scope of named objects (see 5.3.3).

5.3.1 General principles

The files shall contain only characters from the 8-bit ISO 8859-1 character set, except where literal characters or strings are used. For literal character types, it is legal to use 16-bit characters where supported by tools and a computer system.

NOTE The actual representation of long characters must be checked in the programmer's references.

Only printable characters, the *newline* control code and *whitespace* are allowed.

No more than 80 printable characters (including *whitespace*) are allowed on one line. A continuation symbol can be used to continue a long line before a *newline* control code. The continuation symbol is the backslash, '\'.

5.3.2 Tokens

The basic element of the PCS description language is the token. A token in the PCS can be of one of the following types:

- a) *Identifier*: token to refer to a named entity in a specification. An identifier is defined by its name and scope;
- b) *Keywords*: reserved words of the PCS description language (see annex A);
- c) *Literal constants*: tokens to express a constant value. Literal constants can be of several data types (e.g. integer, floating point, character or string);
- d) delimiters;
- e) white space.

The following paragraphs give the typographic conventions that apply to these tokens.

5.3.2.1 Identifiers

Identifiers can be any sequence of letters or numbers. The first character has to be a letter. Capitals and lower case letters will be distinguished between. All characters of an identifier are significant.

5.3.2.2 Keywords

Keywords are always in capitals. A list of the PCS keywords is given in annex A. To enhance readability, keywords should be avoided when used out of context, for example in comments.

NOTE Because the PCS description language distinguishes between capitals and lower case, it is generally legal to use keywords in lower case or in mixed-case variants, for example for identifiers. However, to avoid confusion for the reader, it is not recommended to do so.

5.3.2.3 Literal constants

Literal constants are representing explicit values. Depending on the data type expressed by a literal constant, the following types can be distinguished:

- a) *Integer constant*: representation of integers by a sequence of digits, for example `4123`. The number can be signed or unsigned, in hexadecimal, octal or decimal;
- b) *character constant*: representation of one character (see ISO/IEC 8859-1), for example `'s'`. Long characters are also legal as literal where supported by computers and tools;
- c) *floating constant*: a floating constant consists of an integer part, a decimal sign, a fractional part, an optional `e` or `E` as exponential sign and an unsigned or signed integer exponent. The integer as well as the fractional part consists of a sequence of decimal digits. For instance `3.1415926` or `123.1e-5` are valid floating constants;
- d) *string constant*: a string is a sequence of characters from the ISO/IEC 8859-1 character set, for example `"this is a string"`. Long characters are legal where supported by computer and tools;
- e) *aggregate constant*: this type is used to represent more complex data structures. Aggregates can be records or arrays of the type `{<token1>, <token2>, ..., <tokenN>}`.

A formal definition of the allowed formats of these literal constants can be found in the annexes of IEC 61162-400.

5.3.2.4 Delimiters

Three types of delimiters are used:

- a) *whitespace* (see next paragraph) is used to separate other tokens in the documentation;
- b) *newline* is to be understood as the line separator character(s) used on a given platform, for example a single line-feed (on a UNIX system) or a carriage-return followed by a line-feed (on PCs);
- c) *block separators* (see 5.4) are separated by two or more *newline* delimiters, i.e. with one or more empty lines between them.

Whitespace is understood as one or more tokens that act as separators between tokens, but which have no syntactic meaning. Any contiguous number of the tokens listed below is defined to be "one" whitespace.

5.3.2.5 Indentation

For indentation the space character, ' ', shall be used. Indentations should be used to structure the text (e.g. mark one block).

5.3.2.6 Comments

Comments can be placed anywhere in the text to structure the document. The general operator that starts a comment to the end of a line is the semicolon ‘;’. Comments can be at the beginning of a line to mark the entire line as a comment or within a line – in the latter case, the rest of the line is a comment.

NOTE A newline following a whitespace is a newline token even if the whitespace is part of a full line comment.

If a comment is extended over more than one line, it has to be marked with an asterisk, ‘*’, as the first character. A multiple line comment has to be terminated by a block delimiter (two or more newlines).

<code>;-----</code>	Comment over an entire line.
<code>DATA BLOCK Time</code>	
<code> * A representation of</code>	Multiple line comment with
<code> relative time</code>	indentation, delimited by newlines.
<code>word32_m sec ;seconds</code>	
<code>word32_m usec ;micro-seconds</code>	Declaration followed by comment

IEC 2411/01

Figure 4 – Example comments

NOTE Although the comments described here are not evaluated by a computer reading the description files, they have a significant impact on the human readability of PCS documents and are highly recommended.

