International Standard

ISO 22166-202

Robotics — Modularity for service robots —

Part 202: Information model for software modules

Robotique — Modularité des robots de service —

Partie 202: Modèle d'information pour les logiciels

First edition
2025-03

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Reference number
ISO 22166-202:2025(en)

© ISO 2025

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

COPYRIGHT PROTECTED DOCUMENT

© ISO 2025

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents. ISO shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 299, *Robotics*.

A list of all parts in the ISO 22166 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Introduction

This document provides an information model for software modules based on ISO 22166-1 and ISO 22166-201. The information model for software modules defined here is based on current industrial practices and recent research results in model-driven system engineering. It is designed to enhance the interoperability, reusability and composability of modules. The document presents guidelines for interoperability, reusability and composability of modules for ensuring their effective connectivity and their correct functionality providing the relevant information for safety/security.

Using the information model for software modules, robot system builders and developers of composite modules (as defined in ISO 22166-1) can effectively compare modules from different vendors, identify the modules matching their requirements and easily integrate them to create a service robot, a subsystem thereof, or even larger composite modules. The information model can be considered a digital datasheet for software modules for service robots. This datasheet provides pertinent information for module users. This, encompasses anyone who integrates the module with other modules and components (as defined in ISO 22166-1) to develop a service robot or any of its subsystems, like composite modules (as defined in ISO 22166-1).

It is the task of the module provider to also provide the information described in this document in the form described herein. And users of such modules, such as robot system integrators, are able to utilize the model information when designing a service robot and assessing its consistency and suitability. This document addresses two main user groups: service robot makers, including service robot integrators, and providers of modules for service robots. Within these user groups, it primarily targets software developers, including system designers and integrators. For module providers, it further addresses personnel concerned with technical documentation and technical marketing. In the case of service robot makers and service robot integrators, it further addresses personnel concerned with safety engineering and personnel concerned with acquiring modules for use in a robot. Providers of modules for service robots have to provide as complete and accurate a model as possible for their modules, including information about the module's runtime requirements, interfaces and information relevant to safety assessments of systems built from such modules. Robot system integrators have to be able to make design decisions based on the information provided by the module makers. Third parties can use the information to develop tools to automate or support aspects of the software system integration process.

The information model of the robot software modules presented in this document is focused on the common characteristics that all types of software modules have, for example:

a) module ID

b) properties

c) input and output variables

d) status

e) services

f) infrastructure

g) safety and security

h) modelling

i) executable forms

This document focuses on the interfaces, properties, variables, behaviour and status of software modules.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Robotics — Modularity for service robots —

Part 202:
Information model for software modules

1 Scope

This document specifies requirements and recommendations for information models for software modules used in service robots. This document specifies the information model for software modules related to nine principles in ISO 22166-1.

It specifies a structured method to define the characteristics of a software module, or a module that has a software-related interface (modules with software aspects, as defined in ISO 22166-1).

This document is not a safety standard. However, it specifies the information necessary for software modules, including safety-related information.

This document focuses on interfaces, properties, composition and execution-specific information, which are related to software modules. The information is utilized in the runtime and design/developing stages. In particular, the interfaces are classified and described into two types such as variables and methods. The document can also be applied to the following software lifecycle stages: the design stage, development stage, operation stage, and maintenance stage.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 22166-1:2021, *Robotics — Modularity for service robots — Part 1: General requirements*

ISO 22166-201:2024, *Robotics — Modularity for service robots — Part 201: Common information model for modules*

IEEE/Open Group 1003.1-2017, *IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(TM)) Base Specifications, Issue 7*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply:

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
information model
IM
abstraction and representation of the entities in a managed environment, their properties, attributes and operations, and the way that they relate to each other

[SOURCE: ISO 22166-1:2021, 3.1.11]

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

3.2
common information model
CIM
information model that modules most frequently use in service robots

Note 1 to entry: This model is a kind of meta model.

[SOURCE: ISO 22166-201:2024, 3.2, modified — Note 1 to entry has been added.]

3.3
module
component or assembly of components with defined interfaces accompanied with property profiles to facilitate system design, integration, interoperability, and re-use

Note 1 to entry: A module can be an assembly of modules.

[SOURCE: ISO 22166-1:2021, 3.3.12, modified — Notes 1 to 4 to entry have been replaced with a new note.]

3.4
module property
property
attribute or characteristic of a module

[SOURCE: ISO 22166-1:2021, 3.3.14, modified — The example has been deleted and the preferred term "property" has been added.]

3.5
software module
SW module
module whose implementation consists purely of programmed algorithms

Note 1 to entry: A software module has software aspects. It consists of software components.

[SOURCE: ISO 22166-1:2021, 3.4.4, modified — The preferred term "SW module" has been added.]

3.6
hardware module
HW module
module whose implementation consists purely of physical parts, including mechanical parts, electronic circuits, and any software, such as firmware, not externally accessible through the communication interface

Note 1 to entry: A hardware module has hardware aspects. It consists of hardware components.

[SOURCE: ISO 22166-1:2021, 3.4.3, modified — Examples 1 and 2 have been deleted.]

3.7
module with hardware aspects and software aspects
hardware-software module
HW-SW module
module whose implementation consists of physical parts, software, and a communication interface that allows data exchange with other modules

[SOURCE: ISO 22166-201:2024, 3.7, modified — The preferred term "HW-SW module" has been added.]

3.8
instance
particular entity instantiated from a specific software module

Note 1 to entry: In object-oriented programming, “instance” means a specific realization of an object.

[SOURCE: ISO 22166-201:2024, 3.9, modified — The hardware-related part in the definition was removed and Note 2 to entry was removed.]

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

3.9
component
part of something that is discrete and identifiable with respect to combining with other parts to produce something larger

Note 1 to entry: Component can be either software or hardware. A component that is mainly software or hardware can be referred to as a software or a hardware component, respectively.

Note 2 to entry: Component does not need to have any special properties regarding modularity.

Note 3 to entry: Component and module have been used interchangeably in general terms, but to avoid confusion the term module is used to refer to a component that meets the guidelines presented in this document.

Note 4 to entry: A module is a component, whereas a component does not need to be a module.

[SOURCE: ISO 22166-1:2021, 3.2.1]

3.10
middleware
software that helps to make communication and data management easy in distributed applications

Note 1 to entry: Middleware provides services to software applications beyond those available from the operating system.

Note 2 to entry: Middleware can be a component on which groups of components and/or modules are executed. ROS1),[1] OpenRTM,[2] and OPRoS[3],[4] are types of middleware.

3.11
hardware abstraction interface
HAI
abstraction interface for a component/module that contains hardware aspects, with the abstraction interface providing control of the component/module via a software interface

3.12
sensing software module
software module for collecting or acquiring data about the world around the robot or the state of the robot for use by other modules to support the robot system in performing its task(s)

Note 1 to entry: The module can access HW-SW modules such as LiDARa, encoders, and cameras using HAI, device drivers or dedicated drivers.

EXAMPLE LiDAR sensing software modules, camera sensing software modules, force sensing software modules, etc.

4 Information model for software modules

4.1 General requirements

The information model for software modules (or software information model, SIM) shall consist of items provided in Table 1, which is based on ISO 22166-201, the common information model. In the document, the symbols ‘M’, ‘O’, and ‘C’ represent Mandatory, Optional, and Conditional, respectively.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

1) ROS is a trademark of Open Source Robotics Foundation, Inc. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of the product named.

Table 1 — Software information and the corresponding tag

No.	Items	SW module	Related group/tag name (abbreviation of each group)
1	Module Name	M	GenInfo
2	Description	O	
3	Manufactures	M	
4	Examples	O	
5	Information model version	M	IDnType
6	Module ID	M	
7	Software Aspects	O	
8	Module properties [a]	M	ModuleProp
9	Inputs	M [b]	IOVariable
10	Outputs		
11	Status	M	-
12	Services (capabilities)	M [b]	Service
13	Infrastructure	M	Infra
14	Safety/security	O	SafeSecure
15	Modelling	O	Modelling
16	ExecutableForm	M	ExecutableForm

[a] This term is only mandatory to properties that can be influenced (set) from the outside or at least to properties that have an expected effect on other modules.

[b] At least one of Inputs/Outputs and Services is mandatory.

The relationship between the common information model (CIM) and the software information model (SIM) is provided in Figure 1. That is, SIM is inherited from CIM. Like CIM, SIM can be described using the Unified Modelling Language (UML).

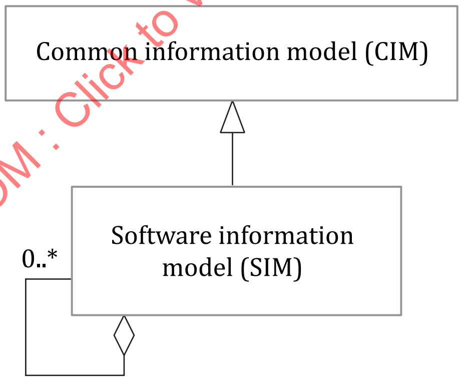
Common information model (CIM)
Software information model (SIM)
0..*

Figure 1 — Relationship between common information model and software information model

NOTE Annex D outlines how to use the SIM and includes information usage among various stakeholders such as the module maker and system integrator.

4.2 Class model of a software information model

4.2.1 General

The software information model shall inherit from the CIM specified in ISO 22166-201 and use the model specified in Figure 2. Access specifiers for the attributes used in SIM shall be public, which are the same as those of CIM.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

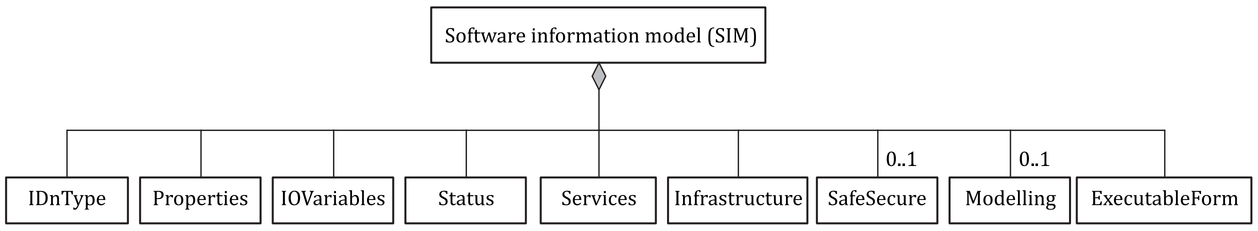

Figure 2 — Class of the software information model

The class of the software information model shall have attributes given in Figure 2 and the attributes for the class SIM are provided in Figure 3.

Values of attributes such as ModuleName, Description, Manufacturer, and Examples are provided in Annex B. The information model version is the version number of the information model used when the module is specified, and is updated whenever the document is upgraded. IDnType, Properties, IOVariables, Status, Services, Infrastructure, SafeSecure, Modelling, and ExecutableForm in Figure 3 refer to the class described in 4.2.2-4.2.10. Most of the information about a module is usually fixed when the module is implemented/manufactured/produced. The information shall be provided in file format. Hence, the information shall not be modified during the execution of the module.

The attributes of the class defined in this document shall utilize the data types specified in Annex H to ensure interoperability and composability.

NOTE 1 The access specifier for an attribute can be one of followings: private (-), protected (*), or public (+). The attribute name is given first and the data types of attributes are defined next. The separation symbol between the attribute name and its data type is a colon ":". If attributes are declared as public, it is not necessary to define the functions to access those attributes.

NOTE 2 Explanations not presented in this document refer to the corresponding part of the CIM.

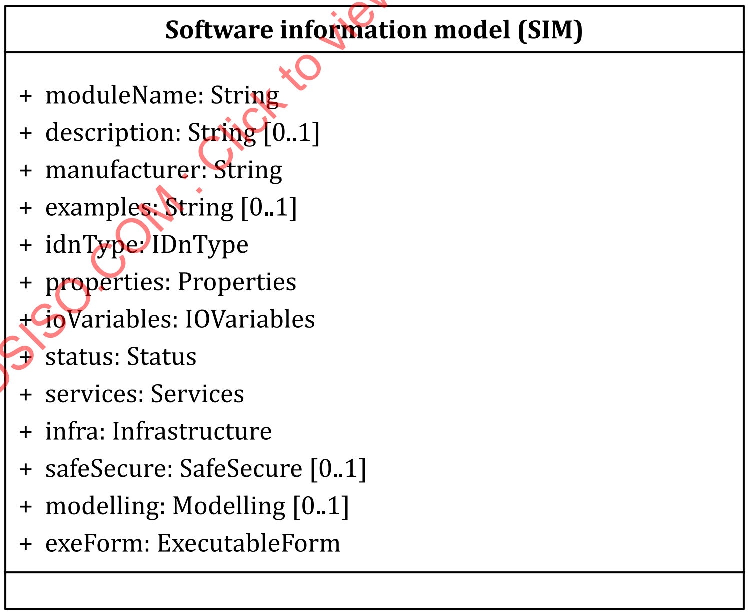
Software information model (SIM)
+ moduleName: String
+ description: String [0..1]
+ manufacturer: String
+ examples: String [0..1]
+ idnType: IDnType
+ properties: Properties
+ ioVariables: IOVariables
+ status: Status
+ services: Services
+ infra: Infrastructure
+ safeSecure: SafeSecure [0..1]
+ modelling: Modelling [0..1]
+ exeForm: ExecutableForm

Figure 3 — Class diagram of class SIM

EXAMPLE Annex C illustrates an information model for a SLAM (Simultaneous Localization and Mapping) module, exemplifying the concept of a composite module based on the class SIM.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

4.2.2 Class for Module ID

Information for Module ID shall be defined in the class IDnType. Class IDnType shall include attributes specified in Table 2 and Figure 4. Values of the attribute, "moduleID" and other attributes in Figure 4 and Table 2 are provided in Annex A and Annex B. SW modules shall be classified into singleton and multiton. The former means that only one instance is created from a SW module and the latter means that multiple instances can be created from a SW module. For singleton type modules, Instance ID (or IID) listed in the class ModuleID in Table 3 shall be zero. For multiton type of modules, IID shall be assigned to an unsigned integer of 8 bit, starting from 0 and increased by one, whenever a new instance is created.

NOTE 1 Types of instances are process type or thread type.

Table 2 — Description of Class IDnType

Description: Class IDnType for the software information model. Refer to: ISO 22166-201:2024, Table 4.6 (Class IDnType) (Detailed attribute descriptions added)				
Derived from: None (Class IDnType of CIM is redefined)				
Attributes:				
moduleID	ModuleID	M	1	The module ID of a module. See Table 3.
informationModelVersion	String	M	1	Version number of the information model
hwAspects	ModuleID	N.A.	N	This field is not used in this specification.
swAspects	ModuleID	O	N	An array of IDs of constituent modules, whose root module is located at the first level in Figure 5. This field is available only for composite module.

NOTE 2 "N.A." stands for "not available".

Table 3 — Description of Class ModuleID

Description: Class ModuleID consisting ID of module and instance ID				
Derived from: None				
Attributes:				
mID	Octet	M	31	The ID of a module except instance ID. See Figure A.1 and Table A.1. An array with a data type of Octet and a size of 31.
iID	Octet	M	1	Instance ID (IID), default value is 0. See Figure A.1 and Table A.1

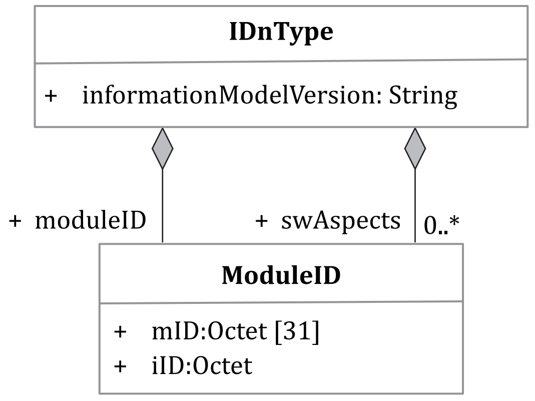

Figure 4 — Relationship between classes for class IDnType

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

4.2.3 Class for Properties

The class Properties is provided in Table 18 and Figure 6. The class Properties shall consist of three mandatory and three optional classes. The mandatory classes are OSType, CompilerType and ExecutionType. The three optional classes are Property, Libraries and Organization. Property shall be defined if a software module uses additional properties except properties defined in the five classes. The class Property is modified based on Table 4.8 in ISO 22166-201:2024 and the class DataProfile is provided in Table 4.7 in ISO 22166-201:2024. Information for the class Property shall be provided using XML or JSON, where the XML format is provided in Annex B.

