International Standard

ISO 22166-202

First edition
2025-03

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

Reference number
ISO 22166-202:2025(en)

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

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.

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.

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]

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.]

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. ROS[1)],[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.

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).

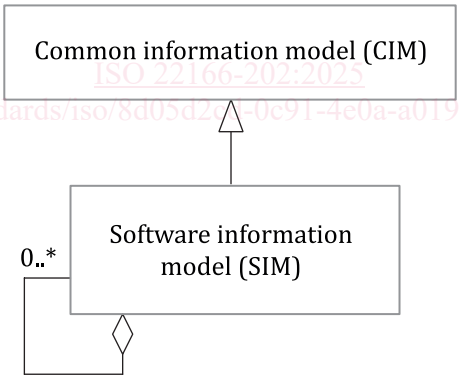

Figure 1 — Relationship between common information model and software information model

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.

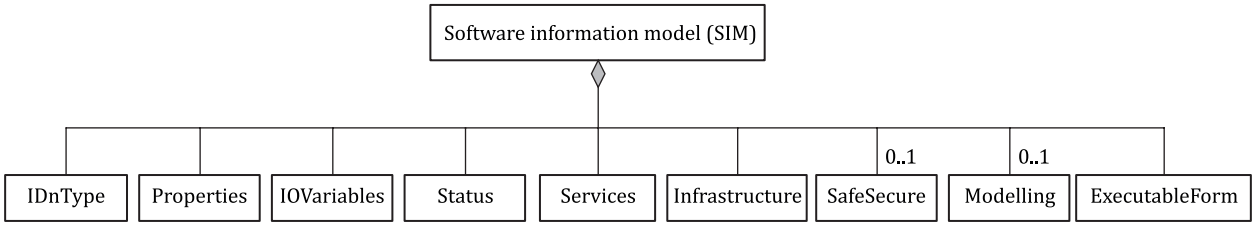

Figure 2 — Class of the software information model

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.

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

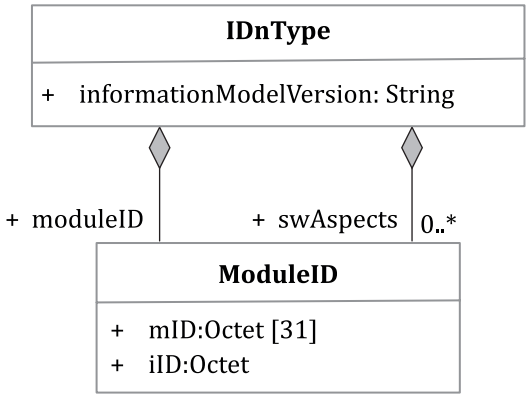

Figure 4 — Relationship between classes for class IDnType

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.

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.