5.3.3 Name structure

Identifiers for data types (interpretation, data blocks or constants), application, interface or information specifications may contain a mix of upper and lower case letters, digits and the special character underscore, ‘_’. They should normally start with an upper case letter.

NOTE The A-profile uses these identifiers as attribute values during connection establishment, but does not enforce any rules on their construction other than that they do not contain the null character.

Other identifiers, for example attribute names, may contain the same mix of printable characters. For uniformity in specifications, they should start with a lower case letter.

5.3.4 Name scope

The scope of an identifier depends on its type. Identifiers declared within application, interface or information specifications are local to the respective specification. Dependent on specification type, this affects the identifiers as follows:

- *applications*: identifiers for interfaces specified within application specifications are local (5.5);
- *interfaces*: identifiers for connection points defined within an interface or identifiers for connection points defined in another interface referenced and reused by this interface are local to this interface (5.6);
- *information specifications*: identifiers for attributes to the information type are local (5.7);
- *data types*: these are scope dependent on being declared as local or global (see below).

Data type specifications contain two different sections, one for local and one for global variables. These sections are marked by the keywords `LOCAL` and `GLOBAL`:

- data types (i.e. interpretations, constants and data blocks) within the `LOCAL` section are local to the respective specification. As for the specification types mentioned above, they are accessible from other specifications only via the scope operator, ‘.’;

- data types within the `GLOBAL` section have global scope. This means that they are accessible from all specifications referencing the respective data type specification directly. In this case, the use of the scope operator is optional.

NOTE The purpose of the `GLOBAL` keyword is to allow the definition of data types that can be used without the scope operator in later documents. This simplifies the text of specifications.

The `LOCAL` section has to precede the `GLOBAL` part of data type specifications. If no keyword is used, the section is assumed to be global.

To make reference to a scope identifier, the name of the scope to which it belongs has to be prefixed and followed by a full stop, '.', delimiter. (e.g., `myInformation.m_myMember`).

5.3.5 Configurable identifiers and literal

Some identifiers, for example MAU and interface names may be specified as configurable during system installation. This is typically necessary if several identical units (MAUs) are installed in one system. In this case, each MAU must be given a unique name.

To specify in companion standard documents that an identifier or literal is configurable, it can be enclosed in angle brackets, '<' and '>'. The documentation should specify where and how the entity is configured.

5.4 General structure of PCS specifications

5.4.1 General

PCS documents are used to specify the following entities:

- application specifications;
- interface (specifications);
- information specifications;
- data types.

An outline of the general document is shown in figure 5.

The specification consists of two main parts:

- the specification **header** defines attributes of a general nature and is similar for all document types;
- the **body** of the specification consists of a sequence of **blocks** defining properties specific to the respective entity type. The structure of this part is dependent on the document type.

The header is delimited from the body by a block delimiter.

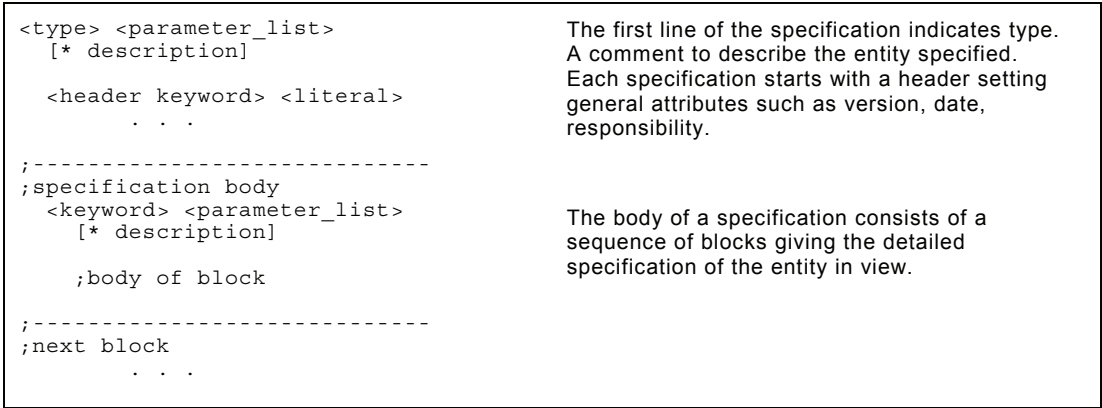


Figure 5 – PCS general structure IEC 2412/01

5.4.2 The specification header

All specifications start with a keyword defining the *type* of entity that shall be specified. <type> may be one of the following:

- a) APPLICATION to specify an application;
- b) INTERFACE for an interface;
- c) INFORMATION to create an information specification;
- d) DATA TYPES to generate a data type specification.