The class OSType is provided in Table 6, where:

— The attribute "type" is the type of operating system (OS).

— The attribute "bit" is the number of bit that the operating system supports.

— The attribute "version" is the version of the operating system.

EXAMPLE 1 Consider the following as examples of OSs: Windows, Android, MacOS, iOS; Linux OSs: Debian, Gentoo, Ubuntu, Mint, Red Hat, CentOS, Fedora, Kali, Arch, OpenSUSE; Real-time OSs: VxWorks, OSE, VRTX, pSOS, Nucleus, SuperTask, uC/OS, QNX, OS-9, LynxOS, WindowsCE, RTX, Xenomai, RTLinux, and RTAI.[2)]

The class Library is provided in Table 7, where:

— The attribute "name" is the name of a library.

— The attribute "version" is the version of that library.

The class Libraries is provided in Table 8, where:

— The attribute "libraries" includes the list of all libraries used in the module.

EXAMPLE 2 If a module uses libraries such as OpenCV 4.5.3 and Boost 1.76.0, these libraries are included in the class Libraries.

The class CompilerType is provided in Table 10. The class RangeString used in the class CompilerType is provided in Table 9. For class CompilerType:

— The attribute "osName" is the target operating system name.

— The attribute "verRangeOS" is the supported version range of the OS.

— The attribute "compilerName" is the name of the target compiler/interpreter suite.

— The attribute "verRangeCompiler" is the supported version range of the compiler/interpreter.

— The attribute "bitnCPUarch" is the target bit-size and CPU architecture.

NOTE 1 If verRangeOS or verRangeCompiler has only one value, such as only one available version number, the attribute "min" is used for storing the version number and the attribute "max" is set to NULL.

NOTE 2 If verRangeOS or verRangeCompiler is available from the specified version number to the higher version number, the attribute "min" is the specified version number and the attribute "max" is the String "Higher".

EXAMPLE 3 Consider Table 4 including some values of compilers as examples.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

2) Windows, Android, MacOS, iOS; Linux OSs: Debian, Gentoo, Ubuntu, Mint, Red Hat, CentOS, Fedora, Kali, Arch, OpenSUSE; Real-time OSs: VxWorks, OSE, VRTX, pSOS, Nucleus, SuperTask, uC/OS, QNX, OS-9, LynxOS, WindowsCE, RTX, Xenomai, RTLinux, and RTAI are examples of suitable products available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of these products.

Table 4 — Examples of attributes used in the class CompilerType

Attribute	Example OS type 1	Example OS type 2
osName	Windows	Ubuntu
verRangeOS (min, max)	7, 10	18.04, 20.04
compilerName	Microsoft Visual C++ 2017	GCC
verRangeCompiler (min, max)	14.0, Higher	9.0, Higher
bitnCPUarch	64, X86 (min, max)	32, armv7hf

The class ExecutionType provided in Table 13 has the following five attributes: “opType”, “hardRT”, “timeConstraint”, “priority” and “instanceType”, where:

— The attribute “opType” is the operation type of a software module, which shall have one of the following enumeration values: {PERIODIC, EVENTDRIVEN, NONRT}, where “NONRT” means non-real-time, which is provided in Table 11.

— The attribute “hardRT” is used to set whether the hard real-time property has to be satisfied. If so, the value is TRUE. Its default value is FALSE.

— The attribute “timeConstraint” is used to set the period for the operation type of “PERIODIC” and the deadline for the operation types of “EVENTDRIVEN”. Its default value is 0, which means that the related item is not used.

— The attribute “priority” is used to set the instance’s priority of the software module.

— The attribute “instanceType” is related to the instance of a module and shall have one of the following enumeration values: {Singleton, MultitonStatic, MultitonCommutative}, as provided in Table 12.

NOTE 3 Singleton means that only one instance is created. MultitonStatic means that one or more instances are created, but the created instance is linked to only one hardware-software module, such as a sensor module or an actuator module. MultitonCommutative means that one or more instances are created and can be exchanged between modules with the same type.

NOTE 4 The attributes “priority” and “timeConstraint” for MultitonStatic and MultitonCommutative instances can be modified after their creation.

The class Organization, which is related to the composite module, is provided in Table 16. This class has the following four attributes: “owner”, “member”, “dependency” and “additionalInfo”, where:

— The attribute “owner” is the owner of the module and has the module ID value of the owner.

— The attribute “member” represents the child of the module and contains the module ID value of the child.

— The attribute “dependency” shall have one of following enumeration values: {OWNER, OWNED, OWNEROWNED, NONE}, as provided in Table 14.

— The attribute “additionalInfo” stores the additional information for the class Organization.

If a module is a composite type, the structure of the module has a hierarchical type. An example is provided in Figure 5. The attribute “dependency” shall be set using a dependency type in Table 14. All software modules, which are member modules of a composite module, shall have the class Organization with the proper dependency type.

NOTE 5 Member modules of a composite module are listed in the attribute “swAspects” of class IDnType in 4.2.2.

EXAMPLE 4 Figure 5 shows an example of a composite module where dependency types are shown. The composite module consists of two basic modules and one composite module, where the member composite module also consists of one composite module and two basic modules.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

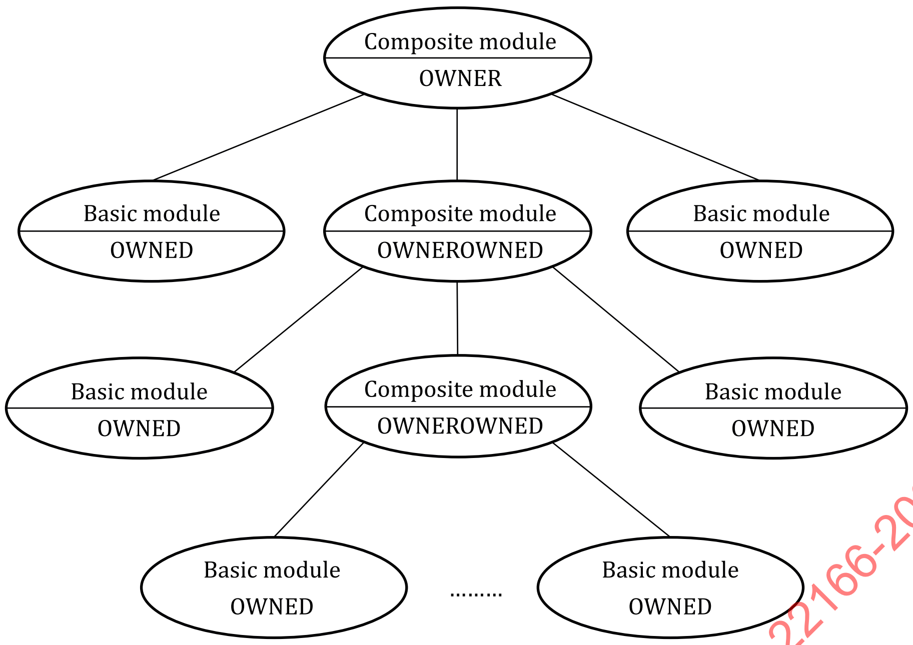

Figure 5 — Example diagram of composite module with dependency type

The class Property is provided in Table 17 and shall inherit the class DataProfile, which is provided in ISO 22166-201:2024, Figure 4.6. An added attribute for SIM is the attribute "immutable", which determines whether it can be changed to a new value during the operation of the module.

Table 5 — Enumeration NoBit

Description: Number of bit processed in OS, enumeration type	
Attributes:	
BIT16	16 bit
BIT32	32 bit
BIT64	64 bit

Table 6 — Class OSType

Description: Defines the OS type that a module uses				
Derived from: None				
Attributes:				
type	String	M	1	OS types such as Windows or Linux, case insensitive
bit	NoBit	M	1	See Table 5
version	String	M	1	OS version number

Table 7 — Class Library

Description: Defines a library that a module uses				
Derived from: None				
Attributes:				
name	String	O	1	Library name, case insensitive
version	String	O	1	Library version number

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Table 8 — Class Libraries

Description: Defines libraries using class Library				
Derived from: None				
Attributes:				
libraries	Library	O	N	array of libraries, case insensitive

Table 9 — Class RangeString

Description: Class representing a range of minimum and maximum values				
Derived from: None				
Attributes:				
min	String	M	1	Minimum value
max	String	M	1	Maximum value

Table 10 — Class CompilerType

Description: Defines a compiler or interpreter that a module uses. This class is related to class ExecutableForm described in 4.2.10				
Derived from: None				
Attributes:				
osName	String	M	1	Target OS name
verRangeOS	RangeString	M	1	Supported version range of OS
compilerName	String	M	1	Name of the target compiler/interpreter suite
verRangeCompiler	RangeString	M	1	Supported version range of compiler/ interpreter
bitnCPUarch	String	M	1	Target bit-size and CPU architecture

Table 11 — Enumeration OpTypes

Description: Operation type of a software module	
Attributes:	
PERIODIC	Instance that operates or runs periodically
EVENTDRIVEN	Instance that operates or runs sporadically according to an occurrence of events
NONRT	Non-real-time instance

Table 12 — Enumeration InstanceType

Description: Instance type that a module can create	
Attributes:	
Singleton	Create only a single instance
MultitonStatic	Create multiple instances but tied to specific hardware such as robot arms or sensors
MultitonComm	Create multiple instances dynamically; its example is just an instance of a module for processing.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Table 13 — Class ExecutionType

Description: Class providing the information needed to run an instance of a module				
Derived from: None				
Attributes:				
opType	OpTypes	M	1	Operation types that a module instance runs (see Table 11)
hardRT	Boolean	M	1	If the instance runs with hard real-time, this value is TRUE. Otherwise FALSE If opType is NONRT, the value of realtime shall be FALSE
timeConstraint	Float	M	1	Unit: micro seconds This value is the period for Periodic instance or deadline for Eventdriven instance
priority	Octet	M	1	Priority of the instance highest priority: 0
instanceType	InstanceType	M	1	Instance type of a module (see Table 12)

Table 14 — Enumeration DependencyType

Description: Dependency type between two modules in a composite module. A composite module has a hierarchical structure (see Figure 5)	
Attributes:	
OWNER	Owner module of a composite module
OWNED	Owned module (or basic module), which is a member module of the composite module
OWNEROWNED	Owner and owned module (or composite module), which is a member module of the composite module
NONE	Module which is not an element of the composite module

Table 15 — Class OrgMemberType

Description: Class representing the Module ID and its Dependency type of a member within Organization				
Derived from: None				
Attributes:				
member	ModuleID	M	N	See Table 3
dependency	DependencyType	M	1	See Table 14

Table 16 — Class Organization

Description: Class representing the structure of a composite module				
Derived from: None				
Attributes:				
owner	ModuleID	M	1	See Table 3
dependency	DependencyType	M	1	See Table 14 (dependency type of owner)
member	OrgMemberType	M	N	See Table 15 (members' IDs and their dependency types)
additionalinfo	NVList	O	1	A list for NameValue for additional information. See ISO 22166-201:2024, Table 4.14.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Table 17 — Class Property

Description: Class Property for software modules,				
Derived from: ISO 22166-201:2024, Table 4.8 (Class Property)				
Attributes:				
value	any	M	1	Value of the property
immutable	Boolean	M	1	true: immutable false: mutable/configurable

Table 18 — Class Properties

Description: Class Properties for Software modules				
Derived from: ISO 22166-201:2024, Table 4.9 (Class Properties)				
Attributes:				
osType	OSType	M	1	Type of operating system used in a module (see Table 6)
libs	Libraries	O	1	Libraries used in a module (see Table 8)
organization	Organization	C	1	Structure of a composite module. If the module is a kind of composite module, this attribute is mandatory
compiler	CompilerType	M	1	Type of compiler/interpreter used for a module (see Table 10)
exeType	ExecutionType	M	N	Execution types of instances generated from a module. The number of ExecutionType equals to the number of instances (see Table 13)
property	Property	O	N	Array of attributes generated according to the class Property (see Table 17)

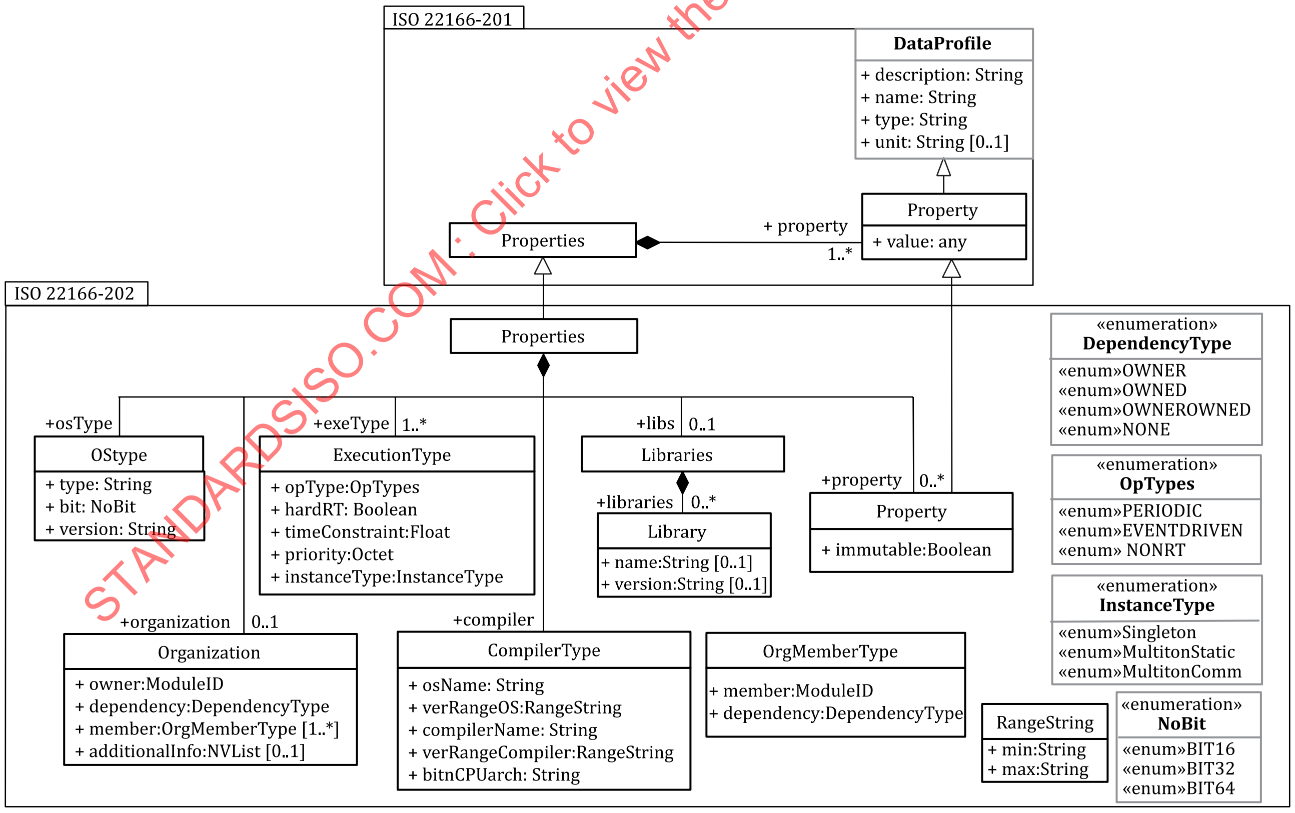

Figure 6 — Relationships between classes for class Properties

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

4.2.4 Class for IOVariables

Information for input and output variables of a module shall be defined in the class IOVariables in Table 19 and Figure 7. The class IOVariables for the software information model shall use the class IOVariables for the common information model defined in ISO 22166-201. For composite modules, the 'moduleID' attribute shall be incorporated into the IOVariables class for the software information model. For classes not specified in this document, refer to ISO 22166-201.

