Note: Changes in class names and attribute names by Issue 13441 are not marked here as there will be too many marks.

June 2009~~ly 2008~~

Robotic Localization Service Specification

FTF Beta 2~~1~~

OMG Document Number: dtc/2008-07-01
Standard document URL: http://www.omg.org/spec/RLS/1.0/PDF
Associated File(s)*: http://www.omg.org/spec/RLS/20080501

http://www.omg.org/spec/RLS/20080502

- original files: robotics/2008-05-03 (C++ header file), robotics/2008-05-04 (xmi)

This OMG document replaces the submission document (robotics/2008-05-01, Alpha 1). It is an OMG Adopted Beta Specification and is currently in the finalization phase. Comments on the content of this document are welcome, and should be directed to issues@omg.org by February 23, 2009.

You may view the pending issues for this specification from the OMG revision issues web page *http://www.omg.org/issues/*.

The FTF Recommendation and Report for this specification will be published on July 2 , 2009. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page http://www.omg.org, under Documents, Report a Bug/Issue (http://www.omg.org/technology/agreement.htm).

Table of Contents

List of Figures

List of Tables

Preface

About the Object Management Group

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable, and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies, and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at http://www.omg.org/.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. A Specifications Catalog is available from the OMG website at:
http://www.omg.org/technology/documents/spec_catalog.htm

Specifications within the Catalog are organized by the following categories:

OMG Modeling Specifications

UML

MOF

XMI

CWM

Profile specifications

OMG Middleware Specifications

CORBA/IIOP

IDL/Language Mappings

Specialized CORBA specifications

CORBA Component Model (CCM)

Platform Specific Model and Interface Specifications

CORBAservices

CORBAfacilities

OMG Domain specifications

OMG Embedded Intelligence specifications

OMG Security specifications.

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
140 Kendrick Street
Building A, Suite 300
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult *http://www.iso.org*

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.

`Courier - 10 pt. Bold:` Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

NOTE: Terms that appear in italics are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

1. Scope

13184 This specification defines a robotic localization (RoLo) service that can handle data and usages specific to use in robotics. It includes a platform independent model (PIM) as well as a mapping of this PIM to a platform specific model (PSM) defined by C++. In addition, two informative annex parts are provided for the filter condition functionality. The first defines a PSM by XML and the another shows naming rules~~localization service that can handle data and usages specific to use in robotics. It includes a platform-independent model (PIM) as well as a mapping of this PIM to platform-specific models (PSM) defined by C++~~.

2. Conformance

Any implementation or product claiming conformance to this specification shall support the
13426 following conditions:~~platform independent model described in chapter 6.~~

- Implementations shall provide the interfaces described in section 7.5 Interface Package.
- Implementations shall provide their ability descriptors and the necessary attribute definitions described in section 7.5 Interface Package.
- Data treated by implementations shall follow the data structure described in 7.3 Architecture Package and the data formats described in 7.4 Data Format Package. This does not mean that modules shall be able to treat every structure or formats described herein. However, every module shall support at least one of the common data formats and the relevant data structure.
- Implementations shall support the return codes described in section 7.2.

13427 3. References

3.1 Normative References

4. ~~Normative Reference~~

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

[ISO19103] International Organization for Standardization, Geographic information —

Conceptual schema language, 2005

[ISO19107] International Organization for Standardization, Geographic information – Spatial schema, 2003

[ISO19111] International Organization for Standardization, Geographic information – Spatial referencing by coordinates, 2007

[ISO19115] International Organization for Standardization, Geographic information – Metadata, 2003

[PER] International Telecommunication Union Telecommunication Standardization Sector, Specification of Packed Encoding Rules (PER), ITU-T Rec. X.691 (2002) / ISO/IEC 8825-2:2002

[UML] Object Management Group, OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.2, OMG document number formal/2009-02-02, 2009

13427

4.1 Non-Normative References

[ISO/DIS19142] International Organization for Standardization, Geographic information – Web Feature Service, DIS, 2009

[ISO/DIS19143] International Organization for Standardization, Geographic information – Filter Encoding, DIS, 2009

[Wikipedia] Wikipedia, the free encyclopedia, http://www.wikipedia.org/

5. Terms and Definition

13428

- *Cartesian coordinate system:* Coordinate system which gives the position of points relative to n mutually perpendicular axes [ISO19111]. Note that in this specification, in contrast to [ISO19111], there is no limitation to the number of dimensions~~No terms are defined in this document~~.
- *Coordinate reference system (CRS):* Coordinate system which is related to the real world by a datum [ISO19111].
- *Coordinate system (CS):* Set of mathematical rules for specifying how coordinates are to be assigned to points [ISO19111].
- Coordinate value: N-tuple of scalars assigned with respect to a coordinate system. In this specification, every coordinate value shall be associated with a single coordinate reference system. Note that, there exists no uncertainty with a coordinate value; error through the measurement process shall be represented by 'error' values elsewhere.
- *Covariance:* Covariance is a measure of how much two variables change together (variance is a special case of the covariance when the two variables are identical).If two

13428 variables tend to vary together (that is, when one of them is above its expected value, then the other variable tends to be above its expected value too), then the covariance between the two variables will be positive. On the other hand, if one of them tends to be above its expected value when the other variable is below its expected value, then the covariance between the two variables will be negative [Wikipedia].

- *Datum:* Parameter or set of parameters that define the position of the origin, the scale, and the orientation of a coordinate reference system [ISO19111]. More specifically, a datum is a mathematical system that defines the mapping from a space defined by coordinate system to a certain phenomenon space of interest, mostly in the real world.
- *Geodetic coordinate system:* Coordinate system in which position is specified by geodetic latitude, geodetic longitude and (in the three-dimensional case) ellipsoidal height, associated with one or more geographic coordinate reference systems [ISO19111].
- *Geographic(al) Information System (GIS):* Information system for storing, analyzing, managing or displaying various data in a way associated with location data. The location data used in GIS is in most cases 2 or 3 dimensional position on the earth.
- *Kalman filter:* Kalman filter is an efficient recursive filter that estimates the state of a linear dynamic system from a series of noisy measurements. It is used in a wide range of engineering applications from radar to computer vision, and is an important topic in control theory and control systems engineering. Together with the linear-quadratic regulator (LQR), the Kalman filter solves the linear-quadratic-Gaussian control problem (LQG). The Kalman filter, the linear-quadratic regulator and the linear-quadratic-Gaussian controller are solutions to what probably are the most fundamental problems in control theory [Wikipedia].No terms are defined in this document.
- *Localization*: Action of locating some physical entities through analysis of sensing data. The word “locate” here may include not only measuring the position in the spatio-temporal space but also may include additional information such as identity, heading orientation or pose information of the target entity, measurement error estimation or measurement time.
- *Normal distribution:* A continuous probability distribution described by the following probability density function:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

A normal distribution is also called a Gaussian distribution [Wikipedia].

- *Particle, particle set:* A particle is a word used to denote a single sample obtained through random sampling algorithms such as Monte Carlo method. A particle set is a set of samples obtained through some sampling or estimation algorithms. In robotics, particles and particle sets are often used to represent distributions obtained from estimation algorithms such as sequential Monte Carlo method or CONDENSATION (**Con**ditional **Dens**ity Propag**ation**) algorithm.
- *Physical entity:* The target to be localized such as robots, humans or other objects.
- *Polar coordinate system:* Two-dimensional coordinate system in which position is specified by distance and direction from the origin [ISO19111]. In this specification, three-dimensional coordinate system (spherical coordinate system) or n-dimensional coordinate

13428 system may also be called as polar coordinate system.

- *Data Instance:* Here, the word 'data instance' is used for a RoLo data or its subcomponent such as a RoLo element, a RoLo position, a RoLo symbolic position or a GM_Position object.
- *Implicit (Type) Specification:* When a structure embedding some data instance holds a type specification for those data instances, those data instances are described to have an "implicit type specification." For example, a RoLo data is implicitly associated with a RoLo data specification when the data is passed through a RoLo stream that holds a RoLo data specification defined in its ability description.
- *Explicit (Type) Specification:* A data instance is said to have an "explicit type specification" if a reference to corresponding specification is provided in its attribute. For example, when a RoLo data has a reference to RoLo data specification as its 'spec' attribute, the RoLo data is said to have an explicit type specification.
- *Type Specification:* A "type specification" of a data instance is either an implicit type specification or an explicit type specification of an instance.
- *Incomplete (Type) Specification:* A type specification is called "incomplete" when it includes one or more "don't-care" elements.
- *Complete (Type) Specification:* A type specification is called "complete" when it does not include any "don't-care" element.
- *Consistent Type Specifications:* Two type specifications are called "consistent" when the two specifications own the same structure and each corresponding parts of them is the same or have a base-derivation (generalized-specialized) relation with each other, or when one of the corresponding parts are specified as "don't-care".
- *Unified (Type) Specification:* A "unified type specification" of a data instance is the result of unification of all the type specifications associated with the data instance. The type specifications to be unified shall be consistent. The unification of two type specifications is done by the following operation:

  For each part of the type specifications, do:

  1. When both of the corresponding type specifications are "don't-care", use "don't-care".
  2. When one of part of the two specifications is "don't-care", use the corresponding part from another specification.
  3. When both of the corresponding type specifications are not "don't-care", use the one which is much specialized.

6. Symbol

13429 *x, y, z* Cartesian coordinate

13429 r, θ, φ spherical coordinate

φ, λ, h geodetic coordinate (latitude, longitude, height)

α, β, γ orientation

x, y, z a fixed Cartesian coordinate system

X, Y, Z a rotating Cartesian coordinate system

~~No symbols are defined in this document.~~

7. Additional Information

13430 7.1 Submitters

The initial submissions that this specification is based on were submitted by the following people:

Kyuseo Han, Electronics and Telecommunications Research Institute (ETRI)
Yeonho Kim, Samsung Electronics Co., Ltd.
Shuichi Nishio, Japan Robot Association (JARA) / Advanced Research Institute International (ATR)

7.2 Submitting Organizations

The following organizations made the initial submission that this specification is based on:

Electronics and Telecommunications Research Institute (ETRI)
Japan Robot Association (JARA)
Samsung Electronics Co., Ltd.

7.3 Supporting Organizations

The following organization~~companie~~s supported parts of this specification:

Hitachi, Ltd.
National Institute of Advanced Industrial Science and Technology (AIST)
New Energy and Industrial Technology Development Organization (NEDO)
Shibaura Institute of Technology
Technologic Arts Incorpo~~p~~rated
University of Tsukuba

7.4 Acknowledgements

The following people supported parts of this specification:

Su-Young Chi, Electronics and Telecommunications Research Institute
Yun Koo Chung, Electronics and Telecommunications Research Institute
Miwako Doi, Toshiba Corporation
Kenjirou Fujii, Hitachi Industrial Equipment Systems Co., Ltd.
Yoshimasa Hata, Japan Robot Association
Yasuo Hayashibara, Chiba Institute of Technology
Ryota Hiura, Mitsubishi Heavy Industries, Ltd.
Toshio Hori, National Institute of Advanced Industrial Science and Technology
Masato Iehara, Mitsubishi Heavy Industries, Ltd.
Wataru Inamura, IHI Corporation
13431 Jaeyeong Lee, Electronics and Telecommunications Research Institute
Takahide Kanehara, Yaskawa Electric Corporation
Tetsuo Kotoku, National Institute of Advanced Industrial Science and Technology
Makoto Mizukawa, Shibaura Institute of Technology
Kouji Murakami, Kyushu University
Yoshisada Nagasaka, National Agriculture and Food Research Organization
Itsuki Noda, National Institute of Advanced Industrial Science and Technology
Kohtaro Ohba, National Institute of Advanced Industrial Science and Technology
Fumio Ozaki, Toshiba Corporation
Takeshi Sakamoto, Technologic Arts Incorporated
Takashi Suehiro, National Institute of Advanced Industrial Science and Technology
Tetsuo Tomizawa, National Institute of Advanced Industrial Science and Technology
Takashi Tsubouchi, University of Tsukuba
Tomoki Yamashita, Maekawa MFG Co. Ltd.
Masayoshi Yokomachi, New Energy and Industrial Technology Development Organization
Wonpil Yu, Electronics and Telecommunications Research Institute

7.5 Background

This specification defines a localization service that can handle data and usages specific to use in robotics. It includes a platform-independent model (PIM) as well as a mapping of this PIM to platform-specific models (PSM) defined by C++.

Location information is a crucial factor in providing robotic services of every kind. Typically, a robotic system is defined as an apparatus equipped with the function of interacting with physical entities in the environment. Navigation, manipulation and human-robot interaction are typical features that require physical interaction with the environment, which distinguish a robotic system from information appliances. On performing such tasks, robots require geometric association between physical entities of interest and the robot itself for implementing and/or performing the given service scenario. Besides these examples, the number of location-based robotic tasks is continuously increasing as personal or service robot fields gradually expand, from controlled, stable factory environments to indeterminate, uncertain daily environments. However, currently there exists no standard means to represent the necessary location-related information in robotics, nor any common interface for constructing localization related software modules.