A list of parameters is appended to <type>. The format of this list depends on the type specified. The first parameter of the list always specifies the identifier (i.e. the name) of the entity. For details on the syntax of the specific types, see below.

The first line can be followed by a block comment giving a description of the content of the specification.

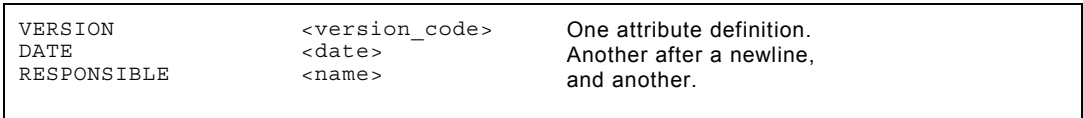


Figure 6 – Header IEC 2413/01

In the rest of the header, each line consists of a keyword naming an attribute to be specified and a literal constant giving the attribute value. An example is shown in figure 6. The lines shall be separated by one or more newlines and/or block comments. The parser will normally look for the first keyword of the body of the specification to determine that the header has ended.

The following header fields are common and required for all PCS specification documents:

- a) VERSION <version_code>: the VERSION keyword sets the version of a PCS entity to the value given by <version_code>. The version code shall be written in the format N.N where N may be any integer-constant. Leading zeros are allowed (e.g. 0.001);
- b) DATE <date>: defines the date of creation of the entity to <date>. The date shall be in the form YYYY-MM-DD with YYYY specifying the year of creation, MM the month and DD the day (all fields are integer constants). For instance 2000-08-03 is a valid date;
- c) RESPONSIBLE <name>: this field specifies the responsible organization and author of the specification. It may be any string constant. This field should normally name the organization and person in the organization responsible for further maintenance of the document.

5.4.3 The specification body

The basic element of the specification body is the block structure. Each block starts with a keyword indicating the type of block followed by a list of parameters. The available types of blocks are dependent on the type of entity to be specified. Details on legal blocks and the respective syntax will be given in the following sections. Blocks are separated from each other by at least one block separator.

5.5 Application specifications

5.5.1 Overview

An application specification is the description of the interfaces of one application unit that shall be connected to the IEC 61162-4 network. Each application specification consists of a set of one or more interface specifications. A PCS application specification, with its referenced documents, contains all necessary information to create the programming code that implements the interfaces of the application unit.

New application specifications can be derived from existing ones. The new specification inherits all properties from the existing application specification, i.e.:

- attribute values defined in the header of the respective base specification;
- all interfaces specified by the base specification.

5.5.2 General layout

Figure 7 shows the general layout of the application specification. Examples of application specifications can be found in the annexes.

The keyword `APPLICATION` defines the start of an application specification. The syntax of the statement is as follows:

```
APPLICATION <appl_name> DERIVED [FROM] <base_application>
```

The identifier `<appl_name>` defines the name of the application (the MAU name). The keyword `DERIVED` shows that the application is derived from a base specification given by the identifier `<base_application>`. This identifier has to be a valid reference to another application specification. The keyword `FROM` is used to make the statement more readable. A tangible application specification shall be derived from a base specification. Only abstract specifications (specifications which are part of PFS) need not to be derived from base specifications.

```

APPLICATION <appl_name> DERIVED [FROM]
<base_application>

    <general header>

;-----
-
REFERENCES
    <interface_name>    [version]
    . . .

;-----
-
USAGE
    [* description]

;-----
--
INTERFACES
    [* description]

    ACCEPT <interface_name>
        [* description]

        <properties of the interface>
        . . .

    CONNECT <interface_name> ON <server_name>
        [* description]

        <properties of the interface>
        . . .

```

Figure 7 – General layout of an application specification

IEC 2414/01

A block comment should follow the first line, giving a short overview of the application specified. It is recommended also to supply a revision history where appropriate. This is not part of the formal syntax and can be included after the description by another comment block.

5.5.3 Header

As for all entity types covered by the PCS, an application specification is divided into a header and a body. In addition to the general properties listed in clause 5.4, the following entries can be defined in the header of application specifications:

```

VERSION          <version_code>
DATE             <date>
RESPONSIBLE      <name>
[MANUFACTURER    <name>]
[MODEL           <name>]
[WATCHDOG        <interval ms>]
[AUTHENTICATION  [<user>:<password>,< user>:<password>, . . . ]]

```

Figure 8 – Application header

IEC 2415/01

- a) MANUFACTURER <name>: this attribute specifies the manufacturer of the application specified here. The name may be any character string;
- b) MODEL <name>: the model name of the application should be given here. This attribute is used, for example to distinguish between different variants of a product family. The name may be any character string;
- c) WATCHDOG <interval ms>: this field specifies an optional watchdog interval (in milliseconds) that is used by the LNA to periodically interrogate the state of the MAU;
- d) AUTHENTICATION <user>: <password>: this field specifies that authentication is used for all interfaces in the MAU and may, optionally, specify the available user and password codes. The latter is normally not included in open documents. Authentication uses a dedicated PFS interface.