Table 19 — Class IOVariables

Description: Class for input and output variables of a module. Refer to: ISO 22166-201:2024, Table 4.13 (Class IOVariables)				
Derived from: None				
Attributes:				
variable	Variable	O	N	See Table 20

Table 20 — Class Variable

Description: Class for input and output variables of a module				
Derived from: ISO 22166-201:2024, Table 4.12 (Class IOVariables)				
Attributes:				
value	any	M	1	List of attributes generated according to the tag Variable See ISO 22166-201:2024, Table 4.12
moduleID	ModuleID	O	1	This attribute is exclusively used for composite modules and represents the module where this method is utilized or generated (see Table 3)

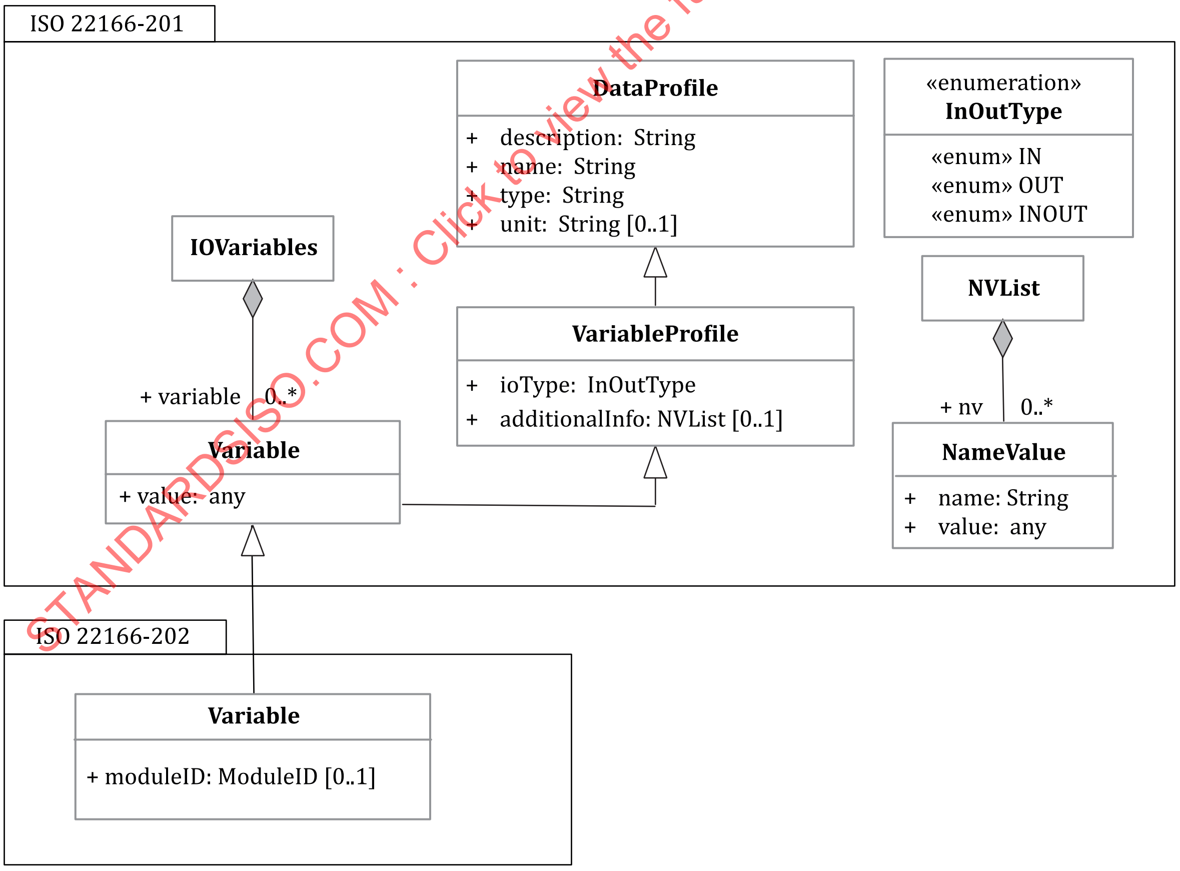

Figure 7 — Relationships between classes for class IOVariables

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

4.2.5 Class for Status

Information for the status of a module shall be defined in the class Status in Table 22 and Figure 8. The class Status provides the status of a module which represents the status of the execution life cycle of a SW module and the error type which occurs in operation.

NOTE 1 The class Status of this document inherits from CIM's class Status, but it does not inherit the “healthCond” attribute of CIM's class Status defined in ISO 22166-201. This attribute is not used for software modules but rather for modules with hardware and software aspects.

Error numbers shall use those defined in IEEE/Open Group 1003.1-2017 IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(TM)) Base Specifications, Issue 7. Additional error numbers can be defined for each module, which shall not conflict with the error numbers in POSIX.1003.1.

NOTE 2 Examples of error numbers used in ErrorType are as follows: Operation not permitted, Authentication Error, Bad Parameter, Unsupported Service, out of Range, and Precondition not met.

NOTE 3 ExeStatus can have one of the following values as defined in ISO 22166-1: CREATED, IDLE, EXECUTING, DESTRUCTED, and ERROR (see Table 21).

Table 21 — Enumeration ExeStatus

Description: ExeStatus represents the execution status of the instance of a module, which is defined in ISO 22166-1	
Attributes:	
CREATED	Created state of an instance
IDLE	Idle state of the instance
EXECUTING	Executing state of the instance
DESTRUCTED	Destructed state of the instance
ERROR	Error state of the instance

Table 22 — Class Status

Description: Class for the status of a module				
Derived from: None [Class Status of CIM (ISO 22166-201:2024, Table 4.17) is redefined.]				
Attributes:				
healthCond	HealthCond	N.A.	–	This field is for hardware modules, and this is not used in this document
executionStatus	ExeStatus	M	1	See Table 21
errorType	Int	M	1	A module shall provide values based on POSIX.1003.1

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

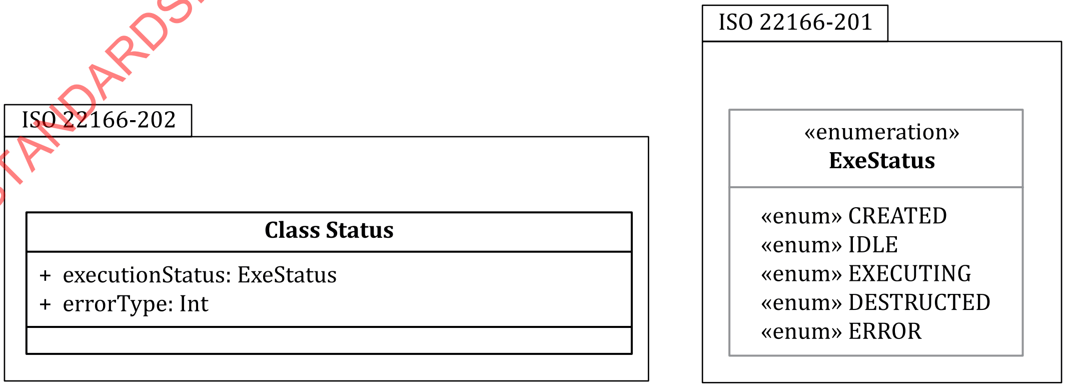

Figure 8 — Class diagram of class Status

4.2.6 Class for Services

Information for services of a module shall be defined in the class Services in Table 23 and Figure 9. The class Services for the software information model shall use the class Services for the common information model defined in ISO 22166-201. For composite modules, the "moduleID" attribute shall be incorporated into the class ServiceMethod for the software information model. For classes not specified in this document, refer to ISO 22166-201.

Information for the class Service shall be provided using IDL or XML or JSON. The IDL format is provided in ISO 22166-201 and the XML format is provided in Annex B.

NOTE 1 The JSON format is not provided in this document, but it can be used through XML.

For software modules, the class Service shall provide interfaces or methods that the module uses or provides in public. Those interfaces are classified into the following two types:

— Interfaces that the module itself provides or uses

— Interfaces that the module uses for access to middleware such as ROS,[1] OpenRTM,[2] OPRoS,[3],[4] etc.

This document describes interfaces that the module itself provides or uses, which is utilized in the design, development and operation stages.

NOTE 2 A brief description of interfaces that the module uses for access to middleware is provided in Annex E.

EXAMPLE An example of an interface that the module itself provides or uses is a ROS service.

Mapping rules and data types between an information model and ROS 2[1] are described in Annex F, which is important when the information model is implemented as ROS 2. Mapping rules and data types between the information model and the OMG SDO[6]/RTC/OpenRTM's RTC[7] profile are described in Annex G, which is important when the information model is implemented as an RTC and OpenRTM profile.

Table 23 — Class Services

Description: Class for services of a module				
Derived from: None				
Attributes:				
NoOfBasicService	UShort	M	1	Number of Basic services provided
NoOfOptionalService	UShort	M	1	Number of Optional services provided
serviceProfile	ServiceProfile	O	N	Prototype for basic (mandatory) Services, serviceProfile can be a list (see Table 24)

Table 24 — Class ServiceProfile

Description: Class for a profile of services				
Derived from: ISO 22166-201:2024, Table 4.20 (Class ServiceProfile)				
Attributes:				
id	String	M	1	Name of service profile One or more service profiles can be provided
ifURL	String	O	1	URL/path of file defined in IDL (see CIMServicePackage in ISO 22166-201:2024, Table 4.3.6) for service
methodList	ServiceMethod	O	N	List of methods for service, defined in XML (see Table 25)
pvType	PhysicalVirtual	M	1	PhysicalVirtual: enumeration data type {Physical, Virtual} (see ISO 22166-201:2024, Table 4.19) Physical (Mechanical/Elec) Method or Virtual (Software) Method
Only one type of two types of ifURL and methodList shall be provided.				

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Table 24 *(continued)*

moType	MOType	M	1	MOType: enumeration data type {MANDATORY, OPTIONAL}. See ISO 22166-201:2024, Table 4.21
moduleID	ModuleID	O	1	This attribute is exclusively used for composite modules and represents the module where this method is utilized or generated (see Table 3)
additionalInfo	NVList	O	1	Arguments and their default initialization values for service. See ISO 22166-201:2024, Table 4.14.
Only one type of two types of ifURL and methodList shall be provided.				

Table 25 — Class ServiceMethod

Description: Class for a service method				
Derived From: ISO22166-201:2024, Table 4.23 (Class ServiceMethod)				
Attributes:				
methodName	String	M	1	Method's name
argType	ArgSpec	O	N	Ordered list of arguments used in a method (see Table 26)
retType	String	M	1	Data type of return value (see Table H.1)
moType	MOType	M	1	Mandatory or optional method. See ISO 22166-201:2024, Table 4.21.
reqProvType	ReqProvType	M	1	Provided or required method. See ISO 22166-201:2024, Table 4.22.
additionalInfo	NVList	O	1	A list for NameValue for additional information. See ISO 22166-201:2024, Table 4.14.

Table 26 — Class ArgSpec

Description: Class for specifying an argument				
Derived From: None				
Attributes:				
type	String	M	1	Data type of an argument used in a method (see Table H.1)
valueName	String	M	1	Name of an argument used in a method
inout	InOutType	M	1	See ISO 22166-201:2024, Table 4.10.
additionalInfo	NVList	O	1	A list for NameValue for additional information. See ISO 22166-201:2024, Table 4.14.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

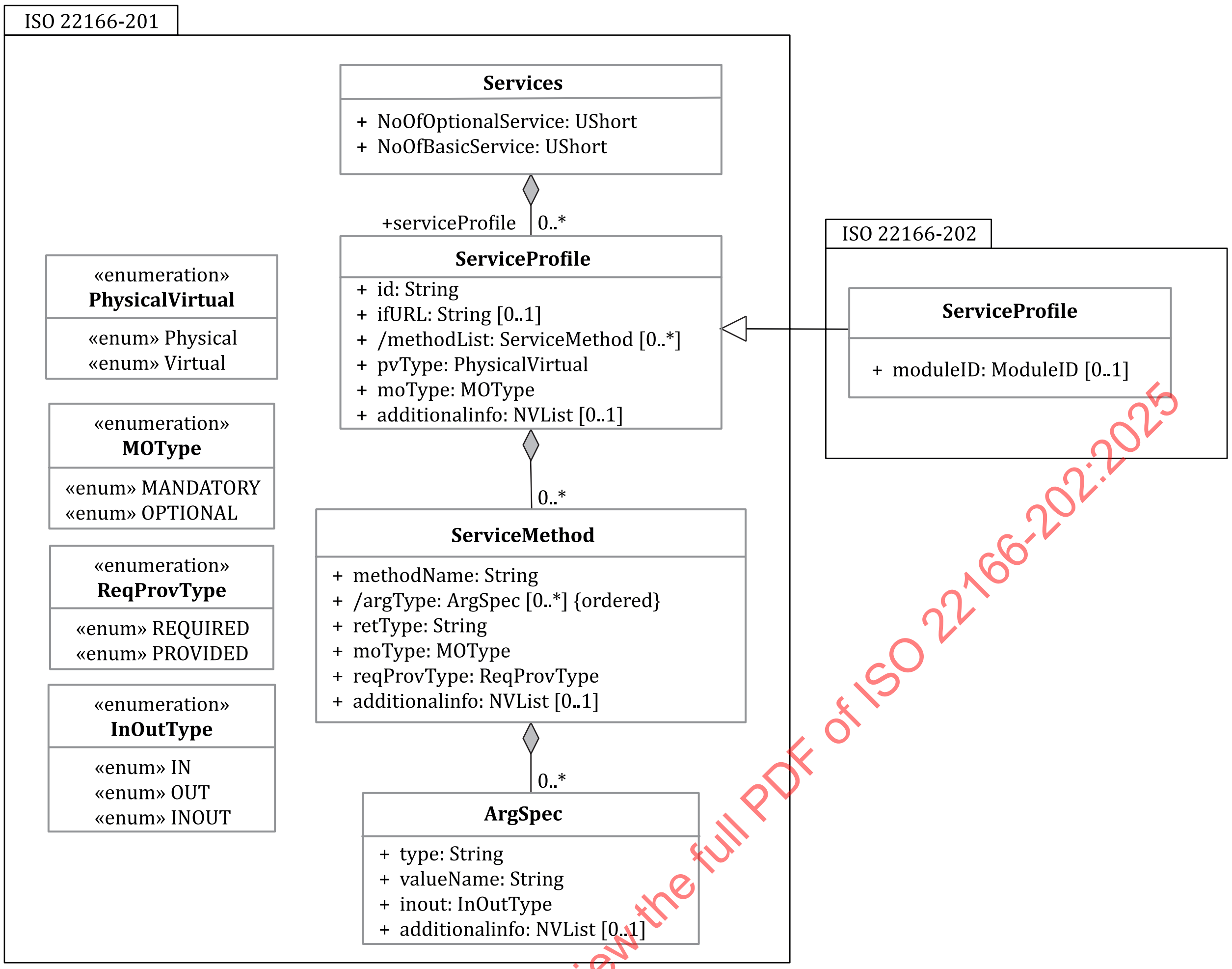

Figure 9 — Relationships between classes for class Services

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

4.2.7 Class for Infrastructure

Information for the infrastructure of a module for a SIM shall be defined in the class Infrastructure in Table 30 and Figure 10. The class Infrastructure defines the following infrastructure modules: database, communication, and middleware for robots.

NOTE Types of middleware used in the class Infrastructure are limited to middleware developed for robots. Examples are ROS, OpenRTM, OPRoS, and OROCOS.

Information for the class Infrastructure shall be provided using XML or JSON. The XML format is provided in Annex B.

The class InfraType is provided in Table 28, where:

— The attribute “name” is the name of a software infrastructure module.

— The attribute “version” is the range of versions of a software infrastructure module.