> Note: In the context of this proposal and the originating RFP, the word "**localization**" means "*to **locate** some **physical entities** through analysis of sensor data*", consistent with the common use of this term in robotics. Here the word "**locate**" may include not only measuring the position in the spatio-temporal space, but also heading orientation or pose information of the entity, or additional information such as error estimation or time of measurement. Also, the word "**physical entity**" (or "entity" in short) is used to describe the target to be localized, including robots, humans or other objects.

13432 Geographic Information System (GIS) is one of the most popular and established systems that treats location information. Many spatio-~~A Geographic Information System (GIS) is one of the most popular and established systems that treats location information. Many spatio~~temporal location related specifications have been standardized in the International Organization for Standardization (ISO/TC211), and there already exist versatile production services based on these standards such as driving navigation systems or resource databases. However, current GIS specifications are not powerful enough to represent or treat information required in the field of robotics.

Although localization is still one of the main research topics in the field of robotics, the fundamental methodology and elements necessary are becoming established. Standardizing localization result representation and related interfaces in a generic form, independent to specific algorithms or equipment, are significant for decreasing costs and accelerating the market growth of robotic services. Moreover, clarifying what types of information are required in the field of robotics shall be useful for equipment vendors such as sensor manufacturers.

In this proposal, a new framework for robotic localization (**RoLo**) service, i.e. representing and treating location information specific to robotic usage, is presented. Notions and items necessary for treating location information in robotic usage are reorganized and rearranged, in a generic form independent to specific algorithms or types of robotic services. This was done through extensive surveys and case studies on current and ongoing robotic products and researches. Based on the widespread GIS standard, a new specification for RoLo~~, i.e. representing and treating location information specific to robotic usage, is presented. Notions and items necessary for treating location information in robotic usage are reorganized and rearranged, in a generic form independent to specific algorithms or types of robotic services. This was done through extensive surveys and case studies on current and ongoing robotic products and researches. Based on the widespread GIS standard, a new specification for robotic localization~~ services is proposed.

Figure 1 illustrates a typical robotic service situation where localization of various entities is required. Here, a robot in service needs to obtain the location of a cellular phone, utilizing information from various robotic entities in the environment. These robotic entities have the ability to estimate the location of the entities within their sensing range. Thus, the problem here is to aggregate the location estimations from the robotic entities, and to localize the cellular phone in target. However, this example also shows several factors that makes the localization service in robotics a difficult, challenging issue. **A) Some sensors only provide partial location information.** For example, the camera sensor can only provide 2D information, and RF tag reader can only provide proximity information. **B) Sensor outputs are not always correct.** Sometimes, they might measure two or more entities as a single object, or even miss it. This erroneous report occurs frequently when sensors are used in the uncontrolled daily environment. In order to tackle this erroneous situation, sensor outputs are usually treated to be probabilistic, with error estimation information. **C) Matching observations between different sensors require efforts.** Imagine you are viewing two photographs of a crowded street corner, taken from different angles but on the same instant. The issue here is to match every single person in one photograph to another. This is much more difficult when matching the observed entities from the wall camera and the output from the laser range scanner installed in the blue robot, as these two sensors measure different aspects of objects. This issue, the *identity association problem*, happens every time multiple sensors are used. In other word, you are always not sure about the identity of the entity sensed. Thus, identity information shall also be treated to be probabilistic.

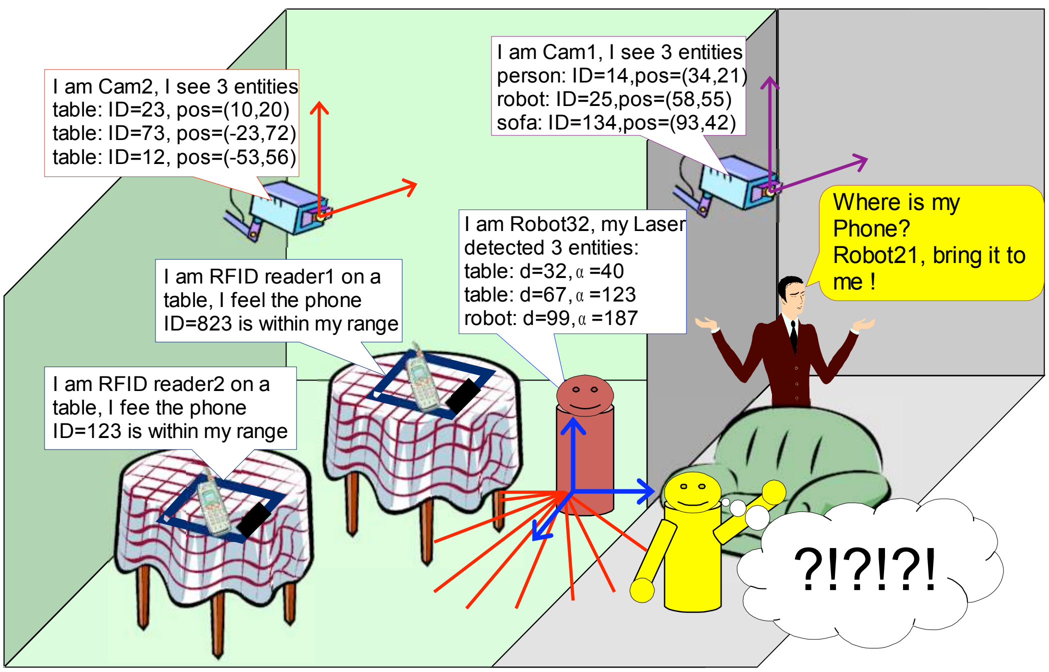


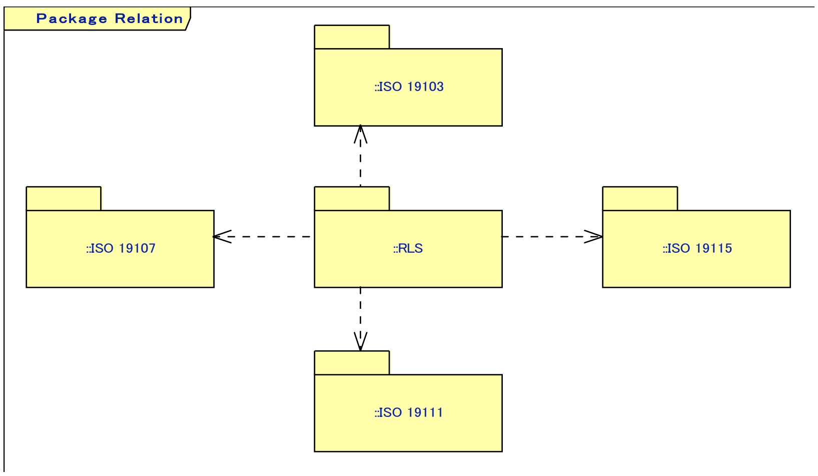
Figure 1: Example of a typical robotic service situation requiring localization of an entity

13432 As can be seen from these examples, operations in robotics require a much more detailed representation of location information. Still, interoperability with the current GIS systems shall be supported. In this proposal, we define a new framework for representing and treating location information suitable for robotic use, by extending existing GIS specifications. Using the GIS specification as a basis of the proposal will make it easy for robots to interconnect with existing GIS-based systems and utilize existing geographic datasets. This will also ease the use

13432 of this specification in the emerging fields of next-generation GIS systems, sensor network systems, or location based systems where advanced positioning methods and complex data processing similar to robotics usage is required. Figure 4 illustrates the existing GIS standards that are related with this specification.

~~**Figure 2 - Example of a typical robotic service situation requiring localization of an entity**~~

~~As can be seen from these examples, operations in robotics require a much more detailed representation of location information. Still, interoperability with the current GIS systems shall be supported. In this proposal, we define a new framework for representing and treating location information suitable for robotic use, by extending existing GIS specifications. Using the GIS specification as a basis of the proposal will make it easy for robots to interconnect with existing GIS-based systems and utilize existing geographic datasets. Figure 3 illustrates the existing GIS standards that are related wit this specification.~~



~~**Figure 3 - Relation of Robotic Localization Service specification with existing GIS specifications**~~

~~In order to fulfill the requirements for robotic localization, the following items are defined in the Platform-Independent Model, some part as an extension to existing GIS specification.~~

- ~~Coordinate reference systems for representing information necessary for robotics usage~~
- ~~Identity information representation~~
- ~~Error information representation~~
- ~~Data architecture for representing and operating on robotic localization results~~
- ~~Robotic localization service interfaces~~

12916

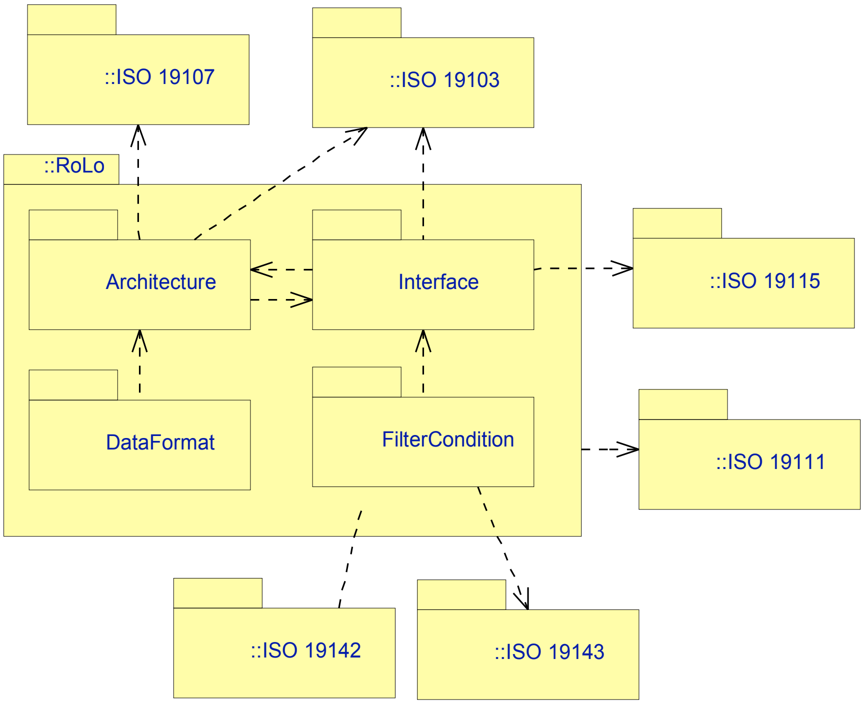


Figure 4: Relation of Robotic Localization Service specification with existing GIS specifications

In order to fulfill the requirements for robotic localization, the following items are defined in the PIM, some part as an extension to existing GIS specification.

- (Architecture package) Data architecture for representing structures and accompanying operations for representing information necessary for robotics usage. These include coordinate system / coordinate reference system definitions for treating essential information such as pose or identity information, or structures for representing error estimation.
- (DataFormat package) Data formats for formatting and exchanging resulting localization data.
- (Interface package) Service interface for treating resulting localization data. This includes advanced facilities that will be a basis for dynamically exchanging or negotiating module functionality information.

8. Platform Independent Model

12916 The PIM consists of three parts:

1. Architecture package

   The architecture package defines a new framework for representing location information required in the field of robotics. See section 7.3.

12916 2. DataFormat package

The data format package defines how the defined data is represented for exchange amongst RoLo modules. See section 7.4.

3. FilterCondition package

The filter condition package defines methods for filtering localization results. See section 7.5.

4. Interface package

The interface package defines an API for data passing and configuration of RoLo modules. See section 7.5.

8.1 Format and Conventions

8.1.1 Class

Classes described in this PIM are documented using tables of the following format:

Table xx: <class name>

<table>
<tr><td colspan="6">Description: <description></td></tr>
<tr><td colspan="6">Derived From: <parent class></td></tr>
<tr><td colspan="6">Attributes</td></tr>
<tr><td colspan="2"><attribute name></td><td><attribute type></td><td><obligation></td><td><occurrence></td><td><description></td></tr>
<tr><td colspan="2">…</td><td>…</td><td></td><td></td><td>…</td></tr>
<tr><td colspan="6">Operations</td></tr>
<tr><td colspan="2"><operation name></td><td colspan="4"><description></td></tr>
<tr><td><direction></td><td><parameter name></td><td colspan="2"><parameter type></td><td colspan="2"><description></td></tr>
<tr><td>…</td><td>…</td><td colspan="2">…</td><td colspan="2">…</td></tr>
</table>

13398 Note that derived attributes or operations are not described explicitly. Also, as the type of return code for every operation in this specification is Returncode_t which is defined in section 7.2, this is omitted in the description table.