5.5.4 Body

The specification body shall consist of three main blocks in the order shown in the following subclauses. All blocks shall be present, although they may be empty or consist of just comments as, for example the `USAGE` block.

5.5.4.1 References

Each external interface has to be referenced before it can be used in the `INTERFACE` blocks. This is done in a block that starts with the keyword `REFERENCES`. Each additional line of this block contains a reference to an interface in the form:

```
<interface_name> [version]
```

`<interface_name>` shall be the name of a valid interface. To distinguish between different versions of one interface, the version code may be specified as a text string. This string, if present, will be compared to the version code specified in the header of the referenced interface. A mismatch shall cause a parser error.

5.5.4.2 Usage

This block gives a description on how the application should be used. The block starts with the keyword `USAGE`, but the remainder of the block must be formatted as a block of comments.

5.5.4.3 Interfaces

The `INTERFACE` block contains the specification of the application interfaces with which the application is provided. Each application interface identifier is prefixed by the keyword `ACCEPT` (for a server interface) or `CONNECT` (for a client interface). The keyword line may be followed by a block comment giving a description of the purpose and functionality of the respective application interface.

5.5.4.4 Accept type interface

The syntax used to specify a server interface is illustrated in figure 9.

```
ACCEPT <interface_name>
  [* description]

  [MAX MESSAGE RATE <msg_per_sec>]
  [AUTHENTICATION    [<user>:<password> . . .]]
  [CLIENTS           <numb_clients>]
  [TRANSACTION QUEUE <num_trans>]
  [PASSWORD           [<passwd>]]

  INTERFACE [COMPONENT] <name> . . .
```

Figure 9 – Accept interface template

IEC 2416/01

The specification of a server interface starts with the keyword `ACCEPT`. The newly created interface will be named `<interface_name>`. The name may be specified as configurable by enclosing it in angle brackets. The interface will be composed of a number of interface components as specified at the end of the example.

A number of properties can be specified for the server interface by using a set of property definition statements. These statements must be placed between the header line and the first interface component definition. They may be listed in any order. Each statement shall be one line, separated from other lines by one or more newlines. The following properties may be set for a server interface:

- a) MAX MESSAGE RATE <msg_per_sec>: this entry is not converted to a protocol entity. It is used to check and verify system and module load. It shall specify the maximum number of transactions (including connection attempts) that the server application accepts on this interface;
- b) AUTHENTICATION <user>:<password>: if user authentication is required for the respective interface, this property has to be defined. The user and password fields may optionally be specified to list the available users. The password may be specified as configurable;
- c) CLIENTS <numb_clients>: the maximum number of clients that are allowed to be connected to the server may be specified here. The token <numb_clients> is of integer type. This property may be omitted if the maximum number of clients is not limited. The limitation is enforced by the LNA by denying further connection attempt, after the number of clients has been accepted by the MAU. The limitations do not apply to *urgent* connection attempts;
- d) TRANSACTION QUEUE <numb_trans>: the maximum number of pending transaction requests that the server allows in the input queue may be specified here. The token <numb_trans> is of integer type. This property will limit the number of transactions sent to the server for a given interface. By delaying acknowledgements, the server can use this limit to control its load. The limitation does not apply to *urgent* requests;
- e) PASSWORD [<passwd>]: if the server shall be protected by a password it may be specified here. Note that the password normally will be configurable and that the string specified as <passwd> usually is empty or a string in angle brackets. The string will be the default password for the interface.

The interface connection points are specified by a number of the following statements:

```
INTERFACE [COMPONENT] <name>
```

Each application interface consists of one or more interface components, each specified as in the line above. All components specifications must be in one block (only one newline between each line). The keyword [COMPONENT] is normally used to make the code more readable. <name> has to be the name of a valid interface specification as defined in the REFERENCES section (see above). The complete interface (consisting of all components) will be instantiated as one A-profile interface and given the name set in the ACCEPT statement.

5.5.4.5 Connect type interface

The specification of a client interface is initiated by the keyword **CONNECT**. The definition is similar to the accept type interface definitions, except for the properties that can be set and the definition of a server to connect to.