The class Communication is presented in Table 29, where:

— The attribute “mostTopProtocol” refers to the top-level application protocol

— The attribute “underlyingProtocol” refers to the underlying protocol that supports that application protocol.

The class DataBus is defined in ISO 22166-201:2024, Table 4.22. The application programming interfaces (APIs) designed and used for mostTopProtocol shall be provided using the class Services in 4.2.6.

When using middleware that includes communication, information about the middleware shall be provided using the attribute “middleware” as in Figure 10.

Information related to communication shall not be provided separately. That is, even though the middleware includes communication protocols, the class Communication shall not be defined.

The values of the attributes used in the class DataBus are as follows:

— The attribute “connectionType” uses a String type and can have values such as USB-A, USB-C, RJ45, DB9, DB15, Wireless, and others.

— The attribute “typePhyMac” is also a String type and can have values such as USB, CAN, EtherCAT, Ethernet, EIA 485, WiFi, and more.

— The attribute “typeNetTrans” uses a String type and can have values such as TCP/IP, which includes all basic protocols of TCP/IP.

— The attribute “typeApp” can have values such as FTP, HTTP, BOOTP, DNS, and others, but is set to NULL when only TCP/IP is used.

When providing a new application protocol-based TCP/IP (e.g. RService in EXAMPLE 1 of this subclause), it shall be specified using the attribute “mostTopProtocol”. The APIs provided by the attribute “mostTopProtocol” shall be provided using the class Services in 4.2.6.

EXAMPLE Consider the communication service between the computing server and the service robot. Examples of communication services include controlling the robot, such as starting and stopping it, transmitting the map and the robot's location, detecting obstacles, and so on. It is assumed that the robot uses TCP/IP and WiFi for communication between the robot and the server and ROS middleware. This protocol for communication services is referred to as the RService protocol or simply RService, which is a mostTopProtocol. Instances for the class Communication and the class InfraType for middleware can then be provided as illustrated in Table 27.

Table 27 — Examples for the class Communication and the class InfraType for middleware

Class Type	Variable (or Attribute) name based on Figure 10	Value
Communication	comm.mostTopProtocol.name	RService
	comm.mostTopProtocol.version	(1.0, 1.0)
	comm.underlyingProtocol.typeNetTrans	TCP/IP
	comm.underlyingProtocol.typeApp	NULL
	comm.underlyingProtocol.speed	−1
InfraType	middleware.name	ROS
	middleware.version	(2.0, 2.0)

Table 28 — Class InfraType

Description: Class to describe a single infrastructure module				
Derived from: None				
Attributes:				
name	String	M	1	Name of infrastructure module, e.g. MySQL, FTP, CANopen, HTTP, ROS 2, OpenRTM, OPRoS
version	RangeString	M	1	Valid range of version numbers (see Table 9)

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Table 29 — Class Communication

Description: Class to describe communication protocols used in the infrastructure module				
Derived from: None				
Attributes:				
mostTopProtocol	InfraType	M	N	Can be defined as one or more application protocols used in a module, e.g. FTP, HTTP, Private, CANopen NOTE This part can overlap with the DataBus, but is used to define, from a software module perspective, what application protocols are provided
underlyingProtocol	DataBus	M	1	See ISO 22166-201:2024, Table 4.25

Table 30 — Class Infrastructure

Description: Class to provide information for the infrastructure module for the SIM				
Derived from: ISO 22166-201:2024, Table 4.27 (Class Infrastructure) The attributes power, noBuses, dataBus, dbType, and ipCode are not used in SIM				
Attributes:				
database	InfraType	O	N	List of database management systems used commonly for modules
comms	Communication	M	N	List of communication protocols used commonly for modules
middleware	InfraType	O	N	Middleware used commonly for modules: ROS, OpenRTM, OPRoS, OROCOS, etc.
additionalInfo	NVList	O	1	A list for NameValue for additional information. See ISO22166-201:2024, Table 4.14.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

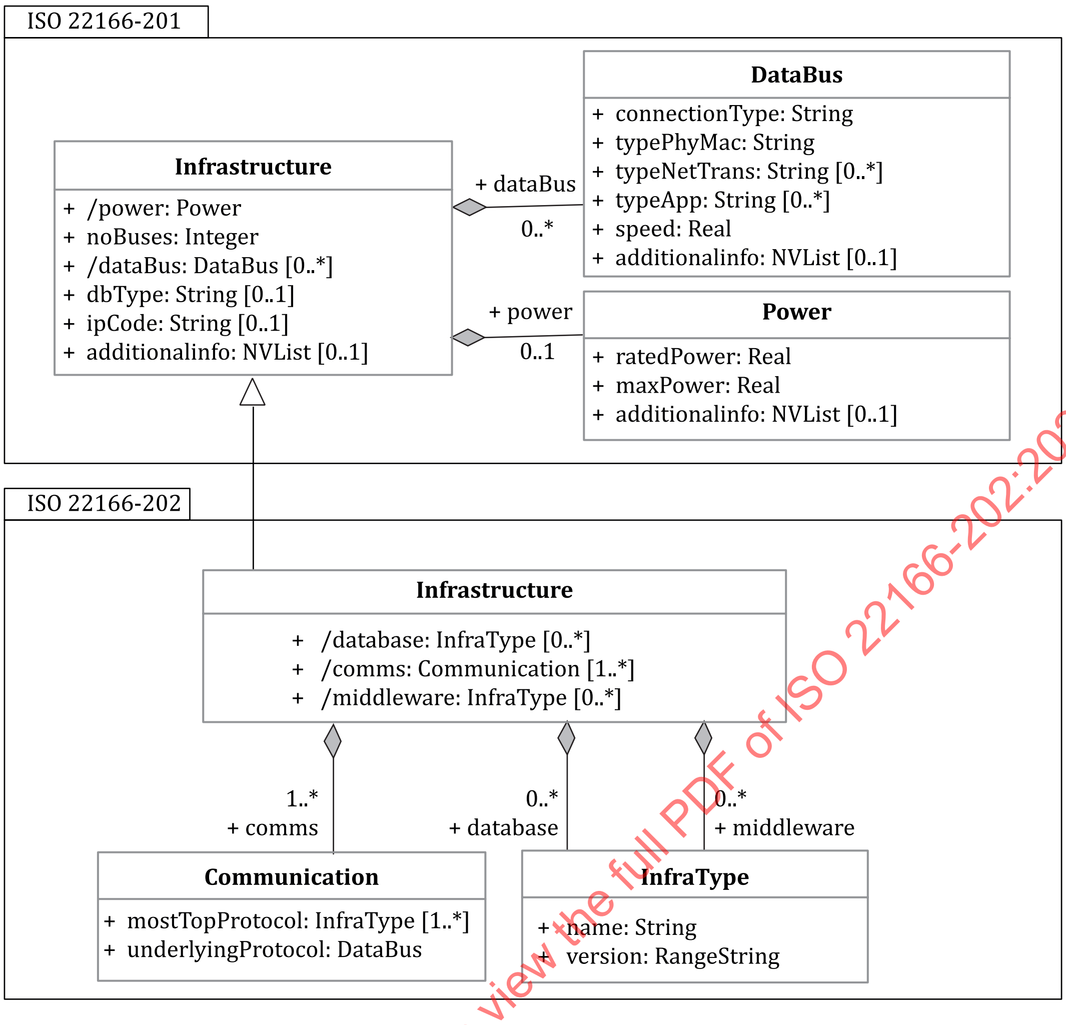

Figure 10 — Relationships between classes for class Infrastructure

4.2.8 Class for SafeSecure

The class SafeSecure class for the SIM shall use the class SafeSecure for the CIM defined in ISO 22166-201:2024, Table 4.38.

4.2.9 Class for Modelling

The class Modelling for the SIM shall use the class Modelling specified in Table 32 and Figure 11, which is inherited from ISO 22166-201:2024, Table 4.40.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Table 31 — Class ModelCase

Description: Used to provide paths to files that provide a model for simulation of a module				
Derived from: ISO22166-201:2024, Table 4.39 (Class ModelCase)				
Attributes:				
simulator	String	M	1	List of simulation programs that a module supports
mdf	String	M	N	Path of model description file or path of directory including the model description files, or “None” A model description file can include a 3D model
libraries	String	O	N	File names of binary/executable codes used for the module's simulation
dynamicSW	ExeForm	O	N	Dynamic programs and/or control programs for the module related simulation
additionalInfo	NVList	O	1	A list for NameValue for additional information. See Table 4.14 in ISO 22166-201.

Table 32 — Class Modelling

Description: Used when providing for multiple simulators, but a 3D model can be used on only one simulator.				
Derived from: Class Modelling of CIM (ISO22166-201:2024, Table 4.40)				
Attributes:				
simulationModel	ModelCase	O	N	Provide information of simulation models used in a module (see Table 31)

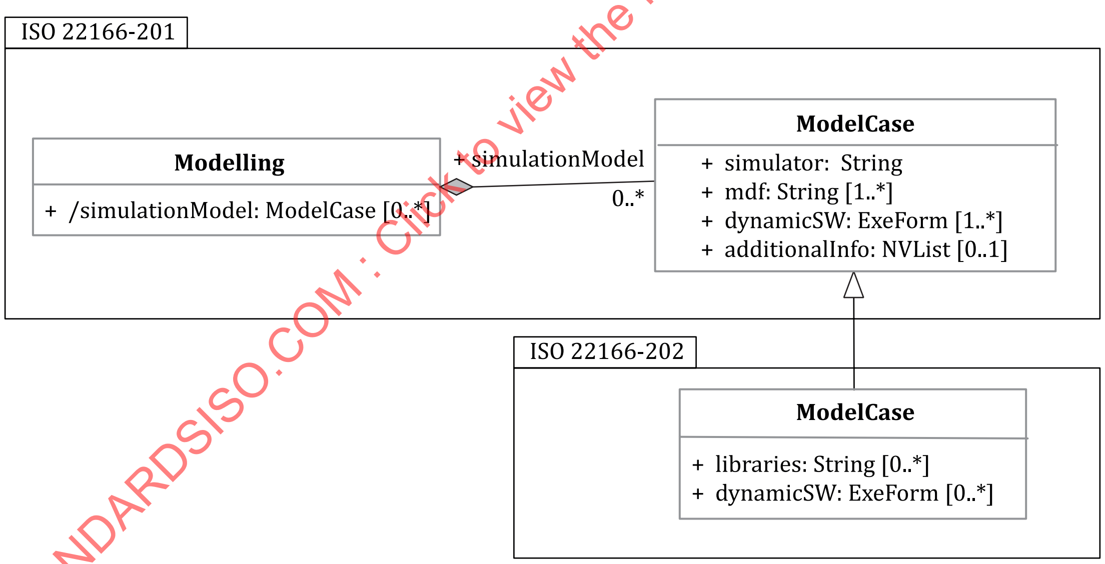
ISO 22166-201
Modelling
+ /simulationModel: ModelCase [0..*]
+simulationModel
0..*
ModelCase
+ simulator: String
+ mdf: String [1..*]
+ dynamicSW: ExeForm [1..*]
+ additionalInfo: NVList [0..1]
ISO 22166-202
ModelCase
+ libraries: String [0..*]
+ dynamicSW: ExeForm [0..*]

Figure 11 — Relationship between classes for class Modelling

4.2.10 Class for ExecutableForm

The class ExecutableForm for the SIM shall use the class ExecutableForm specified in Table 33 and Figure 12, which is inherited from ISO 22166-201.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Table 33 — Class ExecutableForm

Description: Provides the program related information executed to achieve or support the purpose of the module.				
Derived from: Class ExecutableForm in ISO 22166-201:2024, Table 4.41				
Attributes:				
exeForm	ExeForm	M	N	See Table 34
LibraryURL	String	O	N	Binary code or source code used in the module. Source code requires its interpreter. If a library is not needed, this value shall be NULL.

Table 34 — Class ExeForm

Description: Provides the program related information that shall be executed for a module				
Derived from: ISO 22166-201:2024, Table 4.42 (Class ExeForm)				
Attributes:				
exeFileURL	String	M	1	Path of executable file that shall be executed
shellCmd	String	O	N	Command line to execute the “exeFileURL” given as information of the module. This command line is necessary if the program “exeFileURL” utilizes any input in the command line. If not needed, this value shall be NULL.
properties	Property	O	N	See Table 17. Properties and their values required while the program “exeFileURL” is being executed. If not needed, this value shall be NULL.
additionalInfo	NVList	O	1	A list for NameValue for additional information. See ISO 22166-201:2024, Table 4.14.

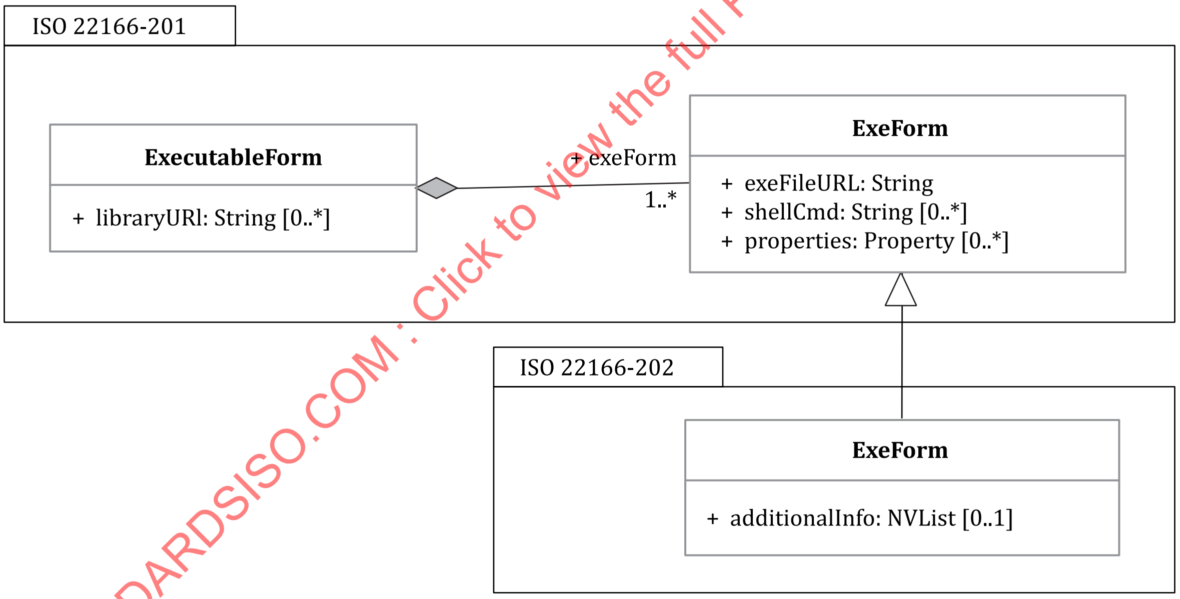
ISO 22166-201
ExecutableForm
+ libraryURl: String [0..*]
+ exeForm
1..*
ExeForm
+ exeFileURL: String
+ shellCmd: String [0..*]
+ properties: Property [0..*]
ISO 22166-202
ExeForm
+ additionalInfo: NVList [0..1]

Figure 12 — Relationship between classes for class ExecutableForm

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Annex A
(normative)

Assignment rule of a software Module ID

A module ID shall consist of six elements. These elements are illustrated in Figure A.1. Their types are given in Table A.1.

The first element of Figure A.1 is the Vendor ID or VID, which shall be assigned by the Universally Unique Identifier (UUID) rev.5 of IETF RFC 4122. The remaining five elements shall be locally assigned by the vendor.

The second element of Figure A.1 is the Product ID or PID, which shall consist of some important information of the module of 1 byte and a product ID of 3 byte. The former shall provide the following five types of information, which are provided in Figure A.1:

— whether the module is composite or basic;

— whether SW aspects exist in the module;

— whether HW aspects exist in the module;

— whether the module provides a safety-related function;

— whether the module provides a security-related function.

The latter shall be the actual product ID, which is represented by alphanumeric characters.

The third element of Figure A.1 shall be the revision number or Rev, which is 4 byte long.

The fourth element of Figure A.1 shall be the serial number or SerialNO, which is 4 byte long, whose value is set to ”00000000” for software modules.