~~Note that derived attributes or operations are not described explicitly. As the type of return code for every operation in this specification is Returncode_t which is defined in section 6.2, this is omitted in the description table.~~

The 'obligation' and 'occurrence' are defined as following.

Obligation

- **M (mandatory):** This attribute shall always be supplied.
- **O (optional):** This attribute may be supplied.
- **C (conditional):** This attribute shall be supplied under a condition. The condition is given as a part of the attribute description.

Occurrence

The occurrence column indicates the maximum number of occurrences of the attribute values that are permissible. The following denotes special meanings.

- **N:** No upper limit in the number of occurrences.
- **ord:** The appearance of the attribute values shall be ordered.
- **unq:** The appeared attribute values shall be unique.

8.1.2 Enumeration

Enumerations are documented as follows:

Table xx: <enumeration name>

<constant name>	<description>
…	…

8.2 Return Codes

At the PIM level, we have modeled errors as operation return codes typed **Returncode_t**. Each PSM may map these to either return codes or exceptions. The complete list of return codes is indicated below.

Table 1 Returncode_t enumeration

OK	Successful return.
ERROR	Generic, unspecified error.
BAD_PARAMETER	Illegal parameter value.
UNSUPPORTED_PARAMETER	Unsupported parameter.
UNSUPPORTED_OPERATION	Unsupported operation.
TIMEOUT	The operation timed out.

12916

8.3 Architecture Package

Modern robotic algorithms related to localization require not only simple spatial positioning information. Generally, various types of information related to spatial position are also required. In order to obtain precise results, measurement time and error estimation is crucial, especially when integrating measurements from multiple sensors. For robotics usage, complex spatial positioning such as pose information is also important. When sensors in use can perform measurements of multiple entities at once, identity information is also necessary in order to distinguish and associate measurements. As such, there is a variety of other information to be expressed in combination with simple spatial positioning. In order to make various robotic services treat and process this versatile information easily and effectively, our idea is to represent this heterogeneous information under a common, unified framework.

In this section, we propose a new framework for representing location information required in the field of robotics, by extending existing GIS specifications. Three types of information required in robotics usage are defined, and lastly, a generic framework for representing structured robotic localization results (RoLo architecture) is defined.

13434 Note that, although the ISO 19111 specification assumes every CS to be 2 or 3 dimensional [ISO19111], in this specification, we do not assume any limitation on the number of dimensions on any coordinate systems. This is to enable representation of complex data such as feature points defined over multi-dimensional space. Also note that this does not violate the ISO 19111 standard where no formal limitation is specified on the number of dimensions. One issue is how to treat the attribute bounded to specific feature in the real space such as axisDirection (type CS_AxisDirection) in CS_CoordinateSystemAxis which is a mandatory attribute and where the type is defined as an finite enumeration of direction names such as 'north' or 'south'. It is clear that these values are not suitable for some robotics usage such as for relative or mobile coordinate reference systems. We thus recommend that implementers and users of this specification to simply ignore this attribute and to set this value as the first element in the enumeration, 'north', if necessary. This is a safe solution as we cannot expect GIS systems to treat data based on this specification correctly; we only expect data from GIS systems to be treated on systems based on this robotic specification.

13106 8.3.1 Relative Coordinate Reference Systems

In this section, relative coordinate reference systems are defined which may lack fixed relation with the earth or users have no interest in referencing them to other coordinate reference systems. We categorize relative coordinate reference systems in two types, static and dynamic. A coordinate reference system on mobile platforms, mobile coordinate reference system, is defined as a dynamic relative coordinate reference system. That is, the relation with other coordinate reference systems may change by time.

The GIS standard on spatial reference system [ISO19111] allows the definition and use of such relative and mobile coordinate reference systems. However, there is no specific model or description on these systems. As these systems are quite commonly used in the field of robotics, here we explicitly define structures and operations specific to these coordinate reference systems. Although here we only define coordinate reference systems based on two coordinate systems of frequent usage, SC_CartesianCS and SC_PolarCS, users may define derivatives of relative or mobile coordinate reference system based on the coordinate system of their interest.

12916, 12919, 12920, 12922, 13116, 13398, 14003

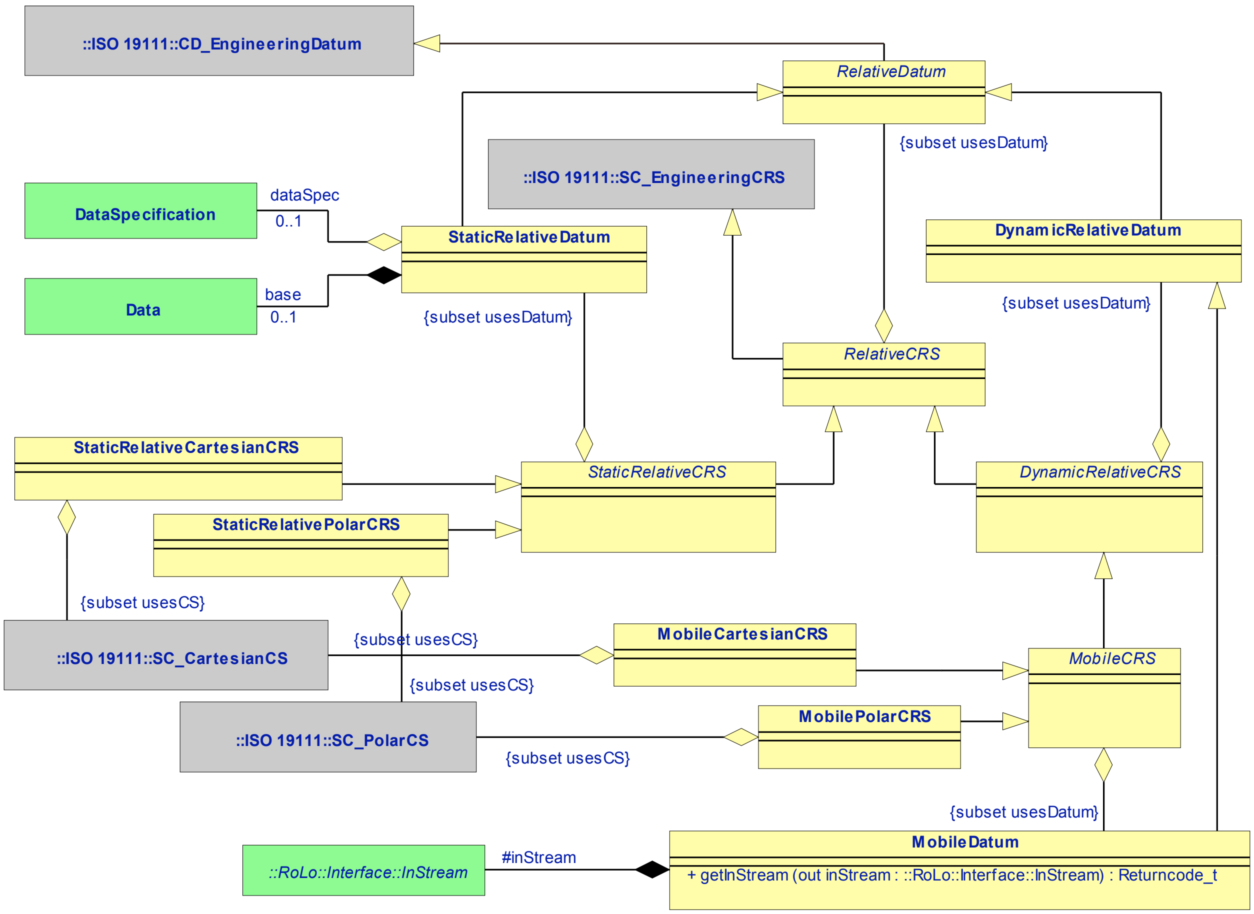


Figure 5: Relative and Mobile coordinate reference system

13106 **Table 2 - RelativeCRS class**

Description: Base abstract class for representing relative coordinate reference systems.
Derived From: SC_EngineeringCRS [ISO19111]
Note: Values for the attribute 'usesDatum' which is derived from parent class shall be limited to instances of RelativeDatum or its inherited classes.

13106 **Table 3 - RelativeDatum clas~~ Representing Robotic Localization Result~~s**

Description: Datum for relative coordinate reference systems.
Derived From: CD_EngineeringDatum [ISO19111]

~~Modern robotic algorithms related to localization require not only simple spatial positioning information. Generally, various types of information related to spatial position are also required. In order to obtain precise results, measurement time and error estimation is crucial, especially when integrating measurements from multiple sensors. For robotics usage, complex~~

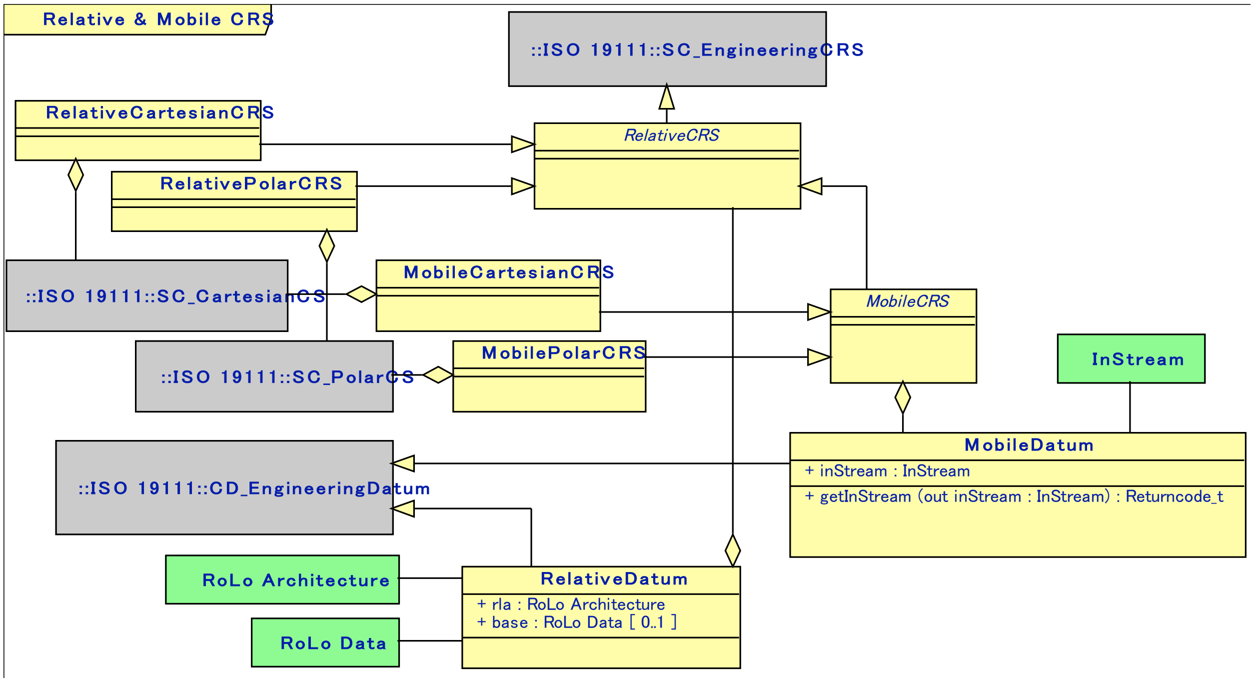
spatial positioning such as pose information is also important. When sensors in use can perform measurements of multiple entities at once, identity information is also necessary in order to distinguish and associate measurements. As such, there is a variety of other information to be expressed in combination with simple spatial positioning. In order to make various robotic services treat and process this versatile information easily and effectively, our idea is to represent this heterogeneous information under a common, unified framework.

In this section, we propose a new framework for representing location information required in the field of robotics, by extending existing GIS specifications. Three types of information required in robotics usage are defined, and lastly, a generic framework for representing structured robotic localization results is defined.

8.3.2 Relative and Mobile Coordinate Reference Systems

In this specification, a relative coordinate reference system is defined as a coordinate reference system where the relation with the fixed world is not known at some instant, or users have no interest in referencing it to other coordinate reference systems. A mobile coordinate reference system is defined as a relative coordinate reference system on mobile platforms. That is, the relation with other coordinate reference systems may change by time.

The GIS standard on spatial reference system [ISO19111] allows the definition and use of such relative and mobile coordinate reference systems. However, there is no specific model or description on these systems. As these systems are quite commonly used in the field of robotics, here we explicitly define structures and operations specific to these coordinate reference systems. Although here we define coordinate reference systems based on two coordinate systems of frequent usage, SC_CartesianCS and SC_PolarCS, users may define relative or mobile coordinate reference systems derivatives based on the coordinate system of their interest.