The identifier <server_name> has to be a valid name of an application acting as the server for the respective application interface (the server MAU name). The <interface name> must likewise be the name of one of the application interfaces on the server. Both these names may be specified as configurable.

```
CONNECT <interface_name> ON <server_name>
  [* description]

  [MAX MESSAGE RATE <msg_per_sec>]
  [PRIORITY [interface_comp[.conn_point]] <priority>]
  [TRANSACTION QUEUE <num_trans>]
  [PASSWORD <passwd>]
  . . .

INTERFACE [COMPONENT] <name>
  . . .
```

Figure 10 – Connect application interface template

IEC 2417/01

The following properties may be set for a client interface:

- a) `MAX MESSAGE RATE <msg. per sec.>`: this entry has the same function as for the `ACCEPT` statement. It shall specify the maximum number of transactions generated by this client application for this interface. This is a measure that may be enforced by the application, but it is normally only tested against and is used as a guideline for total system load calculations;
- b) `PRIORITY [interface_comp[.conn_point]] <priority>`: the `PRIORITY` flag can be used to set the priority of an interface component or a specific connection point. If not specified, the priority will be set to `NORMAL`. The connection point of the interface component affected by this property is specified by the name argument. If the name argument is omitted, the priority applies to all connection points in the application interface. If the `.conn_point` component is omitted, the priority applies to all connection points in the interface component specified. The constant-token `<priority>` shall have the values `NORMAL` or `URGENT`;
- c) `TRANSACTION QUEUE`: this is as for the accept interface;
- d) `PASSWORD`: this is as for the accept interface, except that the password specified is that used for connection to the server.

The remainder of the specification consists of one or more lines as follows:

```
INTERFACE [COMPONENT] <name>
```

These lines are used to specify the components of the application interface in the same format as for the accept interface.

Note that a connect interface may contain a sub-set of components that the server supports in its interface. It cannot, however, have components that are not supported by the server. This will result in a run time connection failure.

5.5.4.6 Handling of anonymous broadcast

If the application interface name is `ABCMn` or `ABCn` where `n` is a digit from 1 to 5, the interface components shall consist of only anonymous broadcast connection points. For accept interfaces, this means that a MAU name is disregarded.

These connection points shall be exported as listening (connect) or sending (accept) on the specified broadcast port.

5.6 Interface component specifications

5.6.1 Overview

Interfaces are used in application specifications to describe the functionality and connectivity of the application. The application specification defines application interfaces which consists of a number of interface components. Each of the interface components is defined in an interface definition document.

The format of interface component descriptions is defined in this clause. Interface components themselves form aggregations of connection points. Each connection point specifies input and/or output information available from the interface.

NOTE In essence, interfaces describe the view on an application from the outside world – from an object-oriented point of view, interfaces specify the methods giving access to the internal structure and behaviour of an application object. That means that all functionality and behaviour of an application visible from other applications is given by the interfaces.

5.6.2 General layout

Figure 11 shows the general layout of the specification for an interface component. Examples of interface specifications can be found in the annexes. The following subclauses describe the document in more detail.

```

INTERFACE <interface_name> DERIVED [FROM] <base_specification>

    <header>

;-----
REFERENCES
    <info_name> [version]
    <data_type> [version]
    . . .
;-----
USAGE
    [*description]

;-----
REQUIRED DOCUMENTATION
    <entity_name> [version]
    . . .
;-----
CONNECTION POINTS
    <connection_type> <connection_name>
    [* description]

    INPUT
        [* description]

        <element declaration>
        . . .

    OUTPUT
        [* description]

        <element declaration>
        . . .

```

Figure 11 – General layout of an interface specification

IEC 2418/01

5.6.3 Header specification

The keyword `INTERFACE` defines the beginning of an interface component specification. The syntax is as follows:

```
INTERFACE <interface_name> DERIVED [FROM] <base_specification>
```

The identifier `<interface_name>` defines the name of the interface specification. This is later used in application specification documents to identify the interface component. A block comment should normally follow the first line, explaining properties and usage of the interface component. The keyword `DERIVED` indicates that the interface is derived from a base specification given by the identifier `<base_specification>`. This identifier has to be a valid reference to another interface specification. The keyword `FROM` can be used to make the semantics easier to understand.

It is mandatory to derive a tangible interface specification from a base specification. Only abstract specifications (parts of the basic PFS) need not be derived from base specifications.

As for all entity types covered by the PCS, an interface specification is divided into a header and a body. The properties to be set in the header are given in 5.4.