The fifth element shall be the module Category ID, which is 3 byte long.

The sixth element of Figure A.1 shall be the instance ID, or IID, which is 1 byte long represented by numeric characters, 0~255. The default value of the instance ID is 0. If a module has two or more modules of the same type, the modules shall have different instance IDs.

All data in the module ID are represented in hexadecimal code and/or alpha-numeric code.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

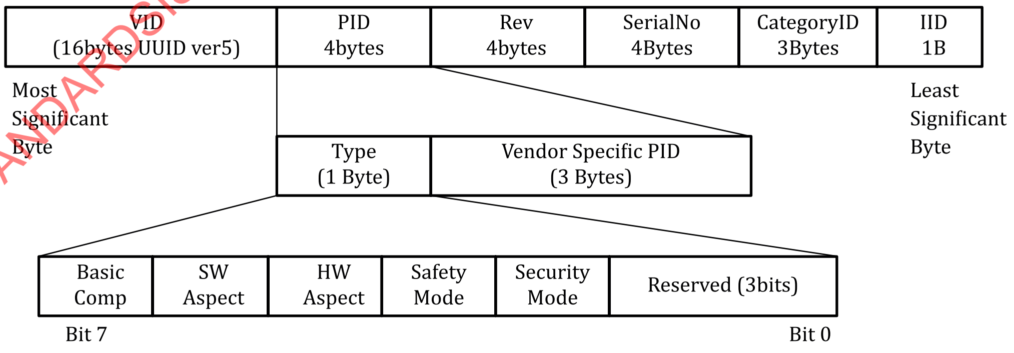

Figure A.1 — Data types of elements of a module ID

Table A.1 — Data types of elements of a module ID

Name	Data Type	Length (Byte)	Remarks
VID	UUID	16	Vendor ID
PID	unsigned char	4	Product ID
Rev	unsigned char	4	Revision number
SerialNo	unsigned long	4	Serial number
CategoryID	unsigned char	3	Category ID
IID	unsinged char	1	Instance ID

NOTE The revision number consists of major number(1byte), minor number (1byte), development state (1byte, if needed), and patch number (1byte) in order.

The CategoryID for software modules shall follow the structure of Figure A.2 and Tables A.2 and A.3. Table A.2 and Table A.3 provide values of level 0 classification for modules and values of the first level and second level classifications for software modules, respectively. Also, some examples of the third level classification are suggested in Table A.3. The numbers in parentheses in Table A.3 are the identification numbers of the relevant classifications, where the numbers are represented in hexadecimal.

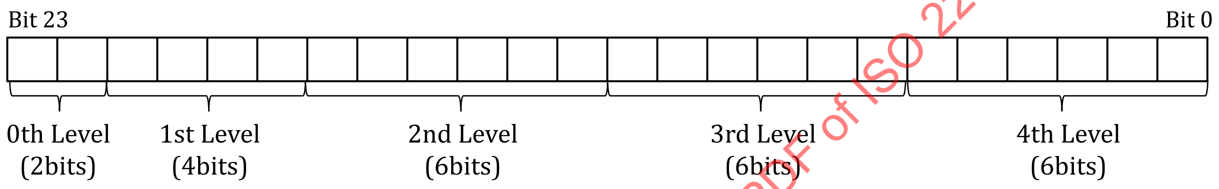

Figure A.2 — Fields of CategoryID

Table A.2 — Classification for level 0 in CategoryID of Figure A.2

Classification	Value	Remarks
Module with HW aspects and SW aspects	00	–
Hardware module	01	–
Software module	10	–
Development/testing tools	11	Tools/modules for development/analysis/testing

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Table A.3 — the first, second and third level classification in CategoryID of Figure A.2

First level	Second level	Third level (examples)
Planning (0000)	Motion planning (000000)	Service robot manipulator planning, trajectory planning, mobile robot planning
	Grasp planning (000001)	
	Task planning (000010)	
Communication (0001)	To/From Server (000000)	
	To/From Other robot (000001)	
	To/From Inner modules of a robot (000010)	(Modules in one computing modules, modules in other computing modules)
	Clouds (000011)	
Interaction (0010)	Speech recognition (000000)	
	Speech generation (000001)	
	Gesture recognition (000010)	
	Structured dialog-based interaction (not speech-based) (000011)	
General Computing (0011)	Localization (000000)	
	Mapping (000001)	
	Feature detection (000010)	
	Generic data transformation (000011)	
	Learning (000100)	
	Control (000101)	
Orchestration/ Management (0100)	Orchestration service (000000)	
	Monitoring service (000001)	Power monitoring, execution monitoring, event logging service
Sensing (0101)	Perception service (000000)	1D scanline perception, 2D image perception, 3D image perception, proximity perception, audio perception, objet-based perception
	Recognition service (000001)	Object recognition, human recognition, environment recognition, situation recognition
	Measurement service (000010)	Range finding, angle, velocity, acceleration, force/ torque, (force, torque, touch), environmental, (temp, wind sonic), motion capturing (leap, vicon), pose (GPS, IMU), RFID, VISION (2D, 3D)
Actuating (0110)	Electrical type (000100)	
	Hydraulic type (000001)	
	Pneumatic type (000010)	
	Hybrid (Elec + Hyd) (000101)	
	Hybrid (Elec + Pneu)(000110)	
	Hybrid (Pneu + Hyd) (000011)	
	Hybrid (Elec+Pneu + Hyd) (000111)	
Reserved (0111 - 1111)	reserved	

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Annex B
(normative)

Representation of common information for software modules

B.1 General

This annex represents information to be used when creating an object for the class defined in Clause 4. The information can be represented in XML or JSON. In this document, the information is represented in XML. XML elements for information of class SIM shall use the same elements used for the common information model defined in ISO 22166-201:2024, Annex C, whose examples are shown in Clause B.3 to Clause B.10. The XML document for SIM shall follow the XML schema provided in Clause B.2.

Parts that differ from or are added to the XML used in ISO 22166-201 are provided. Details are presented in Clause B.2 to Clause B.10. Clause B.3 to Clause B.10 include XML examples and descriptions of XML elements for each class of SIM.

B.2 XML schema for representing information for software modules

The XML schema for a software information module is as follows:

```
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="https://www.w3.org/2001/XMLSchema">

  <xs:element name="NVList" type="NVList"/>
  <xs:complexType name="NVList">
    <xs:sequence>
      <xs:element name="nv" type="NameValue" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="NameValue" type="NameValue"/>
  <xs:complexType name="NameValue">
    <xs:sequence>
      <xs:element name="name" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="value" type="xs:string" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="IDnType">
    <xs:sequence>
      <xs:element type="ModuleIDType" name="moduleID" />
      <xs:element type="xs:string" name="informationModelVersion"/>
      <xs:element type="swAspectsType" name="swAspects" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute type="BasComType" name="type" use="required"/>
  </xs:complexType>

  <xs:simpleType name="BasComType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Bas"/>    <!--Bas: Basic module, See Table B.1 -->
      <xs:enumeration value="Com"/>   <!-- Com:Composite module. See Table B.1-->
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="ModuleIDType">
    <xs:sequence>
      <xs:element type="xs:hexBinary" name="mID" maxOccurs="62" minOccurs="62"/>
<xs:element type="xs:hexBinary" name="iID" maxOccurs="2" minOccurs="2"/>
    </xs:sequence>
  </xs:complexType>
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

```
<xs:complexType name="swAspectsType">
    <xs:sequence>
      <xs:element type="ModuleIDType" name="moduleID" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

<xs:complexType name="PropertyType" mixed="true">
    <xs:sequence>
      <xs:element name="value" minOccurs="0">
        <xs:complexType mixed="true">
          <xs:sequence>
            <xs:element type="xs:string" name="Item" maxOccurs="unbounded" minOccurs="0"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element type="PropertyType" name="Property" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="name"/>
    <xs:attribute type="xs:string" name="type"/>
    <xs:attribute type="xs:string" name="unit" use="optional"/>
    <xs:attribute type="xs:string" name="description"/>
    <xs:attribute type="xs:boolean" name="immutable"/>
    <xs:attribute type="xs:string" name="value"/>
    <xs:attribute type="xs:string" name="complex" use="optional"/>
    <xs:attribute type="xs:string" name="complexName" use="optional"/>
</xs:complexType>

 <xs:complexType name="OStypeType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute type="xs:string" name="type"/>
        <xs:attribute type="NoBitType" name="bit"/>
        <xs:attribute type="xs:string" name="version"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name="NoBitType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="BIT16"/>
      <xs:enumeration value="BIT32"/>
      <xs:enumeration value="BIT64"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="ExeItemType" mixed="true">
    <xs:sequence>
      <xs:element type="opTypeType" name="opType"/>
      <xs:element type="xs:nonNegativeInteger" name="priority"/>
      <xs:element type="xs:boolean" name="hardRT"/>
      <xs:element type="xs:float" name="timeConstraint"/>
      <xs:element type="instanceTypeType" name="instanceType"/>
    </xs:sequence>
  </xs:complexType>

<xs:simpleType name="opTypeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="PERIODIC"/>
      <xs:enumeration value="EVENTDRIVEN"/>
      <xs:enumeration value="NONRT"/>
</xs:restriction>
  </xs:simpleType>

<xs:simpleType name="instanceTypeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Singleton"/>
      <xs:enumeration value="MultitonStatic"/>
      <xs:enumeration value="MultitonComm"/>
</xs:restriction>
  </xs:simpleType>
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

```
    <xs:complexType name="ExecutionTypesType">
      <xs:sequence>
        <xs:element type="ExeItemType" name="Item" maxOccurs="unbounded" minOccurs="1"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="RangeStringType">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="xs:string" name="min"/>
          <xs:attribute type="xs:string" name="max"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>

    <xs:complexType name="CompilerTypeType">
      <xs:sequence>
        <xs:element type="xs:string" name="osName"/>
        <xs:element type="RangeStringType" name="verRangeOS"/>
        <xs:element type="xs:string" name="compilerName"/>
        <xs:element type="RangeStringType" name="verRangeCompiler"/>
        <xs:element type="xs:string" name="bitnCPUarch"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="LibItemType" mixed="true">
      <xs:sequence>
        <xs:element type="xs:string" name="name" minOccurs="0"/>
        <xs:element type=" RangeStringType" name="version" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute type="xs:string" name="name" use="optional"/>
      <xs:attribute type="xs:string" name="version" use="optional"/>
    </xs:complexType>

    <xs:complexType name="LibrariesType">
      <xs:sequence>
        <xs:element type="LibItemType" name="Item" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="OrganizationType" mixed="true">
      <xs:sequence>
        <xs:element type="ModuleIDType" name="owner"/>
        <xs:element type="dependencyType" name="dependency"/>
  <xs:element type="OrgMemberType" name="member" maxOccurs="unbounded" minOccurs="1"/>
        <xs:element type="NVList" name="additionalInfo" minOccurs="0"/>
      </xs:sequence>
  </xs:complexType>

  <xs:complexType name="OrgMemberType" mixed="true">
      <xs:sequence>
        <xs:element type="ModuleIDType" name="member"/>
        <xs:element type="dependencyType" name="dependency"/>
  </xs:sequence>
    </xs:complexType>


    <xs:simpleType name="dependencyType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="OWNER"/>
        <xs:enumeration value="OWNED"/>      <!-- Basic Type -->
        <xs:enumeration value="OWNEROWNED"/>    <!-- Composite Type -->
        <xs:enumeration value="NONE"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="Properties" mixed="true">
      <xs:sequence>
        <xs:element type="PropertyType" name="Property" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element type="OStypeType" name="osType" />
        <xs:element type="ExecutionTypesType" name="exeType" maxOccurs="unbounded"/>
        <xs:element type="CompilerTypeType" name="compiler"/>
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

```
      <xs:element type="LibrariesType" name="Libraries" minOccurs="0"/>
      <xs:element type="OrganizationType" name="organization" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="InoutType" mixed="true">
    <xs:sequence>
      <xs:element type="xs:string" name="name"/>
      <xs:element type="xs:string" name="type"/>
      <xs:element type="xs:string" name="unit" minOccurs="0"/>
      <xs:element type="xs:string" name="description"/>
      <xs:element type="NVList" name="additionalInfo" minOccurs="0"/>
      <xs:element type="InoutType" name="Inout" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="name" use="optional"/>
    <xs:attribute type="xs:string" name="type" use="optional"/>
    <xs:attribute type="xs:string" name="unit" use="optional"/>
    <xs:attribute type="xs:string" name="description" use="optional"/>
    <xs:attribute type="xs:string" name="className" use="optional"/>
  </xs:complexType>

  <xs:complexType name="InputsType">
    <xs:sequence>
      <xs:element type="InoutType" name="Input" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="OutputsType">
    <xs:sequence>
      <xs:element type="InoutType" name="Output" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="InOutsType">
    <xs:sequence>
      <xs:element type="InoutType" name="Inout"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="IOVariables">
    <xs:sequence>
      <xs:element type="InputsType" name="Inputs" minOccurs="0"/>
      <xs:element type="OutputsType" name="Outputs" minOccurs="0"/>
      <xs:element type="InOutsType" name="InOuts" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="Services">
    <xs:sequence>
<xs:element type="xs:string" name="ServiceIDL" maxOccurs="unbounded" minOccurs="0"/>
<xs:element type="ServiceTypeXML" name="ServiceXML" maxOccurs="unbounded" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="ServiceTypeXML" mixed="true">
    <xs:sequence>
      <xs:element type="xs:string" name="ID"/>
      <xs:element type="PhysicalVirtual” name="pvType"/>
      <xs:element type="MOType" name="moType"/>
      <xs:element type="ServiceMethods" name="methodList" minOccurs="0"/>
<xs:element type="ModuleIDType" name="moduleID" minOccurs="0"/>
      <xs:element type="NVList" name="additionalInfo" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="PysicalVirtual">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Physical"/>
      <xs:enumeration value="Virtual"/>
    </xs:restriction>
  </xs:simpleType>
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

```
  <xs:simpleType name="MOType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Mandatory"/>
      <xs:enumeration value="Optional"/>
    </xs:restriction>
  </xs:simpleType>

<xs:complexType name="ServiceMethods">
    <xs:sequence>
<xs:element type="ServiceMethod" name="Method" maxOccurs="unbounded" minOccurs="0"/>
</xs:sequence>
  </xs:complexType>

<xs:complexType name="ServiceMethod">
    <xs:sequence>
      <xs:element type="xs:string" name="methodName"/>
      <xs:element type="ArgSpec" name="argType" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element type="xs:string" name="retType"/>
      <xs:element type="MOType" name="moType"/>
      <xs:element type="ReqProvType" name="reqProvType"/>
      <xs:element type="NVList" name="additionalInfo" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="ReqProvType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="REQUIRED"/>
      <xs:enumeration value="PROVIDED"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="InOutType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="IN"/>
      <xs:enumeration value="OUT"/>
      <xs:enumeration value="INOUT"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="ArgSpec" mixed="true">
    <xs:sequence>
      <xs:element type="xs:string" name="type" minOccurs="0"/>
      <xs:element type="xs:string" name="valueName" minOccurs="0"/>
      <xs:element type="InOutType" name="inout" minOccurs="0"/>
      <xs:element type="NVList" name="additionalInfo" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute type="xs:string" name="type" use="optional"/>
    <xs:attribute type="xs:string" name="valueName" use="optional"/>
    <xs:attribute type="InOutType" name="inout" use="optional"/>
<xs:attribute type="xs:byte" name="order" use="optional"/>
<xs:attribute type="xs:string" name="inDataType" use="optional"/>
</xs:complexType>

  <xs:complexType name="DataBusType">
    <xs:sequence>
      <xs:element type="xs:string" name="connectionType"/>
      <xs:element type="xs:string" name="typePhyMac"/>
      <xs:element type="xs:string" name="typeNetTrans" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element type="xs:string" name="typeApp" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element type="xs:float" name="speed"/>
        <xs:element type="NVList" name="additionalInfo" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="InfraType">
    <xs:sequence>
      <xs:element type="xs:string" name="name"/>
      <xs:element type="RangeStringType" name="version"/>
    </xs:sequence>
  </xs:complexType>

<xs:complexType name="SpeedType">
    <xs:simpleContent>
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