~~Figure 6 - Relative and Mobile c oordinate reference system~~

12922, 13106

Table 4 - Static RelativeCRS class

Description: Abstract class for representing relative coordinate reference systems that have static relation with other CRS(s).
Derived From: RelativeCRS
Note: Values for the attribute 'usesDatum' which is derived from parent class shall be limited to instances of StaticRelativeDatum or its inherited classes.
~~**Description:** Base abstract class for representing relative coordinate reference systems.~~
~~**Derived From:** SC_EngineeringCRS [ISO19111]~~

12922, 13106

Table 5 - Static RelativeCartesianCRS class

Description: Static relative coordinate reference systems based on Cartesian coordinate system.
Derived From: StaticRelativeCRS
Note: Values for the attribute 'usesCS' which is derived from parent class shall be limited to instances of SC_CartesianCS [ISO19111] or its inherited classes.
~~**Description:** Relative coordinate reference systems based on Cartesian coordinate system.~~
~~**Derived From:** RelativeCRS~~

12922, 13106

Table 6 - StaticRelativePolarCRS class

Description: Static relative coordinate reference system based on polar coordinate system.
Derived From: StaticRelativeCRS
Note: Values for the attribute 'usesCS' which is derived from parent class shall be limited to instances of SC_PolarCS [ISO19111] or its inherited classes.
~~**Description:** Relative coordinate reference system based on Polar coordinate system.~~
~~**Derived From:** RelativeCRS~~

12923, 13106, 13116, 13435, 13441, 13398

Table 7 - StaticRelativeDatum class

Description: Datum for static relative coordinate reference system.				
Derived From: RelativeDatum				
Attributes				
dataSpec	DataSpecification	O	1	A RoLo data specification indicating allowed structure for the 'base' attribute. If the coordinate reference system in target holds no relation with other coordinate reference systems, this may be omitted.
base	Data	O	1	A RoLo data for determining relation to other coordinate reference system. Typically, this data includes spatial position for origin and pose for axis direction. If no relation with other coordinate reference systems is required, this may be omitted.
~~**Description:** Datum for the relative coordinate reference system. Holds a RoLo Data that~~				
~~**Derived From:** CD_EngineeringDatum [ISO19111]~~				
~~**Attributes**~~				
~~rla~~	~~RoLo Architecture~~	~~O~~	~~1~~	~~RoLo Architecture indicating the allowed structure for the base attribute. If the base attribute contains structural information, this may be omitted.~~
~~base~~	~~RoLoData~~	~~O~~	~~1~~	~~Data for determining the reference to other coordinate reference systems. Typically, this data includes spatial position for origin and pose for axis direction. If no relation with other coordinate reference systems is required, this can be empty.~~

13106

Table 8 - DynamicRelativ~~Mobil~~eCRS class

Description: Abstract base class for representing dynamic relative coordinate reference systems.
Derived From: RelativeCRS
Note: Values for the attribute 'usesDatum' which is derived from parent class shall be limited to instances of DynamicRelativeDatum or its inherited classes.
~~**Description:** Abstract base class for representing mobile coordinate reference system.~~
~~**Derived From:** RelativeCRS~~

Table 9 - DynamicRelativeDatum~~MobileCartesianCRS~~ class

Description: Datum for dynamic relative coordinate reference system.

Derived From: RelativeDatum
Description: Mobile coordinate reference systems based on Cartesian coordinate system.
Derived From: MobileCRS

12922, 13106

Table 10 - MobilePolarCRS class

Description: Abstract base class for representing mobile coordinate reference systems.
Derived From: DynamicRelativeCRS
Note: Values for the attribute 'usesDatum' which is derived from parent class shall be limited to instances of MobileDatum or its inherited classes.
Description: Moblie coordinate reference system based on Polar coordinate system.
Derived From: MobileCRS

12922

Table 11 - MobileCartesianCRSDatum class

Description: Mobile coordinate reference systems based on Cartesian coordinate system.
Derived From: MobileCRS
Note: Values for the attribute 'usesCS' which is derived from parent class shall be limited to instances of SC_CartesianCS [ISO19111] or its inherited classes.

Description: Datum for the mobile coordinate reference system. This datum holds an InStream instance that is used to obtain positional information for determining the relation between the mobile coordinate reference system in target and another coordinate reference system. Users shall connect an OutSteram object to this InStream, or shall supply positional information directly by the setData method of InStream. For example, if the mobile coordinate system is based on Cartesian coordinate system, spatial position information for mapping the origin and orientation information for determining axis directions may be supplied. However, some transformation algorithm require more complicated information such as measurement time or error information. The necessary information required can be determined by the ability information obtainable from the inStream object.				
Derived From: CD_EngineeringDatum [ISO19111]				
Attributes				
inStream	InStream	M	1	Input stream for obtaining base position.
Operations				
getInStream	Returns the input stream in use.			
out inStream	InStream	Reference to the inStream object used in this datum.		

12922

Table 12 - MobilePolarCRS class

Description: Mobile coordinate reference system based on polar coordinate system.
Derived From: MobileCRS
Note: Values for the attribute 'usesCS' which is derived from parent class shall be limited to instances of SC_PolarCS [ISO19111] or its inherited classes.

12916, 13106, 13436, 13441, 14003

Table 13 - MobileDatum clas~~Figure 7 - Mobile CRS operation~~s

Description: Datum for mobile coordinate reference systems. This datum holds a RoLo input stream that is used to obtain positional information for determining the relation between the mobile coordinate reference system in target and another coordinate reference system. Users shall connect a RoLo output stream to this input stream, or shall supply positional information directly by the 'setData' method of this input stream. For example, if the mobile coordinate system is based on Cartesian coordinate system, spatial position information for mapping the origin and orientation information for determining axis directions may be supplied. However, some transformation algorithms require more complicated information such as measurement time or error information. The necessary information required can be determined by the ability description of the input stream.				
Derived From: DynamicRelativeDatum				
Attributes				
inStream (protected)	InStream (RoLo::Interface)	M	1	Input stream for obtaining base position.
Operations				
getInStream	Returns the input stream in use.			
out inStream	InStream (RoLo::Interface)	InStream instance used in this datum.		

12924

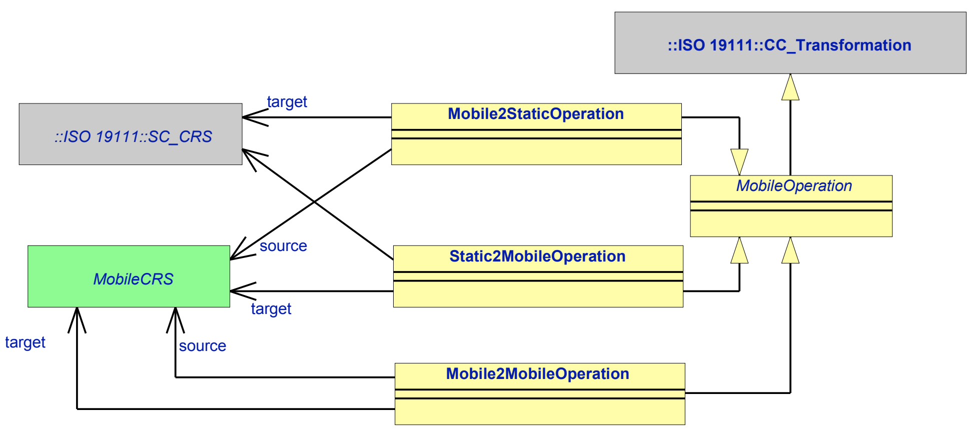


Figure 8: Mobile CRS operations

Table 14 - Mobile~~RoLo Mobile~~ Operation class

Description: Abstract base class for operations between mobile coordinate reference system and other coordinate reference systems.	
Derived From: CC_Transformation [ISO19111]	

Table 15 - Mobile2Static Operation class

Description: Transformation operation from mobile coordinate reference systems to other static, non-mobile coordinate reference systems.				
Derived From: MobileOperation				
Attributes				
source	MobileCRS	M	1	The source mobile coordinate reference system.

13107

target	CS_CRS [ISO19111]	M	1	The target coordinate reference system.
Note: Values for the attribute ‘target’ shall not be an instance of DynamicRelativeCRS or its inherited classes.				
~~**Description:** Transformation operation from mobile coordinate reference systems to other static, non-mobile coordinate reference systems..~~				
~~**Derived From:** RoLo Mobile Operation~~				
~~**Attributes**~~				
~~source~~	~~MobileCRS~~	~~M~~	~~1~~	~~The source mobile coordinate reference system.~~
~~target~~	~~CS_CRS [ISO19111]~~	~~M~~	~~1~~	~~The target coordinate reference system.~~

12940, 13107

Table 16 - Static2Mobile~~Fixed2Mobile~~ Operation class

Description: Transformation operation from other static, non-mobile coordinate reference systems to mobile coordinate reference systems.				
Derived From: MobileOperation				
Attributes				
source	CS_CRS [ISO19111]	M	1	The source coordinate reference system.
target	MobileCRS	M	1	The target mobile coordinate reference system.
Note: Values for the attribute ‘source’ shall not be an instance of DynamicRelativeCRS or its inherited classes.				
~~**Description:** Transformation operation from other static coordinate reference systems to mobile coordinate reference systems..~~				
~~**Derived From:** RoLo Mobile Operation~~				
~~**Attributes**~~				
~~source~~	~~CS_CRS [ISO19111]~~	~~M~~	~~1~~	~~The source coordinate reference system.~~
~~target~~	~~MobileCRS~~	~~M~~	~~1~~	~~The target mobile coordinate reference system.~~

Table 17 - Mobile2Mobile Operation class

Description: Transformation operation between mobile coordinate reference systems.				
Derived From: MobileOperation				
Attributes				
source	MobileCRS	M	1	The source mobile coordinate reference system.
target	MobileCRS	M	1	The target mobile coordinate reference system.

8.3.3 Identity Information

Identity (ID), which is assigned for each localized targets, can also be treated as a value on some coordinate reference system. For example, MAC addresses used in Ethernet communication protocols can be represented as a coordinate value on a two-dimensional coordinate system, vendor code and vendor-dependent code. Electric Product Code (EPC) or ucode, used for identifying RF tags, is another example of identification systems defined by a multi-dimensional coordinate system. There also exist some ID systems, such as family names, that are usually not explicitly defined over some mathematical structure.

In general, each sensor holds its own ID system and each entity observed is assigned an ID from this local ID system. This is because, at least on the initial stage, there are no means to assign the observed entity a global ID. Thus, when multiple sensors are in use, there exist multiple local ID systems independent to each other, and it becomes necessary to properly manage and integrate these ID systems. Resolving the bindings between each local ID systems is called the ID association problem, and is one of the major research issues in the robotic localization field. Also, as we saw in the overview section, ID assignments are probabilistic, just like other location information.

Under these considerations, here we define coordinate reference systems and related structures for representing identity information. Here, two coordinate reference systems and accompanying coordinate systems are defined, for identity systems that are represented in numerical values and symbolic values. The actual coordinate value holding structure in GIS standard [ISO19107] only allows numeric values as coordinate value elements. Thus, similar structures in use with symbolic values are also defined.

Note that, operations on identity information (such as conversion from numeric ID to symbolic ID or mapping between different ID systems) can be constructed using CC_CoordinateOperation or relevant classes specified in GIS standard [ISO19111]. This is because the identity information define here is represented by using derived classes from GIS coordinate systems and coordinate reference systems.

12925,
13108,
13998,
14004

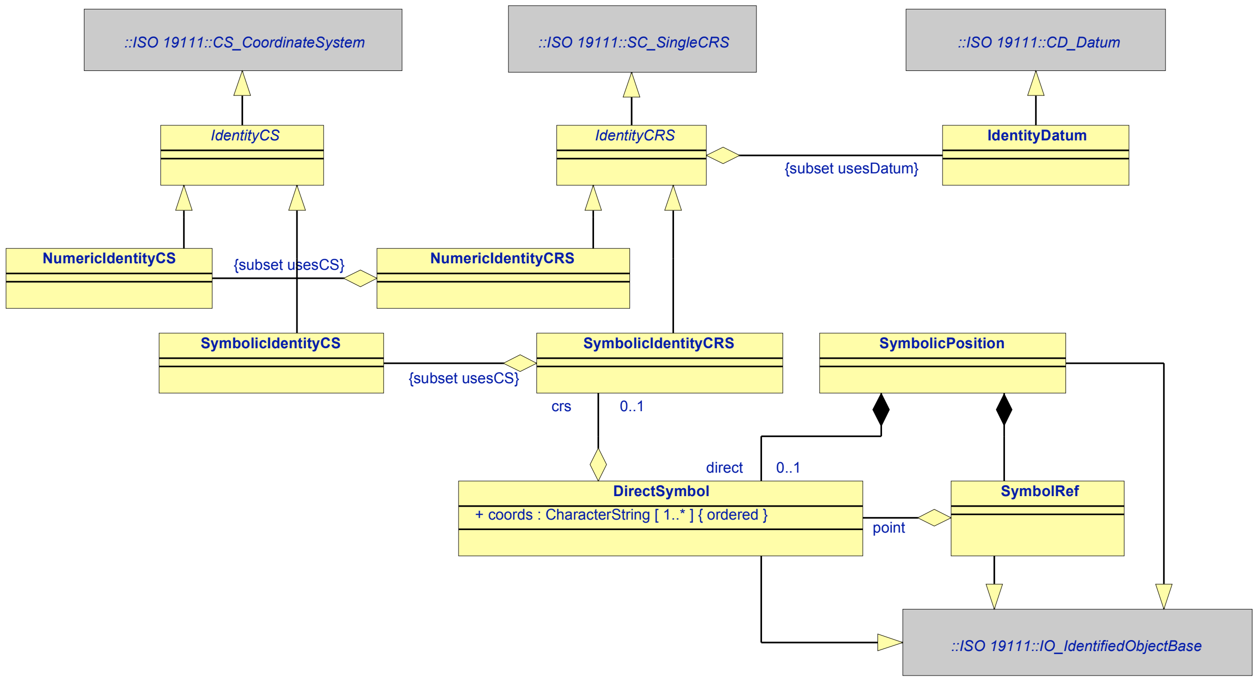


Figure 9: Identity Information

Table 18 - IdentityCS class

Description: Coordinate systems for identity information.	
Derived From: CS_CoordinateSystem [ISO19111]	

Table 19 - NumericIdentityCS class

Description: Coordinate system for identity information, where each axis is defined over numerical values.	
Derived From: IdentityCS	

13108

Table 20 - SymbolicIdentityCS class

Description: Coordinate system for identity information, where each axis is defined over a set of symbolic values.	
Derived From: IdentityCS	

Table 21 - Identity Datum class

Description: Datum for identity coordinate reference systems.	
Derived From: CD_Datum [ISO19111]	

12925, 12940 **Table 22 - IdentityCRS class**

Description: Base abstract class for representing coordinate reference systems for identity information.
Derived From: SC_SingleCRS [ISO19111]
Note: Values for the attribute 'usesDatum' which is derived from parent class shall be limited to instances of IdentityDatum or its inherited classes.
~~**Description:** Coordinate reference system for identity information.~~
~~**Derived From:** CD_Datum [ISO19111]~~

12925 **Table 23 - NumericIdentityCRS class**

Description: Coordinate reference system for identity information, where each axis is defined over numerical values.
Derived From: IdentityCRS
Note: Values for the attribute 'usesCS' which is derived from parent class shall be limited to instances of NumericIdentityCS or its inherited classes.
~~**Description:** Coordinate reference system for identity information, where each axis is defined over numerical values.~~
~~**Derived From:** IdentityCRS~~

12925, 13108 **Table 24 - SymbolicIdentityCRS class**

Description: Coordinate reference system for identity information, where each axis is defined over a set of symbolic values.
Derived From: IdentityCRS
Note: Values for the attribute 'usesCS' which is derived from parent class shall be limited to instances of SymbolicIdentityCS or its inherited classes.
~~**Description:** Coordinate reference system for identity information, where each axis is defined over a set of symbolic values.~~
~~**Derived From:** IdentityCRS~~

12940, 13108, 13998 **Table 25 - DirectSymbol class**

Description: Class for holding symbolic identity information.				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes				
coords	CharacterString	M	N ord	Values for each of the coordinate system axis.
crs	SymbolicIdentityCRS	O	1	Reference to the coordinate reference system this data belongs to.
~~**Description:** Class for holding symbolic identity information.~~				
~~**Derived From:** IO_IdentifiedObjectBase [ISO19111]~~				
~~**Attributes**~~				
~~coordinate~~	~~CharacterString~~	~~M~~	~~1~~	~~Values for each of the coordinate system axis.~~
~~dimension~~	~~Integer~~	~~O~~	~~1~~	~~Number of dimensions of this data.~~
~~coordinateReferenceSystem~~	~~SymbolicCRS~~	~~O~~	~~1~~	~~Reference to the coordinate reference system this data belongs to.~~

13318 **Table 26 - ~~RoLo~~ SymbolRef class**

Description: Data holder for a reference to DirectSymbol				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes				
point	DirectSymbol	M	1	Reference to the target DirectSymbol class instance.

12926 **Table 27 - Symbolic~~RoLo Symbolic~~ Position class**

Description: Union of DirectSymbol and SymbolRef. This class is used as a data holder for accessing symbolic information transparently, whether it is directly held or indirectly referenced.				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes				
direct	DirectSymbol	C	1	Symbolic identity data.
indirect	SymbolRef	C	1	Reference to symbolic identity data.
Condition: Either one of the element shall be contained.				

8.3.4 Error Information

Every sensing system in the real world cannot avoid having measurement error. As such, it is essential to know the reliability or deviation of measurements for performing localization and for utilizing the resulting estimation. Error information plays an important role in robotic operations. In GIS specifications, the only error concerned is the expected reliability of inter-coordinate transformation. However, complex and detailed error descriptions are required in modern localization methods. Thus, here we define additional structures for representing and operating on error information.

RoLo Error Type

Similar to the relation of coordinate reference system and the position in the traditional GIS systems, we here define RoLo error types for describing the nature of error information. Every RoLo error holds a reference to an error type (either implicitly or explicitly; see section 8.7), which indicates how this error is represented. This means that, the same error data can be represented in a different manner. Thus, operations for transforming between different error types are defined. ~~Error types for describing the nature of error information. Every error information data references an error type, which indicates how the error is represented. This means that, the same error data can be represented in a different manner. Thus, operations for transforming between different error types are defined.~~

13317 RoLo error types may also be structured to for a hierarchy. Just as the normal class inheritance relationships, often error types may be related to each other. For example, a linear mixture model distribution is one limited form of general mixture model where models mixture is

13317 performed through linear operations. Here the hierarchy of RoLo error types is specified by inter-object relationships, and not by inter-class relationships. This is to be consistent with other specification data types such as coordinate reference system, coordinate system or RoLo data specification. Figure 10 shows some RoLo error types and their relationships corresponding to the RoLo error classes defined afterwards~~Note that, error information in the context of localization cannot exist solely by itself. Error information is an attribute to the location value. Thus, there exist two types of operation on error information in general. 1) Change in error representation type, and 2) change in the coordinate system the target location value is based on. The former operation is described in this section, and the latter is described later with the description on RoLo Architecture~~.

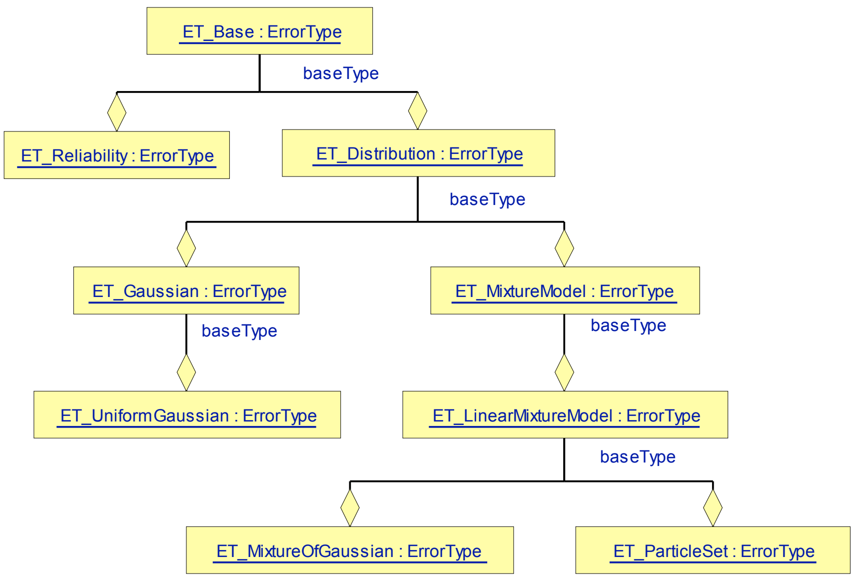


Figure 10: Hierarchy of RoLo Error Types

13116 Note that, error information in the context of localization cannot exist solely by itself. Error information is an attribute to the location value. Thus, there exist two types of operation on error information in general. 1) Change in error representation type, and 2) change in the coordinate system the target location value is based on. The former operation is described in this section, and the latter is described later with the description on RoLo data specification.

13317

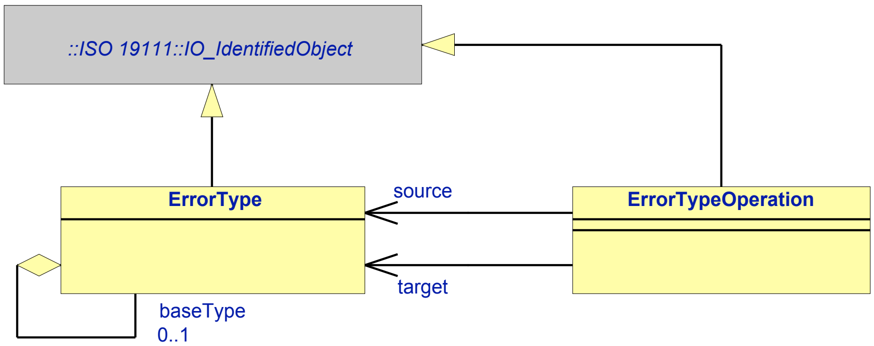


Figure 11 - RoLo Error Type

13317

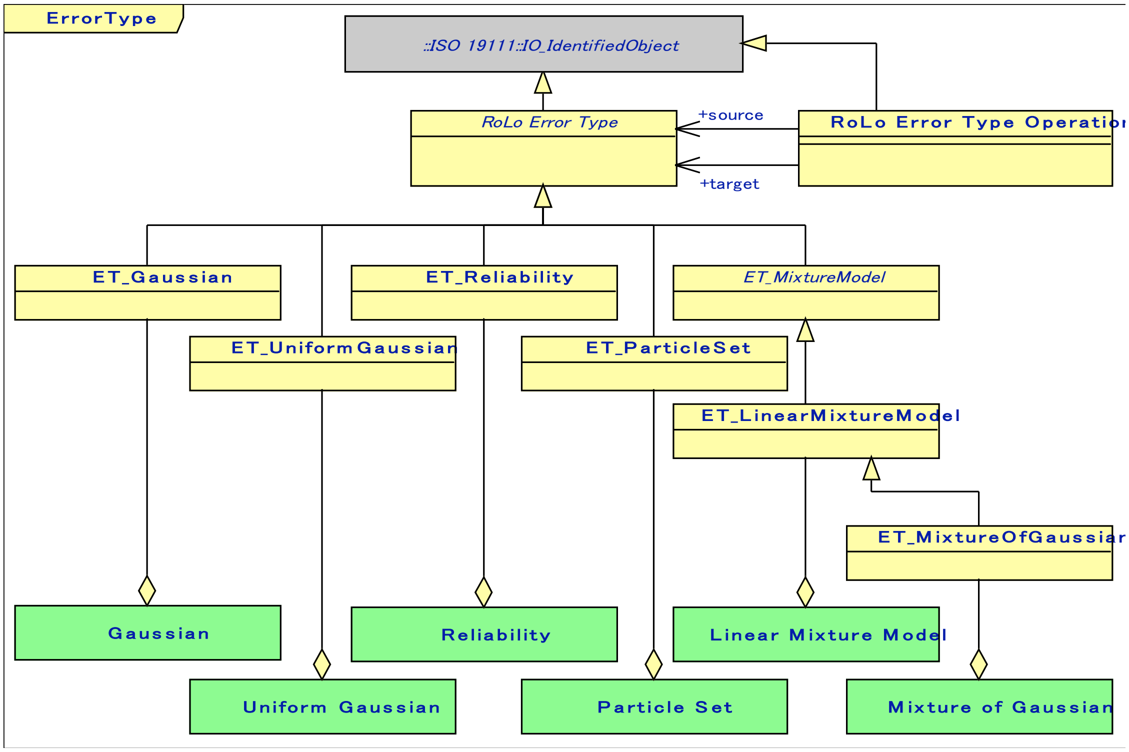
Table 28 - ErrorType class

Description: Class for representing RoLo error types.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes				
baseType	ErrorType	O	1	Reference to a RoLo error type which is the base type of this RoLo error type. This attribute is used to represent hierarchical relationships among RoLo error types.

~~**Figure 12 - Error Type**~~

~~Table 29 - RoLo Error Type class~~

~~**Description:** Base class for describing error type.~~
~~**Derived From:** IO_IdentifiedObject [ISO19111]~~



13441

Table 30 - ErrorType~~RoLo Error Type~~ Operation class

Description: Denotes transformation of RoLo error into a different RoLo error type.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes				
source	ErrorType	M	1	Source RoLo error type.
target	ErrorType	M	1	Target RoLo error type.