```
        <xs:extension base="xs:string">
          <xs:attribute type="xs:int" name="value" use="optional"/>
          <xs:attribute type="xs:string" name="unit" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>

  <xs:complexType name="TypeAppType">
      <xs:sequence>
        <xs:element type="xs:string" name="Item" maxOccurs="unbounded" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="CommunicationType">
      <xs:sequence>
        <xs:element type="InfraType" name="mostTopProtocol" maxOccurs="unbounded"/>
        <xs:element type="DataBusType" name="underlyingProtocol"/>
      </xs:sequence>
  </xs:complexType>

    <xs:complexType name="Infrastructure">
      <xs:sequence>
        <xs:element type="CommunicationType" name="comms" maxOccurs="unbounded"/>
        <xs:element type="InfraType" name="database" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element type="InfraType" name="middleware" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element type="NVList" name="additionalInfo" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>

    <xs:complexType name="OverallSafetyType">  <!--  this type should be used as the attribute
  -->
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="PLSILType" name="mode" use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>

    <xs:simpleType name="PLSILType"> <!-- Table 4.33 in ISO22166-201 -->
      <xs:restriction base="xs:string">
        <xs:enumeration value="PL"/>
        <xs:enumeration value="SIL"/>
        <xs:enumeration value="BOTH"/>
        <xs:enumeration value="NONE"/>
      </xs:restriction>
    </xs:simpleType>

  <xs:simpleType name="SafetyLevelPL"> <!-- Table 4.31 in ISO22166-201 -->
      <xs:restriction base="xs:string">
        <xs:enumeration value="n"/>
        <xs:enumeration value="a"/>
        <xs:enumeration value="b"/>
        <xs:enumeration value="c"/>
  <xs:enumeration value="d"/>
  <xs:enumeration value="e"/>
      </xs:restriction>
    </xs:simpleType>

  <xs:simpleType name="SafetyLevelSIL"> <!-- Table 4.32 in ISO22166-201 -->
      <xs:restriction base="xs:string">
        <xs:enumeration value="0"/>
        <xs:enumeration value="1L"/>
        <xs:enumeration value="2"/>
        <xs:enumeration value="3"/>
  <xs:enumeration value="4"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="SafetyFunction">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="SafetyType" name="safetyFunctionType"/>
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

```
        <xs:attribute type="PLSILType" name="validSafetyLevelType"/>
<xs:attribute type="SafetyLevelPL" name="eachSafetyLevelPL" use="optional"/>
        <xs:attribute type="SafetyLevelSIL" name="eachSafetyLevelSIL" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name="SafetyType"> <!-- Table 4.30 in ISO22166-201 -->
    <xs:restriction base="xs:string">
      <xs:enumeration value="ESTOP"/>
      <xs:enumeration value="PSTOP"/>
      <xs:enumeration value="LIMWS"/>
      <xs:enumeration value="SRSC"/>
      <xs:enumeration value="SRFC"/>
      <xs:enumeration value="HCOLA"/>
      <xs:enumeration value="STCON"/>
    </xs:restriction>
  </xs:simpleType>

<xs:complexType name="SafetyTypeSafeSecure">
    <xs:sequence>
<!-- overallSafetyLevelPL and overallSafetyLevelSIL, OverallvalidSafetyLevel is obtained from
these two types -->
      <xs:element type="OverallSafetyType" name="overall" maxOccurs="2"/>
      <xs:element type="SafetyFunction" name="inSafetyLevel" maxOccurs="unbounded"
minOccurs="0"/>
    </xs:sequence>
</xs:complexType>


  <xs:simpleType name="SecurityTypeList"> <!-- Table 4.35 in ISO22166-201 -->
    <xs:restriction base="xs:string">
      <xs:enumeration value="HU_IA"/>
      <xs:enumeration value="SD_IA"/>
      <xs:enumeration value="ACNT_MGT"/>
      <xs:enumeration value="ID_MGT"/>
      <xs:enumeration value="AUTH_MGT"/>
      <xs:enumeration value="WIRELEE_MGT"/>
      <xs:enumeration value="PW_AUTH"/>
      <xs:enumeration value="PK_CERT"/>
      <xs:enumeration value="STR_PK_AUTH"/>
      <xs:enumeration value="LOGIN_NO"/>
      <xs:enumeration value="ACC_UNTRUST_NET"/>
      <xs:enumeration value="AUTHORIZE"/>
      <xs:enumeration value="WIRELESS_USE"/>
      <xs:enumeration value="SESS_LOCK"/>
      <xs:enumeration value="SESS_TERM"/>
      <xs:enumeration value="SECC_CNTR"/>
      <xs:enumeration value="AUDT_EVT"/>
      <xs:enumeration value="TIMESTM"/>
      <xs:enumeration value="NON_REP"/>
      <xs:enumeration value="COMM_INTG"/>
      <xs:enumeration value="PROT_MALI_CODE"/>
      <xs:enumeration value="SECUR_VERIFY"/>
      <xs:enumeration value="SW_INTGT"/>
      <xs:enumeration value="INPUT_VALD"/>
      <xs:enumeration value="DET_OUT"/>
      <xs:enumeration value="ERR_HNDL"/>
      <xs:enumeration value="SESS_INTGT"/>
      <xs:enumeration value="INFO_CONFI"/>
      <xs:enumeration value="INFO_PERS"/>
      <xs:enumeration value="CRYTO"/>
      <xs:enumeration value="RSTIC_FLOW"/>
      <xs:enumeration value="DoS"/>
      <xs:enumeration value="RESOU_MGT"/>
      <xs:enumeration value="CNTR_RECOV_RECON"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="SecurityLevel"> <!-- Table 4.36 in ISO22166-201  -->
    <xs:restriction base="xs:string">
      <xs:enumeration value="0"/>
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

```
        <xs:enumeration value="1"/>
        <xs:enumeration value="2"/>
        <xs:enumeration value="3"/>
        <xs:enumeration value="4"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="SecurityType">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute type="SecurityTypeList" name="type" use="optional"/>
          <xs:attribute type="SecurityLevel" name="value" use="optional"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>

  <xs:complexType name="CyberSecurityType">
      <xs:sequence>
        <xs:element type="SecurityLevel" name="OverallCybSecurityLevel"/>
        <xs:element type="SecurityType" name="inCybSecurityLevel" maxOccurs="unbounded"
  minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>

  <xs:complexType name="SecurityTypeSafeSecure">
      <xs:sequence>
        <xs:element type="xs:string" name="PhysicalSecurity"/> <!-- This is not needed for SW
  module -->
        <xs:element type="CyberSecurityType" name="CyberSecurity"/>
      </xs:sequence>
  </xs:complexType>

    <xs:complexType name="SafeSecure"> <!-- this class is diveded into 3 parts -->
      <xs:sequence>
        <xs:element type="SafetyTypeSafeSecure" name="Safety"/>  <!-- collection of safety-
  related data -->
        <xs:element type="SecurityTypeSafeSecure" name="Security"/> <!-- collection of
  security-related data-->
        <xs:element type="NVList" name="additionalInfo" minOccurs="0"/>
      </xs:sequence>
    </xs:complexType>

  <xs:complexType name="ModelFileType">
      <xs:sequence>
        <xs:element type="xs:string" name="Item" maxOccurs="unbounded" minOccurs="0">
        </xs:element>
      </xs:sequence>
      <xs:attribute type="xs:string" name="type" use="optional"/>
    </xs:complexType>

   <xs:complexType name="ModelCaseType">
      <xs:sequence>
        <xs:element type="xs:string" name="simulator"/>
        <xs:element type="ModelFileType" name="mdf" maxOccurs="unbounded"/>
        <xs:element type="xs:string" name="libraries" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element type="ExeFormType" name="dynamicSW" maxOccurs="unbounded" minOccurs="0"/>
        <xs:element type="NVList" name="additionalInfo" minOccurs="0"/>
      </xs:sequence>
  </xs:complexType>

    <xs:complexType name="Modelling">
      <xs:sequence>
        <xs:element type="ModelCaseType" name="SimulationModel" maxOccurs="unbounded"
  minOccurs="0"/>
      </xs:sequence>
  </xs:complexType>

    <xs:complexType name="ExeFormType">
      <xs:sequence>
        <xs:element type="xs:string" name="exeFileURL"/>
        <xs:element type="xs:string" name="shellCmd" minOccurs="0" maxOccurs="unbounded"/>
       <xs:element type="PropertyType" name="properties" maxOccurs="unbounded" minOccurs="0"/>
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

```
      <xs:element type="NVList" name="additionalInfo" minOccurs="0"/>
     </xs:sequence>
</xs:complexType>`

 <xs:complexType name="ExecutableForm">
     <xs:sequence>
       <xs:element type="xs:string" name="libraryURL" maxOccurs="unbounded" minOccurs="0"/>
       <xs:element type="ExeFormType" name="exeForm" maxOccurs="unbounded"/>
     </xs:sequence>
   </xs:complexType>

   <xs:simpleType name="HWSWIMType">
     <xs:restriction base="xs:string">
       <xs:enumeration value="SIM"/>
       <xs:enumeration value="HIM"/>
       <xs:enumeration value="HWSWIM"/>
     </xs:restriction>
   </xs:simpleType>

   <xs:element name="Module" type="ModuleType"/>
   <xs:complexType name="ModuleType">
     <xs:sequence>
       <xs:element type="xs:string" name="moduleName"/>
       <xs:element type="xs:string" name="description" minOccurs="0"/>
       <xs:element type="xs:string" name="manufacturer"/>
       <xs:element type="xs:string" name="examples" minOccurs="0"/>
       <xs:element type="IDnType" name="idnType"/>
       <xs:element type="Properties" name="properties"/>
       <xs:element type="IOVariables" name="ioVariables"/>
       <xs:element type="Services" name="services"/>
       <xs:element type="Infrastructure" name="infra"/>
       <xs:element type="SafeSecure" name="safeSecure" minOccurs="0"/>
       <xs:element type="Modelling" name="modelling" minOccurs="0"/>
       <xs:element type="ExecutableForm" name="execForm"/>
     </xs:sequence>
     <xs:attribute type="HWSWIMType" name="type"/>
   </xs:complexType>
</xs:schema>
```

B.3 Information for Module and General Information of a module

Based on the XML schema for a CIM of a module, an example of information for the SIM of a module is provided as follows:

```
<?xml version="1.0" encoding="UTF-8" ?> <-- version = "1.1" OK -->
<!-- see detailed information in Annex B in ISO 22166-201 -->
<Module type ="SIM">
    <moduleName> string  </moduleName> <!-- see C.2 in ISO 22166-201 -->
    <description> string  </description>
    <manufacturer> string  </manufacturer>
    <examples> string  </examples>
    <idnType> <!-- see B.3 -->  </idnType>
    <properties> <!-- see B.4 -->  </properties>
    <ioVariables>  <!-- see B.4 --> </ioVariables>
    <services>   <!-- see B.5 -->   </services>
    <infra>      <!-- see B.6 -->   </infra>
    <safeSecure>   <!-- see B.7 --> </safeSecure>
  <modelling>    <!-- see B.8 --> </modelling>
  <execForm> <!-- see B.9 --> </execForm>
</Module>
```

XML elements for information of the class GenInfo shall use the same elements used for the common information model defined in ISO 22166-201:2024, Clause C.2.

B.4 Information for Module ID

XML elements for information of the class IDnType shall use the same elements used for the common information model defined in ISO 22166-201:2024, Annex C. The ID shall be assigned using Annex A. Since

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

the class IDnType for SIM is different from that for CIM, XML elements of the modified parts are provided in Table B.1. The module IDs listed in the “SWlist” element are used in the element “organization” of Table B.2.

NOTE The XML element “HWlist” is not used as information for an SIM.

Table B.1 — XML elements for module ID

Element Name	Description
idnType	A module ID, an information model version, lists of all modules used in a module are defined within this element. The list of modules is provided only for a composite module. This element can have an attribute “type” for providing the module type such as basic module or composite module. If its value is “Bas” or the attribute is not defined, the module is a kind of basic module. If its value is “Com”, the module is a kind of composite module.
moduleID	Used for manufacturer’s unique product reference number for a module (see Annex B). This element shall have two attributes, “mID” and “iID”, which are the module ID and the instance ID, respectively. Both mID and iID are of hexadecimal type.
informationModelVersion	Same as the description provided in ISO 22166-201:2024, Table C.3.
swAspects	Used for the set of software modules in the composite module. The moduleIDs of the composite module are listed. This element has one or more attributes “ModuleID”, which are the IDs of the modules constituting the module. The module ID consists of “mID” and “iID”.

Based on the XML schema for module ID, an example of information for module ID of a basic module is provided as follows:

```
<idnType type="Bas">   <!-- example of basic module -->
  <moduleID> <!-- provided by manufacturer and defined in Annex A-->
   <mID>8288d5ae4ee858e6b52d29916bd5dd9f800000ff0000ffff00000000941080</mID>
   <iID>00</iID>
  </moduleID>
  <informationModelVersion> 1.0 </InformationModelVersion>  <!-- string -->
</idnType>
```

An example of information for module ID of a composite module is provided as follows:

```
<idnType type="Com">            <!-- example of a composite module -->
  <moduleID>
   <mID>5766d5ae4ee85ffee52d29916bd5dd9fc00000ff0000ffff00000000941080</mID>
   <iID>00</iID>
  </moduleID>
  <informationModelVersion> 1.0 </informationModelVersion>  <!-- string -->
  <swAspects>
    <moduleID>
     <mID>8288d5ae4ee858e6b52d29916bd5dd9f800000ff0000ffff00000000941080</mID>
     <iID>00</iID>
    </moduleID>
    <moduleID>
     <mID>5566cd564ee858e6b52d24564bd5eeee800000ff000053ff00000000456060</mID>
     <iID>00</iID>
    </moduleID>
    <moduleID>
      <mID>ddcc8288d5ae4ee6b52c5c516bdaaaaf800000ff0000cd4300000000345070</mID>
      <iID>00</iID>
    </moduleID>
  </swAspects>
</idnType>
```

B.5 Information for classes Properties and IOVariables

XML elements for information of the class Properties shall use the same elements used for the common information model defined in ISO 22166-201:2024, Clause C.4. Additional XML elements for the class Properties of the software information model, shall be used as s provided in Table B.2. XML elements for information of the class IOVariables shall use the same elements used for the common information model defined in ISO 22166-201:2024, Clause C.4.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Table B.2 — Additional XML elements for class Property

Element Name	Description
properties	All related properties are defined within this element.
Property	For the SIM, the added attribute is "immutable". The attribute's value is "true" or "false". If this attribute is not provided, the attribute's value shall be considered as "false". See the description provided in ISO 22166-201:2024, Table C.4.
osType	This element is intended to identify the operating system and consists of three attributes, which are "type", "bit", and "version". The attribute "type" means the type of an operating system (OS). The attribute "bit" means the number of bit that the OS supports. The attribute "version" means the version of the OS.
exeType	This element is intended to identify one or more execution types of a software module. Its element is ExecutionType. A module can generate multiple instances (or processes or threads) with different execution-related properties defined in ExecutionType.
ExecutionType	This element is intended to identify the execution type of a software module and consists of five attributes: "opType", "priority", "hardRT", "timeConstraint" and "instanceType". The attribute "opType" is for the operation type of a software module and shall have one of following enumeration values: {PERIODIC, EVENTDRIVEN, NONRT}. NONRT means non real-time. The attribute "priority" is used to set the priority of a module's instance. The highest priority is 0 and the lowest priority is 255. The attribute "hardRT" is used to set whether the hard real-time property shall be satisfied. If so, the value is "true". Its default value is "false". The attribute "timeConstraint" is used to set the period for the operation type of PERIODIC and the deadline for the operation types of EVENTDRIVEN. The default value is 0, which means there is no setting value. The unit of "timeConstraint" is micro second. The attribute "instanceType" is related to instances of a module and shall have one of following enumeration values: {Singleton, MultitonStatic, MultitonCommutative}.
compiler	This element is intended to identify the compiler used for the execution file of a software module and consists of five attributes The attribute "osName" is the target operating system name. The attribute "verRangeOS" is the supported version range of the OS and has sub attributes "min" and "max". The attribute "compilerName" is the name of the target compiler suite. The attribute "verRangeCompiler" is the supported version range of the compiler and has sub attributes "min" and "max". The attribute "bitnCPUarch"is the target bit-size and CPU and has sub-attributes "bit" and "CPUarch".
Libraries	This element is intended to identify the libraries used for the execution of a software module and consists of two attributes. The attributes "name" and "version" mean the name and version of a library.
organization	This element is intended to describe the organization of a composite module and consists of four attributes, which are "owner", "dependency", "member" and "additionalInfo". The attribute "owner" is the owner of the module and has the module ID value of the owner. If a module is a composite type, the attribute "dependency" shall be set to one value of enumeration values: {OWNER, OWNED, OWNEROWNED, NONE}. The attribute "member" is a list of member modules that make up this module. The list consists of member module's ID values (mID and iID) and its dependency type. The attribute "additionalInfo" can have zero or more values utilizing the "nv" attribute, where each value consists of a "name" and "value" pair.
IOVariables	Same as the description provided in ISO 22166-201:2024, Table C.4.

Based on the XML schema for class Properties, an example incorporating additional information for the Properties class is provided below:

```
<properties>  <!-- properties -->
 <Property name="maxRatedCurrent" type="Float" unit="ampere" description = "maximum of rated
current for motor" value ="15" />   <!-- immutable = "FALSE" -->
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

```
  <Property name="maxRatedVoltage" type="Float" unit="volt" description = "maximum of rated
voltage for motor" >
    <value> 5 </value>
    <immutable> true </immutable>   <!-- configurable property -->
</Property>   <!--  example of single property -->
<osType type="Ubuntu" bit="64" version="20.04" />   <!-- see Table 5 -->
  <exeType>  <!-- list of Execution Types -->
      <Item>
       <opType>PERIODIC</opType>
       <priority>1</priority>
       <hardRT>true</hardRT>
       <timeConstraint> 10000 </timeConstraint>    <!-- unit: micro seconds -->
       <instanceType> MultitonComm </instanceType>
      </Item>
      <Item>
       <opType>PERIODIC</opType>
       <priority>5</priority>
       <hardRT>true</hardRT>
       <timeConstraint> 100000 </timeConstraint>   <!-- unit: micro seconds -->
       <instanceType> MultitonComm </instanceType>
      </Item>
  </exeType>
  <compiler>
      <osName> Ubuntu </osName>
      <verRangeOS min="18.04" max="20.04" />
      <compilerName> GCC </compilerName>
      <verRangeCompiler min="14.0" max="Higher" />
      <bitnCPUarch bit="32" CPUarch="armv7hf"/>
  </compiler>
  <Libraries>
   <Item name="OpenCV" version="4.5.3" />
   <Item name="Boost" version="1.76.0" />
  </Libraries>
  <organization>
   <owner>
  <mID>5766d5ae4ee85ffee52d29916bd5dd9fc00000ff0000ffff00000000941080</ mID>
     <iID>00</iID>
</owner>
   <dependency>  OWNER  </dependency>   <!-- If this module is the main of a composite module
-->
   <member>
     <moduleID>
   <mID>5566cd564ee858e6b52d24564bd5eeee800000ff000053ff00000000456060</ mID>
     <iID>00</iID>
     </moduleID>
     <moduleID>
  <mID>ddcc8288d5ae4ee6b52c5c516bdaaaaf800000ff0000cd4300000000345070</ mID>
     <iID>00</iID>
     </moduleID>
   </member>
  </organization>
</properties>
```

B.6 Information for Services of a Module

XML elements for information of class Services shall use the same elements used for the common information model defined in ISO 22166-201:2024, Clause C.5. An additional XML element for class Services of the software information model should be used as provided in Table B.3.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Table B.3 — Additional XML elements for class Service

Element Name	Description
services	All information for Services is defined within this element. This element describes the information about the Services (or Functions/Capabilities) of a module.
ServiceIDL	IDL-related service information items are listed. The provided Service is described in IDL and the element value is the path of the related IDL file.
ServiceXML	XML-related service information items are listed. The provided Service is described in XML and the element values are provided in other XML elements.
ID	Used to identify the interface that a module uses or provides.
pvTYpe	Used to classify the interface type of a module, whose value is either "Physical" or "Virtual" If the module provides a physical interface such as a mechanical interface, the value is "Physical". If the module provides a software interface such as an application programming interface), the value is "Virtual".
moType	Used to declare that the provided interface is mandatory or optional.
Properties	Used to define properties that a module has to use (or set) when the module utilizes the provided interface The element has two attributes, "Name" and "Value", which are the name and the value of the property that the module has.
methodList	Used to define methods that the module has to utilize. One or more methods are listed in the element.
Method	Used to define one individual method in the element "methodLists".
methodName	Used to identify the provided method.
argType	Used to define the list of arguments. An argument consists of four attributes: "type", "valueName", "inout" and "order". "type" represents the data type of the argument, which is defined in ISO 22166-201:2024, Table C.5. "valueName" means the argument name of the method, and "inout" provides the direction of the argument and is input, output or both. The attribute or element "order" indicates the sequence of the arguments. However, if there is only one argument, this attribute cannot be utilized. If the "order" attribute is not provided, the order of arguments is assumed to be the order in which they appear in the XML.
retType	Used to define the data type that the method returns
moduleID	This attribute is exclusively used for composite modules and represents the module ID, where this method is utilized or defined.
additionalInfo	Additional information for variables. This element can have zero or more values utilizing the "nv" attribute, where each value consists of a "name" and "value" pair.

The IDL format is provided as follows:

```
Module CIMServicePackage {
 interface CIMService;    // abstract declaration
 struct NameValue
 {
  string name;
  any value;    // "any" means any data type
 };
 struct ModuleID
 {
  Octet[31] mID;  // module ID, 31 Bytes, 62 hexadecimal codes
  Octet     iID;  // istance ID, 1 byte
 };

// Mandatory/Optional enum type
enum MOType
 {
        MANDATORY,
        OPTIONAL
};
struct ArgSpec {
      string type;            // argument type
      string valueName;      // argument name
      InOut inout;           // enumeration {IN, OUT, INOUT}
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

```
      NVList additionalInfo;  // additional information
} ;
typedef sequence<ArgSpec> ArgSpecList;  //sequence means a list or vector of
// given data
struct ServiceMethod
{
     string methodName;
      ArgSpecList argType;    // ordered
      string retType;
      MOType moType;        // enumeration {"MANDATORY", "OPTIONAL"}
      ReqProvType reqProvType; // Provided or required method.
NVList additionalInfo;         // additional information
};
typedef sequence<NameValue> NVList;
typedef sequence<ServiceMethod> MethodList;
enumeration PhysicalVirtual
{
        PHYSICAL,
        VIRTUAL
};
// ServiceProfile
struct ServiceProfile
{
string id;
PhysicalVirtual pvType; // Enumeration PHYSICAL, VIRTUAL
MethodList methodList; //mandatory/optional method defined in CIMService
MOTypr moType;       // MOType: {MANDATORY, OPTIONAL}.
ModuleID moduleID;       // module's ID
NVList additionalInfo;    //  additional information
};
interface CIMService
{
  // define here for prototypes of methods for CIM Service
  // format: <type_spec | void> <method_identifier> ( <parameter decls> )
  // <parameter decls> ::= <para_dcl>
  // <param_dcl> ::= <"in"|"out"|"inout"> <type_spec> <parameter_name>
  // example:
  //   uint8 initialize(in int16 val1, in float64 val2);
   //   boolean checkElecConnectivity(string connType, string keying,
  //                                string busProtocol)
  //   boolean linkConnectivity(array origin, float mass, array inertia,
  // sring shape, array size,array axis,string connection,array collision)
 };
};
```

Based on the XML schema for class Services, an example including additional information of class Services is provided as follows:

```
<services>
  <ServiceIDL>./IDL/AppLayer/sdo.idl</ServiceIDL>
  <ServiceIDL>./IDL/AppLayer/sync.idl</ServiceIDL>
  <ServiceIDL>./IDL/DeviceDriver/CAN.idl</ServiceIDL>
  <ServiceXML>
    <ID> NMT </ID>ID>
    <pvType> Virtual </pvType>
    <moType> MANDATORY </moType>
    <methodList>
      <Method>
         <methodName value = "NMT_SearchNode" />
         <argType type="IOctet" valueName="iNodeID" inout="IN"/> <!--Order isn't used-->
         <retType value="bool" />
         <moType value = "MANDATORY" />
         <reqProvType value="REQUIRED" />
         <moduleID>
           <mID> hexadecimal ID </mID>
           <iID> 00 </iID>
         </moduleID>
         <additionalInfo>
               <Name> initialValue </Name>
               <Value> 0 </Value>
         </additionalInfo>
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

```
      </Method>
      <Method>
         <methodName value = "NMT_AddNode" />
         <argType order="1">
            <type> IOctet </type>
            <valueName> iNodeID </valueName>
            <inout> IN </inout>
            <additionalInfo>
               <nv>
                  <Name> minValue </Name>
                  <Value> 20 </Value>
               </nv>
            </additionalInfo>
         </argType>
         <argType order="2" type="IOctet" valueName="iType" inout="IN"/>
         <argType order="3" type="int" valueName="iLifeTime" inout="IN"/>
         <retType value="bool" />
         <moType value = "MANDATORY" />
         <reqProvType value="REQUIRED" />
      </Method>
      <Method>
         <methodName value = "NMT_GetNodeInfo" />
         <argType type="IOctet" valueName="iNodeID" inout="IN" />
         <argType type="pointer" inDataType="int" valueName="iType" inout="OUT"/>
         <argType type="pointer" inDataType="int" valueName="pTime" inout="OUT" />
         <argType type="pointer" inDataType="Octet" valueName="pLifeTime" inout="OUT" />
         <retType value="bool" />
         <moType value = "MANDATORY" />
         <reqProvType value="REQUIRED" />
      </Method>
      <additionalInfo>
         <nv>
<Name> InvokeFirstService </Name>
          <Value> NMT_SearchNode </Value>
         </nv>
      </additionalInfo>
    </methodList>
</ServiceXML>
<ServiceXML>
   <ID> RS485Driver </ID>
   <pvType value="Virtual" />
   <moType> OPTIONAL </moType>
    <additionalInfo>
       <nv>
          <Name> Baud </Name>
          <Value> 9600 </Value>
      </nv>
      <nv>
          <Name> STOPBit </Name>
          <Value> No </Value>
      </nv>
   </additionalInfo>
   <methodList>
      <Method>
         <methodName value = "initialize" />
         <argType order="1" type="LongLong" valueName="val1" inout="IN" />
         <argType order="2" type="Float" valueName="val2" inout="IN" />
         <retType value="Octet" />
         <moType value = "MANDATORY" />
         <reqProvType value="PROVIDED" />
      </Method>
      <Method>
         <methodName value = "initialize" />
         <argType order="1" type="LongLong" valueName="val1" inout="IN" />
         <argType order="2" type="Float" valueName="val2" inout="IN" />
         <argType order="3" type="Float" valueName="val3" inout="IN" />
         <retType value="Octet" />
         <moType value = "OPTIONAL" />
         <reqProvType value="PROVIDED" />
      </Method>
      <Method>
         <methodName value = "finalize" />
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

```
            <argType type="Long" valueName="val1" inout="IN" />
            <argType type="Float" valueName="val2" inout="IN" />
            <argType type="Long" valueName="val3" inout="IN" />
            <retType value="Octet" />
            <moType value = "MANDATORY" />
            <reqProvType value="PROVIDED" />
         </Method>
         <Method>
            <methodName value = "read" />
            <argType order="1" type="int" valueName="fd" inout="IN" />
            <argType order="2" type="pointer" inDataType ="Octet" valueName="val2" inout="INOUT"
/>
            <argType order="3" type="UShort" valueName="nbytes" inout="IN"/>
            <moType value = "MANDATORY" />
            <reqProvType value="PROVIDED" />
            <additionalInfo>
                 <nv>
                     <Name> minValue </Name>
                     <Value> 0 </Value>
                 </nv>
            </additionalInfo>
         </Method>
      </methodList>
</ServiceXML>
</services>
```

B.7 Information for class Infrastructure

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Some XML elements for information of class Infrastructure shall use the same elements used for the common information model defined in ISO 22166-201:2024, Clause C.6. XML elements for additional information of class Infrastructure, which is added for the SIM, shall be used as provided in Table B.4.

Table B.4 — XML elements for class Infrastructure

Element Name	Description
infra	All properties for class Infrastructure are defined within this element. This element describes the information about the infrastructure of a module.
Database	Element used to define information for the database utilized by one or more modules.
Middleware	Element used to define information for the middleware utilized by one or more modules.
InfraType	This element has two attributes: “Name” and “Version”. These attributes are explained in this table.
comms	Element used to define information for the list of communication protocols used commonly for modules. Two or more communication protocols can be provided, but application layer protocols(“mostTopeProcol”) and the corresponding underlying protocol(“underlyingProtocol”) are defined together.
Communication	Element used to define one communication protocol within the element “comms”. This element has two attributes: “mostTopeProcol” and “underlyingProtocol”.
mostTopProtocol	This element is used to provide information about application protocols or upper-layer protocols over the application protocol. The application protocol refers to the seventh layer protocol in the OSI model. This element has an attribute called "InfraType," which specifies the highest-level protocols used.
underlyingProtocol	This element is used to provide information about communication protocols that support the "mostTopProtocol" Element. This element has one or more "Databus" attributes.
Item	This element is intended to be used when entering the values of a list or an (unordered) array of objects.
Name	Names of the database, communication protocol and middleware.
Version	Range of valid versions. This element consists of two attributes, “min” and “max”, which are the minimum version number and the maximum version number (String types), within which the corresponding Infra type is valid. If the corresponding field is NULL, it means no limit. That is, if a value of the attribute “min” is NULL, all the versions lower than the value of the attribute “max” are valid. If all versions are valid, this element can be omitted.

Table B.4 *(continued)*

Element Name	Description
Databus	Same as the description provided in ISO 22166-201:2024, Table C.7.
Speed	Same as the description provided in ISO 22166-201:2024, Table C.7. These elements are used to verify whether the HW-SW module used for communication matches the communication information set for the application.
ConnectorType	
TypePhyMac	
TypeNetTrans	Same as the description provided in ISO 22166-201:2024, Table C.7.
TypeApp	Same as the description provided in ISO 22166-201:2024, Table C.7.
additionalInfo	Additional information for variables. This element can have zero or more values utilizing the "nv" attribute, where each value consists of a "name" and "value" pair.