13317

~~Table 31 - ET_Reliablity class~~

~~**Description:** Error representation by a single scalar value.~~
~~**Derived From:** RoLo Error Type~~

~~Table 32 - ET_Gaussian class~~

~~**Description:** Error representation as a single n-dimensional normal distribution.~~
~~**Derived From:** RoLo Error Type~~

13317

~~Table 33 - ET_UniformGaussian class~~

~~**Description:** Error representation as a single n-dimensional normal distribution, where deviation is assumed to be uniform in every dimension.~~
~~**Derived From:** RoLo Error Type~~

~~Table 34 - ET_ParticleSet class~~

~~**Description:** Error representation as a set of particles. Particle set is used for approximating complicated distributions by means of limited number of sampling points.~~
~~**Derived From:** RoLo Error Type~~

~~Table 35 - ET_MixtureModel class~~

~~**Description:** Abstract class for representing error representation type in mixture model.~~
~~**Derived From:** RoLo Error Type~~

~~Table 36 - ET_LinearMixtureModel class~~

~~**Description:** Error representation as a linear mixture of probability distributions.~~
~~**Derived From:** ET_Mixture Model~~

Table 37 - ET_MixtureOfGaussian class

Description: Error representation as a mixture of normal distributions.	
Derived From: ET_LinearMixtureModel	

RoLo Error

RoLo errors are objects for holding error information in different representations. Here we define some frequently used forms. Users may extend these classes to implement their own RoLo error containers, accompanied with appropriate RoLo error type definitionsHere classes for holding error information in different representations are described.

12928, 13185, 13319, 13398, 14005

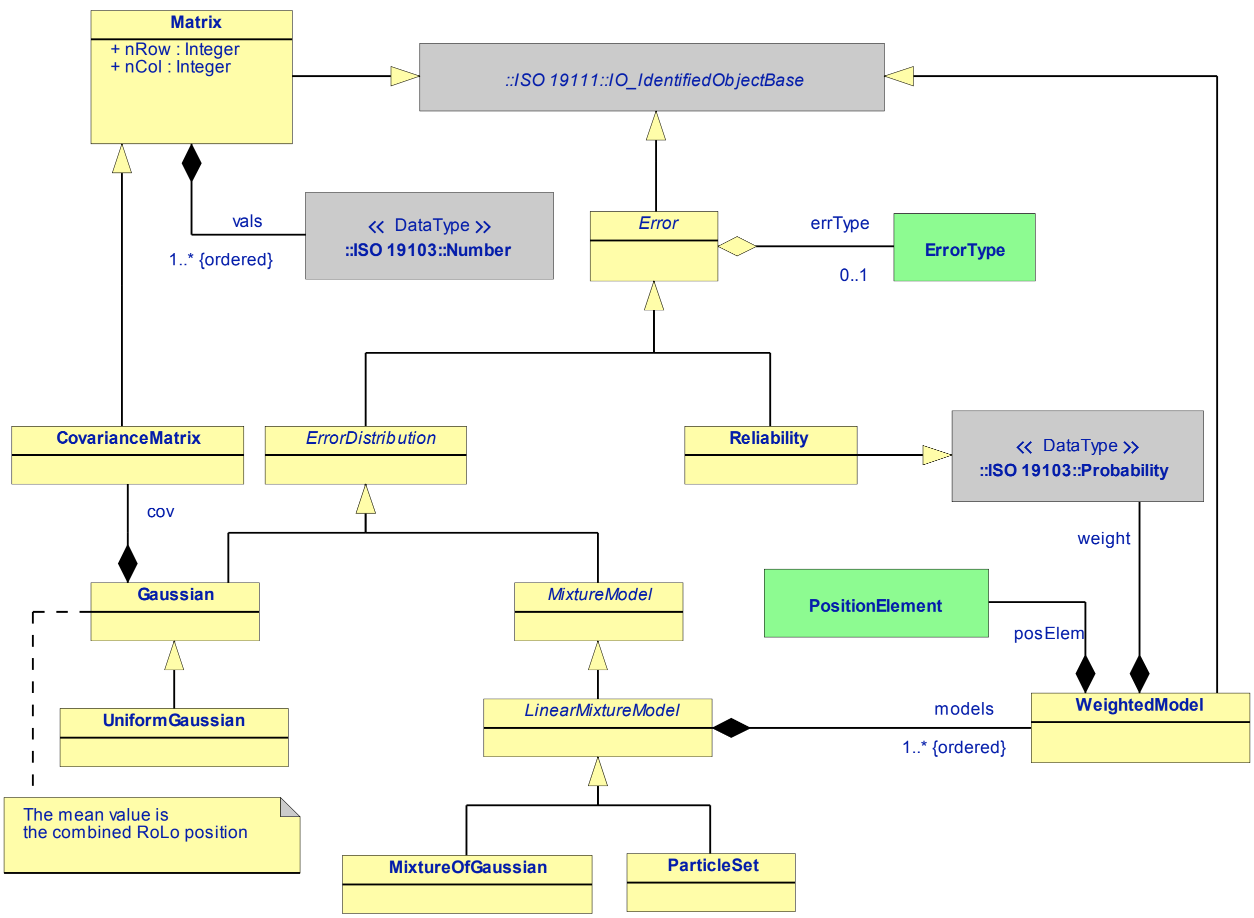


Figure 13: RoLo Error

13441, 13998

Table 38 - Error class

Description: Base abstract class for holding error information.				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes				
errType	ErrorType	O	1	Reference to the RoLo error type indicating how this error information is represented.

Table 39 - Reliability class

Description: Reliability value. The derived attribute 'errType' shall be ET_Reliability.
Derived From: Error, Probability [ISO19103]
~~**Description:** Reliability value. The derived attribute 'errorType' shall be ET_Reliability.~~
~~**Derived From:** RoLo Error, Probability [ISO19103]~~

13998

Table 40 - Error-Distribution class

Description: Base abstract class for error information represented by a probability distribution.
Derived From: Error
~~**Description:** Base class for error information represented by a probability distribution.~~
~~**Derived From:** RoLo Error~~
~~**Attributes**~~

12940, 13441, 13998

Table 41 - Matrix class

Description: N-dimensional matrix.				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes				
nRow	Integer	M	1	Number of matrix rows. The value of attribute 'nRow' should be a positive integer.
nCol	Integer	M	1	Number of matrix columns. The value of attribute 'nCol' should be a positive integer.
vals	Number [ISO19103]	M	N ord	Value elements of the matrix.
~~**Description:** N-dimensional matrix.~~				
~~**Derived From:** IO_IdentifiedObjectBase [ISO19111]~~				
~~**Attributes**~~				
~~numRow~~	~~Integer~~	~~M~~	~~1~~	~~Number of matrix rows.~~
~~numColumn~~	~~Integer~~	~~M~~	~~1~~	~~Number of matrix columns.~~
~~values~~	~~Number~~	~~M~~	~~1~~	~~Value elements of the matrix.~~

13398

Table 42 - Covariance Matrix class

Description: An n-dimensional matrix describing covariance.
Derived From: Matrix
Note: This shall represent a square matrix where nRow = nCol.
~~**Description:** An N-dimensional matrix describing covariance.~~
~~**Derived From:** Matrix~~

13441

Table 43 - Gaussian class

Description: Error represented by an n-dimensional normal distribution. The mean value is denoted by the accompanying RoLo position. The derived attribute 'errType' shall be ET_Gaussian.				
Derived From: ErrorDistribution				
Attributes				
cov	CovarianceMatrix	M	1	Indicates the covariance for the normal distribution.
~~**Description:** Error represented by an n-dimensional single normal distribution. The mean value is the accompanying RoLo Position in the RoLo Element. The derived attribute 'errorType' shall be ET_Gaussian.~~				
~~**Derived From:** Error Distribution~~				
~~**Attributes**~~				
~~covariance~~	~~Covariance Matrix~~	~~M~~	~~1~~	~~Indicates the covariance for the normal distribution. The dimensions of this matrix should match the dimension of the accompanying RoLo Position.~~

13398

Table 44 - Uniform Gaussian class

Description: Error represented by a uniform normal distribution.				
Derived From: Gaussian				
Note: Dimensions of the cov attribute derived from class Gaussian shall all be equal to 1. That is, nRow = nCol = 1.				
~~**Description:** Error represented by a uniform normal distribution. Mean value is the accompanying RoLo Position in the RoLo Element. The derived attribute 'errorType' shall be ET_UniformGaussian.~~				
~~**Derived From:** Error Distribution~~				
~~**Attributes**~~				
~~stddev~~	~~Real~~	~~M~~	~~1~~	~~Standard deviation of the distribution common to every coordinate axis.~~

13185

Table 45 - ParticleSet class

Description: Error represented by a set of particles. As for the 'models' attribute derived from LinearMixtureModel class, the 'posElem' attribute shall either have no 'err' attribute or have an RoLo error like an impulse response (such as a Gaussian distribution with zero standard deviation). Normally, this is used for representing distributions by Monte Carlo approximation, where distributions are approximated by a finite number of random samplings. The derived attribute 'errType' shall be ET_ParticleSet.
Derived From: LinearMixtureModel
~~**Description:** Representation of a particle.~~

~~**Derived From:** IO_IdentifiedObjectBase [ISO19111]~~				
~~**Attributes**~~				
~~pos~~	~~RoLo Position~~	~~M~~	~~1~~	~~Position represented by this particle.~~
~~likelihood~~	~~Probability~~	~~M~~	~~1~~	~~Likelihood of this particle.~~

Table 46 - MixtureModel~~Particle Set~~ class

Description: Abstract base class for representing an error distribution by means of mixture of probability distributions.				
Derived From: ErrorDistribution				
~~**Description:** Error represented by a set of particles. The expected value is the accompanying RoLo Position in the RoLo Element. Normally, this is used for representing distributions by Monte Carlo approximation, where distributions are approximated by a finite number of random samplings. The derived attribute 'errorType' shall be ET_ParticleSet.~~				
~~**Derived From:** Error Distribution~~				
~~**Attributes**~~				
~~numParticles~~	~~Integer~~	~~O~~	~~1~~	~~Number of particles held.~~
~~particles~~	~~RoLo Element~~	~~M~~	~~N~~	~~Particles used to describe the probability distribution.~~

13185, 13998

Table 47 - Weighted~~Mixture~~ Model class

Description: A distribution with a weight. Recall that a PositionElement object can be interpreted to represent a probability distribution. Its 'pos' attribute is treated as the expected coordinate value and its 'err' attribute as the shape of distribution. Thus, in this class the combination of 'weight' and 'posElem' attributes denotes a weighted distribution.				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes				
weight	Probability [ISO19103]	M	1	Weight of this distribution.
posElem	PositionElement	M	1	Expected position for the distribution.
~~**Description:** Abstract class for representing an error distribution by means of mixture of probability distributions.~~				
~~**Derived From:** Error Distribution~~				

13185, 13437

Table 48 - LinearMixtureModel~~Weighted Distribution~~ class

Description: A distribution represented by a linear mixture of probability distributions. The derived attribute 'errType' shall be ET_LinearMixtureModel.				
Derived From: MixtureModel				
Attributes				
models	WeightedModel	M	N ord	List of weighted models to be combined.
~~**Description:** A distribution with a weight.~~				
~~**Derived From:** IO_IdentifiedObjectBase~~				
~~**Attributes**~~				
~~weight~~	~~Probability~~	~~M~~	~~1~~	~~Weight of this distribution.~~

13185

~~pos~~	~~RoLo Position~~	~~M~~	~~1~~	~~Expected position for the distribution.~~
~~dist~~	~~Error Distribution~~	~~M~~	~~1~~	~~Distribution.~~

13185, 13998

Table 49 - MixtureOfGaussian~~Linear Mixture Model~~ class

Description: A distribution represented by a linear mixture of Gaussian distributions. The derived attribute 'errType' shall be ET_MixtureOfGaussian. The models attribute derived from LinearMixtureModel shall have a 'posElem' attribute whose 'err' attribute is restricted to be an instance of Gaussian class.				
Derived From: LinearMixtureModel				
~~**Description:** Error distribution represented by a linear mixture of probability distributions. The derived attribute 'errorType' shall be ET_LinearMixtureModel.~~				
~~**Derived From:** Mixture Model~~				
~~**Attributes**~~				
~~numDists~~	~~Integer~~	~~O~~	~~1~~	~~Number of distributions to be mixed.~~
~~dists~~	~~Weighted Distribution~~	~~M~~	~~N~~	~~List of error distributions to be combined.~~

~~**Table 50 - Mixture of Gaussian class**~~

~~**Description:** Error distribution represented by a mixture of normal distributions. The derived attribute 'errorType' shall be ET_MixtureOfGaussian. The error distributions used for mixture shall be instances of Gaussian class.~~
~~**Derived From:** Linear Mixture Model~~

8.3.5 Robotic Localization Architecture

The ***Robotic Localization (RoLo) Architecture*** defined here is a unified framework for organizing and representing complex data set required in robotic localization. Similar to the relation between GIS location data and coordinate reference system, two sets of structures are defined here.

13116, 13441

1) Classes for holding the localization results (Data, Element and~~RoLo Data, RoLo Element and RoLo~~ Position)

2) Classes for describing the structure or the meaning of localization results (DataSpecification, Element~~RoLo Architecture, RoLo Element~~ Specification)

13116, 13441

These two sets of classes are in relation similar to that between GIS position data and coordinate reference systems: the latter describes the structure and meaning of the former. The RoLo element and RoLo element specification pair binds the main localization data element to error information. The RoLo data and RoLo data specification pair defines the structure and relation among a set of RoLo elements that forms a~~Element and RoLo Element Specification pair binds the main localization data element to error information. The RoLo Data and RoLo Architecture pair defines the structure and relation among a set of RoLo Elements that forms the~~ complete robotic localization results.

Normally, error information is combined with one main localization element. However, in certain cases, there is a need to hold an integrated error among multiple location data. For example, in a typical Kalman filter usage, multiple main location information such as spatial position and velocity are used to form a state vector. When the elements of the state vector are not independent, which is the usual case, the corresponding error, the covariance matrix, is

13112, 13441

related to multiple main elements. In such case, the ErrorElementSpecificaion (derived from ElementSpecification class) specifies which main information slot the error is related to, and the actual error data is contained by the ErrorElement class (derived from Element class) instances. Figure 14 shows a sample data structure and corresponding object diagram ~~Element Specification (derived from RoLo Element class) instance specifies which main information slot the error is related to, and the actual error data is contained by the Error Element class (derived from RoLo Element class) instances~~.

13145

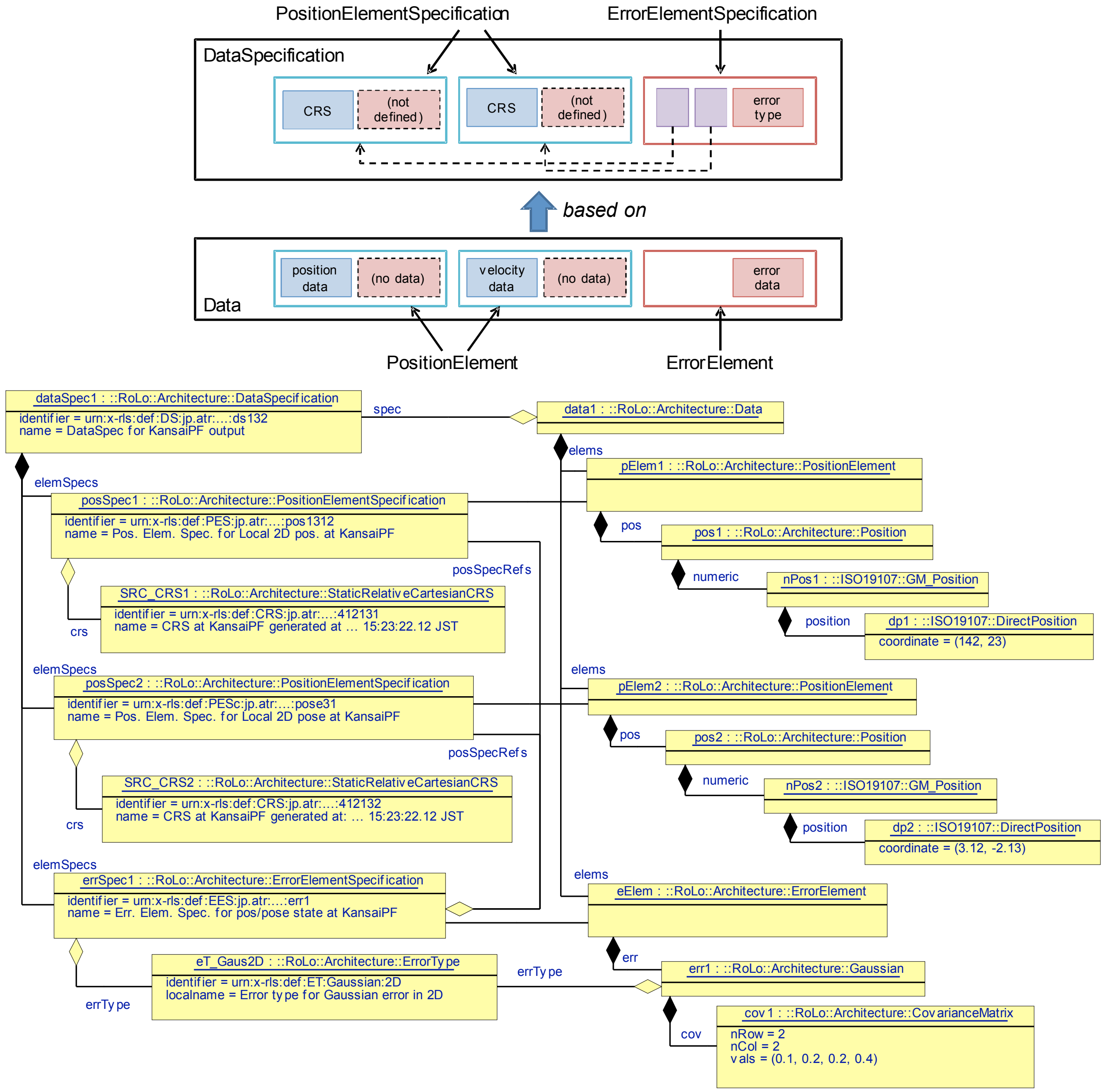


Figure 14: Representation of error information related to multiple localization data

12929, 13116, 13129, 13187, 13998, 14004

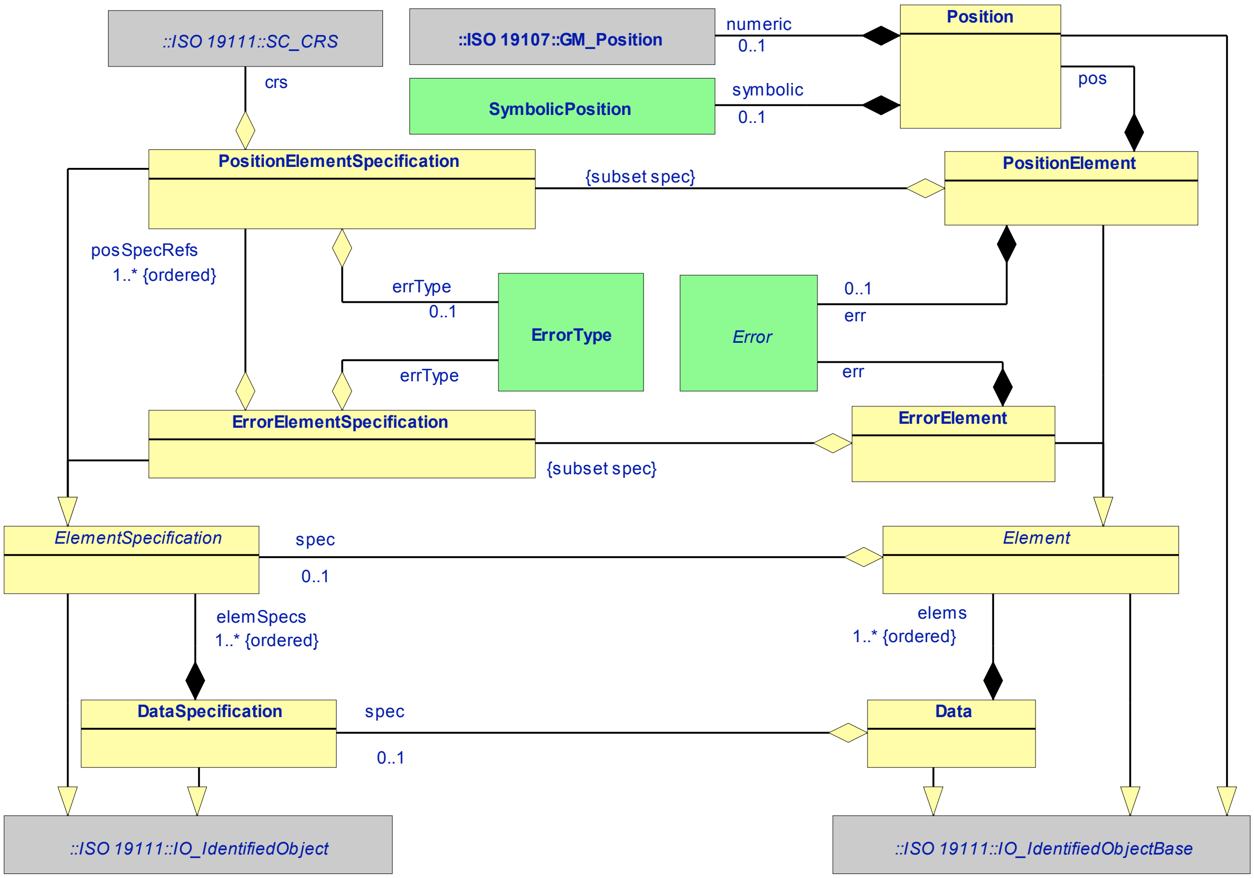


Figure 15: RoLo Architecture

13998

Table 51 - Position class

Description: Data container for localization results without error information. This is formed as a union of SymbolicPosition class and GM_Position [ISO19107] class. The former is a container for symbolic symbols such as identity information, and the latter contains numerical data such as spatial coordinate values.				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes:				
symbolic	SymbolicPosition	C	1	Symbolic data container
numeric	GM_Position [ISO19107]	C	1	Numeric data container.
Condition: One and only one of the choices shall be chosen.				

13129, 13441

Table 52 - ElementSpecification class

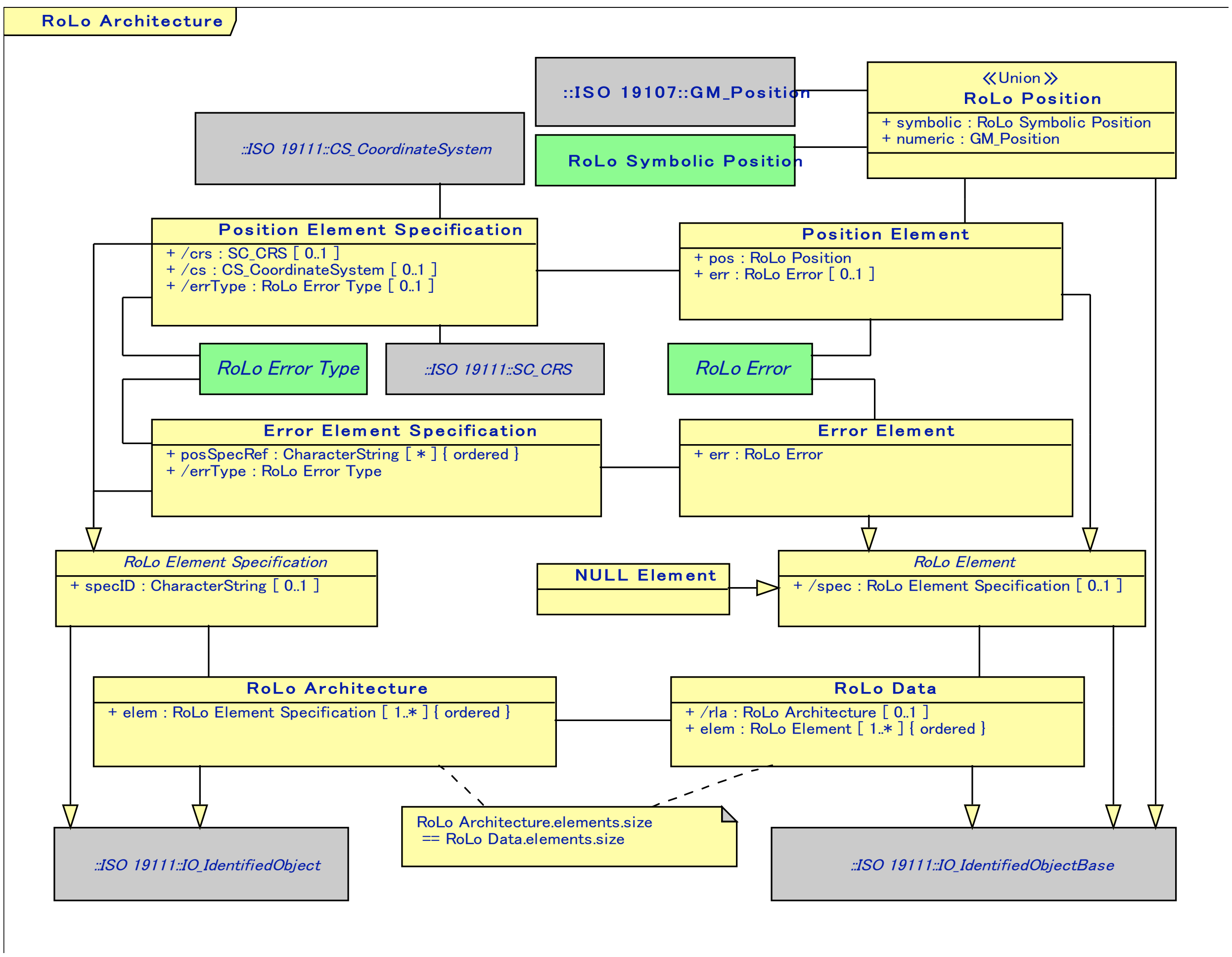
Description: Base abstract class for holding structural definition for RoLo elements. Instances of this class contain meta-level information on what kind of data each RoLo element holds.
Derived From: IO_IdentifiedObject [ISO19111]

~~Figure 16 - Representation of error information related to multiple localization data~~

13187, 13441

Table 53 - PositionElementSpecification class

Description: Specification holder for RoLo position elements.				
Derived From: ElementSpecification				
Attributes:				
crs	SC_CRS [ISO19111]	M	1	Reference to a coordinate reference system that the 'pos' attribute in RoLo position element is based on.
errType	ErrorType	O	1	Reference to a RoLo error type. Specifies the type of 'err' attribute in RoLo position elements. If this attribute is omitted, RoLo position elements related with this instance shall not contain error information.