Based on the XML schema for class Infrastructure, an example including additional information of class Infrastructure is provided as follows:

```
<infra>
 <Database>   <!-- Database type  -->
   <InfraType>
     <Name> MySQL </Name>
     <Version min="5.6" max="" />  <!-- “min” or “max" cannot be used -->
   </InfraType>
 </Database>
 <Middleware>   <!-- middleware type -->
   <InfraType>
     <Name> ROS 2 </Name>
     <Version min="Humble" max="Humble" />
   </InfraType>
   <InfraType>
     <Name> OpenRTM </Name>
     <Version min="1.2.0" max="1.2.2" />
   </InfraType>
 </Middleware>
 <comms>
  <Communication>  <!-- Communication type 1-->
   <mostTopProtocol>    <!-- application protocol -->
    <InfraType>
        <Name> FTP </Name>    <!-- all versions of FTP are valid. Hence no version is defined.
-->
    </InfraType>
    <InfraType>
        <Name> HTTP </Name>    <!-- all versions of HTTP are valid. -->
    </InfraType>
    <InfraType>
        <Name> PrivateProtocol </Name>  <!-- all versions of Private -->
            <Version min="1.0" max="1.0" />
    </InfraType>
   </mostTopProtocol> <!-- end of the mostTopProtocol(app) protocol -->
<underlyingProtocol>    <!-- transport,network,datalink,physical -->
  <Databus>
       <ConnectorType> DB9 </ConnectorType>   <!-- D-Sub 9-->
       <TypePhyMac> CAN </TypePhyMac>
       <TypeNetTrans> CANopen </TypeNetTrans>
       <TypeApp> <!-- list for app. Layer of CANopen -->
         <Item> OBD </Item>
         <Item> NMT </Item>
         <Item> SDO </Item>
         <Item> PDO </Item>
         <Item> SYNC </Item>
       </TypeApp>
       <Speed value = "1000" unit="kbps" />  <!-- 1 Mbps, unit Kbps -->
       <additionalInfo>
            <nv>
               <Name> Maximum speed of EtherCAT </Name>
               <Value> 100000 </Value>
            </nv>
          </additionalInfo>
   </Databus>
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

```
 </underlyingProtocol>
</Communication>
<Communication>  <!-- Communication type 2-->
   <mostTopProtocol>    <!-- application protocol -->
    <InfraType>
        <Name> FTP_based_APP </Name> <!-- all versions are valid. Hence no version is defined.
-->
    </InfraType>
    <InfraType>
        <Name> HTTP_based_APP </Name>    <!-- all versions are valid. -->
    </InfraType>
    <InfraType>
        <Name> PrivateProtocol </Name>  <!-- all versions of Private -->
<Version min="1.0" max="1.0" />
    </InfraType>
   </mostTopProtocol> <!-- end of the mostTopProtocol(app) protocol -->
<underlyingProtocol>    <!-- transport,network,datalink,physical -->
<Databus>
       <ConnectorType> RJ45 </ConnectorType>   <!-- D-Sub 9-->
       <TypePhyMac> Ethernet </TypePhyMac>
       <TypeNetTrans> TCP/IP </TypeNetTrans>
       <TypeApp> <!-- list for app. Layer of CANopen -->
         <Item> ISO_protocol_1 </Item>
         <Item> FTP </Item>
         <Item> HTTP </Item>
       </TypeApp>
       <Speed value = "100000" unit="kbps" />  <!-- 100 Mbps, unit Kbps -->
       <additionalInfo>
            <nv>
              <Name> Maximum speed of EtherCAT </Name>
             <Value> 100000 </Value>
            </nv>
        </additionalInfo>
     </Databus>
   </underlyingProtocol>
</Communication>
 </comms>
</infra>
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

B.8 Information for Safety and Security

Some XML elements for information of class Infrastructure shall use the same elements used for the common information model, which are listed in ISO 22166-201:2024, Clause C.7. XML elements for additional information of class Infrastructure, which is added for the software information model, shall be used as provided in Table B.5.

Table B.5 — XML elements for class SafeSecure

Element Name	Description
SafeSecure	Same as the description provided in ISO 22166-201:2024, Table C.8.
Safety	Same as the description provided in ISO 22166-201:2024, Table C.8.
Overall	Same as the description provided in ISO 22166-201:2024, Table C.8.
inSafetyLevel	Used for providing the performance level of specified individual safety function. Using the attribute “safetyFunctionType”, the individual safety function is specified and its values are shown in the column “Enumeration type” in defined in ISO 2266-201:2024, Table 4.30. The attribute “validSafetyLevelType” is either “PL” or “SIL”, which denote the performance level and the safety integrity level, respectively. According to the value of the attribute “validSafetyLevelType”, only one of the attributes eachSafetyLevelPL and eachSafetyLevelSIL shall be given.
Security	Same as the description provided in ISO 22166-201:2024, Table C.8.
CyberSecurity	Same as the description provided in ISO 22166-201:2024, Table C.8.

Table B.5 *(continued)*

Element Name	Description
OverallCybSecurityLevel	Used to define the security level of cyber security function. Values {0, 1, 2, 3, 4} (see ISO 22166-201:2024, Table 4.36).
inCybSecurityLevel	Used for providing the security level of specified individual security function. Using the attribute “type”, the individual security function is specified and its values are shown in ISO 22166-201:2024, Table 4.35. The attribute “value” is given in ISO 22166-201:2024, Table 4.36.
additionalInfo	Additional information for Safety and Security. This element can have zero or more values utilizing the "nv" attribute, where each value consists of a "name" and "value" pair.

Based on the XML schema for class SafeSecure, an example of information for class SafeSecure is provided as follows:

```
<safeSecure>
  <Safety>
    <overall mode="PL">d</overall> <!--choose one PL from a,b,c,d,e, and none.-->
    <overall mode="SIL">3</overall> <!-- choose one SIL from 0, 1, 2, and 3.  -->
    <!-- individual safety measure. see Table 4.5 -->
    <inSafetyLevel safetyFunctionType = "ESTOP" validSafetyLevelType ="PL" eachSafetyLevelPL =
"d"/>
    <inSafetyLevel safetyFunctionType = "ESTOP" validSafetyLevelType ="SIL" eachSafetyLevelSIL
="3" />
    <inSafetyLevel safetyFunctionType = "SRSC" validSafetyLevelType ="SIL" eachSafetyLevelSIL
= "2" />
  </Safety>
  <Security>
   <CyberSecurity>   <!-- list security functions provided by module -->
      <OverallCybSecurityLevel>2</OverallCybSecurityLevel>  <!-- overall security level of a
module -->
      <inCybSecurityLevel type = "HU_IA">3</SecType> <!-- individual security fct. see
Table 4.10 -->
      <inCybSecurityLevel type = "ACNT_MGT" value = "2"/> <!--see Table 4.10 -->
    </CyberSecurity>
  </Security>
  <additionalInfo>
    <nv>
      <Name> Minimum overall PL </Name>
      <Value> 2 </Value>
    </nv>
    <nv>
      <Name> Minimum overall security </Name>
      <Value> 1 </Value>
    </nv>
  </additionalInfo>
</safeSecure>
```

B.9 Information for Modelling

Some XML elements for information of the Modelling class shall use the same elements used for the common information model defined in ISO 22166-201:2024, Clause C.8. The XML element name used in CIM is modified to match the attribute name used in the Modelling class. The modified XML elements for SIM and the XML elements for CIM are provided in Table B.6. XML elements for SIM in Table B.6 shall be utilized for information of the Modelling class.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Table B.6 — XML Elements for class Modelling

Element Name for SIM	Element Name for CIM (ISO 22166-201)	Description
modelling	Modelling	Same as the description provided in ISO 22166-201:2024, Table C.9.
simulationModel	SimulationModel	
simulator	Simulator	
mdf	ModelFile	
dynamicSW	DynamicSW	
additionalInfo	additionalInfo	
Item	Item	This element is intended to be used when entering the values of a list or an (unordered) array of objects.

Based on the XML schema for class Modelling, an example of information for class Modelling is provided as follows:

```
<modelling>
  <simulationModel>  <!-- the 1-st simulation model to be used -->
     <simulator> Gazebo </simulator>   <!-- Simulation programs. optional -->
<mdf type="URDF">  <!-- model description file type and its file path for model of module -->
      <Item>../../modelDescrtionFileName1</Item> <!-- the 1st path of model description file
-->
      <Item>../../modelDescrtionFilePath2</Item> <!-- the 2nd path of model description file
-->
    </mdf>
    <mdf type="STL">  <!-- 3D graphic model type and its file path for model of module -->
      <Item> ../../module3DGraphicFileName1 </Item> <!--the 1st path of 3Dgraphic file model
file -->
      <Item> ../../module3DGraphicFilePath2 </Item> <!--the 2nd path of 3Dgraphic file model
file -->
    </mdf>
    <dynamicSW>
       <exeFileURL> ../../sim.exe </ExeFileURL>
       <ShellCmd> ../../sim.exe -r </ShellCmd>  <!--example of shell command -->
       <properties>
         <Property name="Shell cmd Arg1" type="IOctet" immutable="true" unit="none"
description = "verbose" value ="0" />
         <Property name="Shell cmd Arg2" type="Octet" immutable="false" unit="none"
description = "Warning level"  value ="2" />
       </properties>
       <additionalInfo>
         <nv>
           <Name> kinematics model(Axis) </Name>
           <Value> 6 Axis </Value>
         </nv>
         <nv>
           <Name> additional info </Name>
           <Value> added </Value>
         </nv>
       </additionalInfo>
    </dynamicSW>
  </simulationModel>
</modelling>
```

B.10 Information for ExecutableForm

XML elements for information of class ExecutableForm shall use the same elements used for the common information model defined in ISO 22166-201:2024, Annex C.

Some XML elements for information of the ExecutableForm class shall use the same elements used for the CIM defined in ISO 22166-201:2024, Clause C.9. The XML element name used in CIM is modified to match the attribute name used in the ExecutableForm class. The modified XML elements for SIM and the XML elements for CIM are provided in Table B.7. XML elements for SIM in Table B.7 shall be utilized for information of the ExecutableForm class.

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Table B.7 — XML Elements for class ExecutableForm

Element Name	Description
exeForm	All executable form related information is defined within this element. This element provides details about an executable form that is to be executed for the operation of the module. It has three attributes: ExeFileURL, SheelCmd, and Properties.

Based on the XML schema for the ExecutableForm class, an example of information for the ExecutableForm class is provided as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<execForm>
  <libraryURL>../lib/windows/modulelib.dll</libraryURL>
  <libraryURL>www.module/org/windows/lib/modulelib.dll</libraryURL>
  <exeForm>
    <exeFileURL> www.module.org/windows/exe/module1.exe</ExeFileURL>
    <ShellCmd>./module1.exe -r </ShellCmd> <!--example of shell command -->
    <Properties>
      <Property name="Shell comd Arg1" type="integer" immutable="true" unit="none" description
= "verbose"  value ="0" />
      <Property name="Shell comd Arg2" type="uint8" immutable="false" unit="none" description
= "Warning level"  value ="2" />
    </Properties>
    <additionalInfo>
      <nv>
            <Name> update date </Name>
            <Value> Jun. 24, 2023 </Value>
      </nv>
      <nv>
            <Name> additional info </Name>
            <Value> added </Value>
      </nv>
     </additionalInfo>
    <exeForm>
  </execForm>
```

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Annex C
(informative)

Examples for application of the software information model

An example of an information model for a simultaneous localization and mapping (SLAM) module as a composite module is illustrated in Figure C.1 using multiple sensing modules, examples of which are Lidar-, camera-, and infrared camera-sensing software modules. For a localization module, at least one or more sensing software modules are utilized. When using a module, the information model provided by the module is used to determine whether to use it. Since the module's interfaces are provided in the module's information model, the modules can exchange data between them using interfaces.

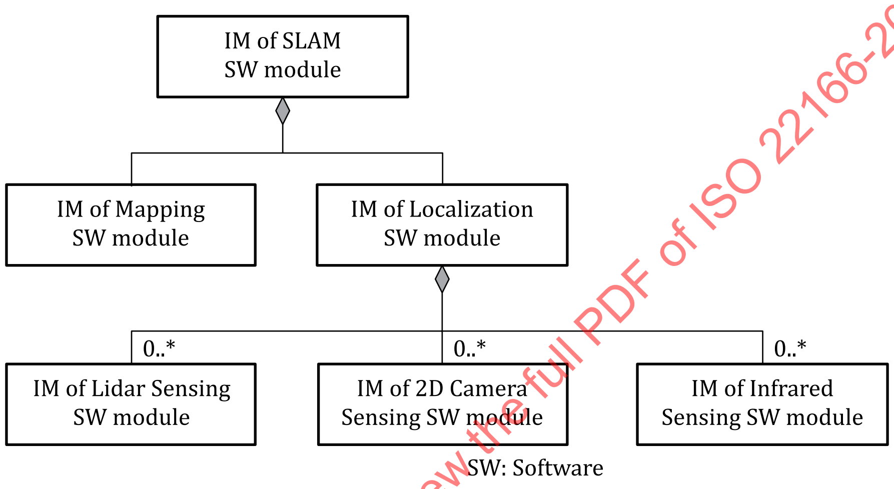
IM of SLAM SW module
IM of Mapping SW module
IM of Localization SW module
0..*
0..*
0..*
IM of Lidar Sensing SW module
IM of 2D Camera Sensing SW module
IM of Infrared Sensing SW module
SW: Software

Figure C.1 — Example of information models for a SLAM software module

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Annex D
(informative)

How to use software information models

This annex describes how to use an information model of a software module or a software information model and includes information usage among various stakeholders such as software module makers, hardware-software module makers, and system integrators. A software information model provides various information which can help the designer/developer/integrator design a new software module or a composite module, where the composite module means the composite software module and the composite module with the software modules and the hardware-software module. Some of this information can be utilized at an early design stage, some at the design and development stage, and some at the operation stage, all of which are provided in ISO 22166-201:2024, Table D.3.

Figure D.1 shows the relationships between module information and its stakeholders, where the module information is described based on the software information model.

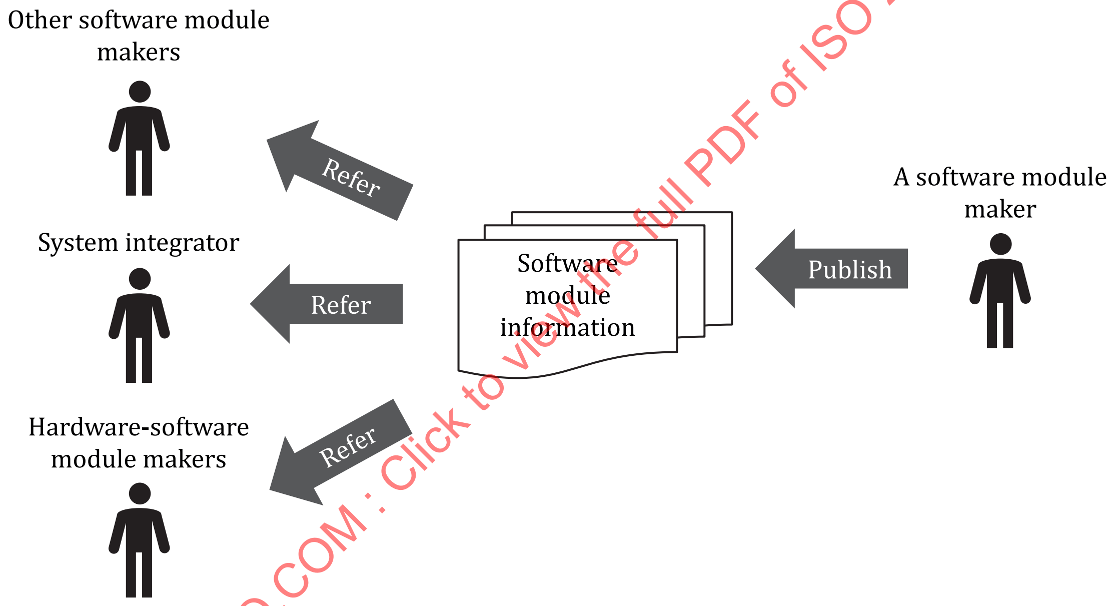
Other software module makers
Refer
System integrator
Refer
Hardware-software module makers
Refer
Software module information
Publish
A software module maker

Figure D.1 — Relationships between the software module information and its stakeholders

STANDARDSISO.COM : Click to view the full PDF of ISO 22166-202:2025

Software modules that were created using existing software module information by software module makers, system integrators and hardware-software makers can be also utilized to create other software modules.

Each element listed in Table 1 of this document is utilized within the lifecycle-related stages depicted in Table D.1. This lifecycle encompasses design, development, operating/runtime, maintenance and demolishment/disposal stages, as depicted in Figure D.2. The scope of the information model for software modules can be split into three main categories, which are further subdivided as follows, and are illustrated in Table D.1:

1) Information Type 1: Information concerning the runtime stage of the module, including configuration data. This information allows the creation of generic (vendor-independent) system monitoring and automatic system reconfiguration.

2) Information Type 2: Information concerning the design/development stage of the module. This information enables the (human) system designer/system integrator to determine if and how two modules can be combined into one system. The (automated) software engineering tools is expected to