~~Figure 17 - RoLo Architecture~~

13129, 13441

Table 54 - ErrorElementSpecifica~~RoLo Posi~~tion class

Description: Definition holder for RoLo error elements.				
Derived From: ElementSpecification				
Attributes:				
posSpecRefs	PositionElementSpecification	M	N ord	An ordered list of references to RoLo position element specifications showing which positional data the RoLo error contained in the RoLo error element is related to. The referred RoLo position element specifications shall be contained in the same RoLo data specification as this class instance.
errType	ErrorType	M	1	Reference to a RoLo error type. Specifies the type of 'err' attribute in RoLo error elements.

~~**Description:** Data container for localization results without error information. This is formed as a union of RoLo Symbolic Position class and GM_Position [ISO19107] class. The former is a container for symbolic symbols such as identity information, and the latter contains numerical data such as spatial coordinate values.~~				
~~**Derived From:** IO_IdentifiedObjectBase [ISO19111]~~				
~~**Attributes:**~~				
~~symbolic~~	~~RoLo Symbolic Position~~	~~C~~	~~1~~	~~Symbolic data container~~
~~numeric~~	~~GM_Position~~	~~C~~	~~1~~	~~Numeric data container.~~
~~**Condition:** One and only one of the choice shall be chosen.~~				

13441

Table 55 - Element class

Description: Base abstract class for RoLo elements which holds the binding between the main positional data and the RoLo error.				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes:				
spec	ElementSpecification	O	1	Reference to RoLo element specification that this element is based on.

12929, 13441

Table 56 - PositionElement class

Description: Data container of each localization result by combining the main positional data and the accompanying RoLo error.				
Derived From: Element				
pos	Position	M	1	The main information.
err	Error	O	1	RoLo error information related to the 'pos' attribute of the same instance. If the RoLo position element specification referred related with this instance does not hold an 'errType' attribute, this attribute shall be omitted.
Note: Values for the attribute 'spec' which is derived from parent class shall be limited to instances of PositionElementSpecification or its inherited classes.				

12929, 13441

Table 57 - ErrorElement class

Description: Data container of error information that is related to multiple positional data in the same RoLo data. RoLo position elements related with this error information are specified in the referenced RoLo error element specification.				
Derived From: Element				
Attributes:				
err	Error	M	1	RoLo error bound with the specified RoLo position elements.
Note: Values for the attribute 'spec' which is derived from parent class shall be limited to instances of ErrorElementSpecification or its inherited classes.				

12940, 13116, 13441

Table 58 - DataSpecification class

Description: Specification holder for RoLo data.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes:				
elemSpecs	ElementSpecification	M	N ord	Ordered list of RoLo element specifications that defines the structure of localization result.

12940, 13109, 13116, 13441, 13998

Table 59 - Data class

Description: Data container for the robotic localization result.				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes:				
spec	DataSpecification	O	1	Reference to the corresponding RoLo data specification.
elems	Element	M	N ord	An ordered list of RoLo elements. Numbers, orders and types of the RoLo elements shall match that of the corresponding RoLo data specification.

13187

Don't-Care

In order to handle generic data specifications, specifications may include "don't care" values in their definition. For example, you may want to build a people tracking service which accepts outputs from another RoLo module bound with a camera sensor and performs some calculation. In such case, the coordinate system of the camera sensor output may be fixed but the coordinate reference system and the datum associated with each camera module may differ, depending on the location where the camera is installed. Building such module is impossible in the normal RoLo framework, as each RoLo stream need to clearly specify a set of RoLo data specifications it can accept; you need to specify an infinite list of RoLo data specifications on the input stream ability description.

That's where don't-cares are used. In such cases, you specify a RoLo data specification for the tracking module's input stream ability using a coordinate reference system which uses a don't-care datum (NULLDatum class). This way you can specify only the specification parts you (the

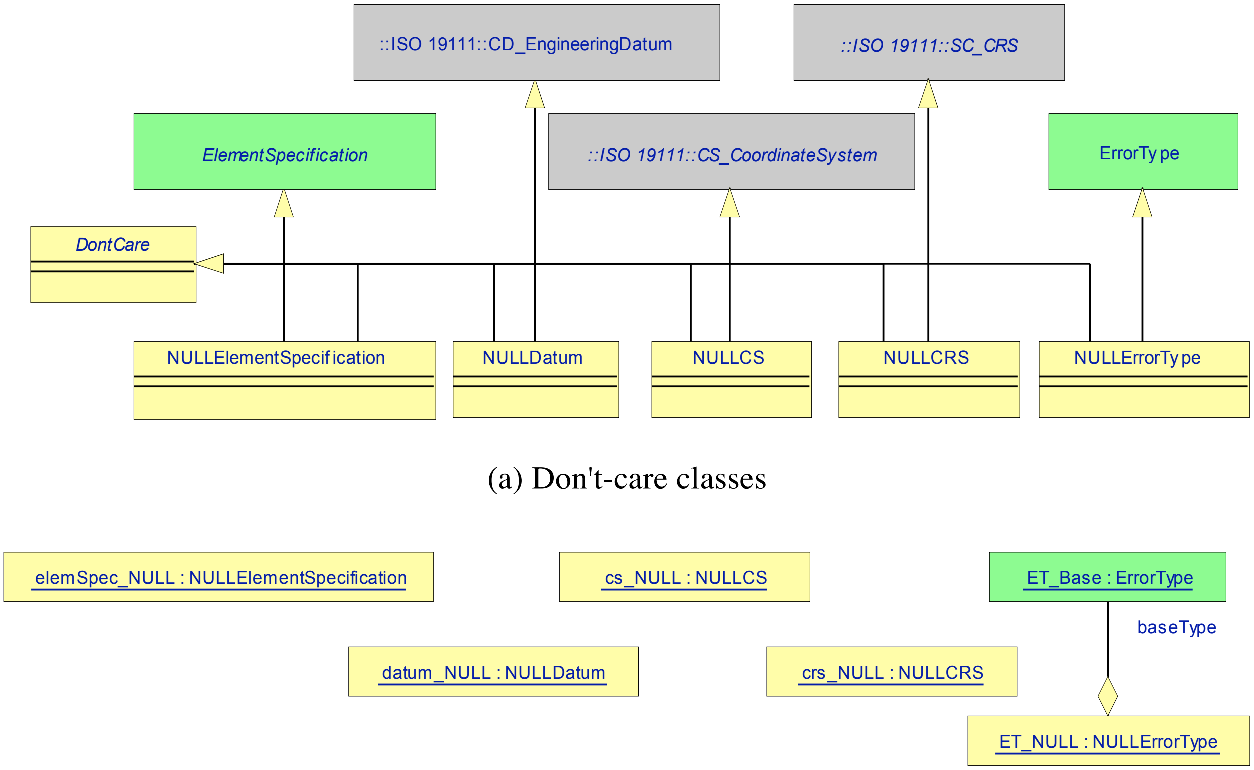
13187 module) is interested, and leave the other parts free. Such is quite a common usage, and so the use of don't-cares will increase the flexibility and usability of the RoLo service. However, this use of don't-care elements may require notice as it may result in high computation cost or ambiguous, useless specifications that break the idea of having specifications for data. Thus, we need some rule to avoid misleading usages. The following describes the rules that shall be followed on using don't-cares:

- When multiple type specifications are associated with a data instance, the specifications shall be consistent with each other.
- Every data instance shall have a complete unified type specification.
- A type specification may include don't-cares for the following attributes:
  - 'elemSpec' in RoLo data specification
  - 'crs' in RoLo position element specification
  - 'errType' in RoLo position element specification or RoLo error element specification
  - 'cs' in SC_CRS [ISO19111]
  - 'datum' in SC_CRS [ISO19111]

  Figure 18 shows the classes and the objects used to indicate don't-care.
- A RoLo stream that is associated with an incomplete data specification should check consistency of each Data passed through the stream.

The last rule means that any RoLo stream that is associated with a complete RoLo data specification may skip checking explicit specification of each RoLo data or its subcomponents passed through itself. Thus, modules equipped with low computation power can avoid unnecessary processing by specifying explicit data specifications as their RoLo input stream ability.

13187



(a) Don't-care classes

(b) Don't-care objects

Figure 18: Don't-care classes and objects

Table 60 - DontCare class

Description: Base abstract class for don't-care classes.
Derived From: (none)

Table 61 - NULLCS~~- RoLo Element Specification~~ class

Description: Don't-care indicator. Used for indicating that this coordinate system shall be ignored.
Derived From: DontCare, CS_CoordinateSystem [ISO19111]

~~**Description:** Base abstract class for holding structural definition for RoLo Element. Instances of this RoLo Element Specification class contain meta-level information on what kind of data each RoLo Element data container holds.~~				
~~**Derived From:** IO_IdentifiedObject [ISO19111]~~				
~~**Attributes:**~~				
~~specID~~	~~CharacterString~~	~~M~~	~~1~~	~~Identifier for each RoLo Element Specification instances within a certain RoLo Architecture instance. This value should be interpreted as a local identifier within the RoLo Architecture instance this RoLo Element Specification instance belongs.~~

Table 62 - Position Element Specification class

Description: Definition holder for Position Element class. Every instance of Position Element Specification class shall hold a reference to either a coordinate reference system or a coordinate system that the positional data belongs to. When the coordinate system reference is specified, it is recommended that Position Element objects based on this specification objects hold information on which coordinate reference system they belong to.				
Derived From: RoLo Element Specification				
Attributes:				
crs	SC_CRS [ISO19111]	C	1	Reference to the Coordinate Reference System instance that the pos attribute in Position Element class belongs to.
cs	CS_CoordinateSystem [ISO19111]	C	1	Reference to the Coordinate System instance that the pos attribute in Position Element class is based on.
errType	ErrorType	O	1	Reference to RoLo Error Type instance. Specifies the type of err attribute in Position Element class. If this attribute does not exist, the Position element related with this instance shall not contain error information.
Condition: One and only one of the listed attributes shall be supplied				

Table 63 - Error Element Specification class

Description: Definition holder for Error Element class.				
Derived From: RoLo Element Specification				
Attributes:				
posSpecRef	CharacterString	M	N ord	Reference to Position Element Specification instance contained in the same RoLo Architecture instance as this class instance. The value of specID attribute in the target Position Element Specification is used. Indicates the Position Element Specification instances that the error information contained in Error Element instance is related with.
errType	ErrorType	M	1	Reference to RoLo Error Type instance. Specifies the type of err attribute in Position Element class.

13187

Table 64 - NULLCRS class

Description: Don't-care indicator. Used for indicating that this coordinate reference system shall be ignored.
Derived From: DontCare, SC_CRS [ISO19111]

Table 65 - NULLDatum class

Description: Don't-care indicator. Used for indicating that this datum shall be ignored.
Derived From: DontCare, CD_EngineeringDatum [ISO19111]

13187 ### Table 66 - NULLErrorType class

Description: Don't-care indicator. Used for indicating that this RoLo error type shall be ignored.	
Derived From: DontCare, ErrorType	

Table 67 - NULLElementSpecification class

Description: Don't-care indicator. Used for indicating that this slot in RoLo element specification shall be ignored.	
Derived From: DontCare, ElementSpecification	

RoLo Data Operation

12930, 13116, 13129, 13143, 13998

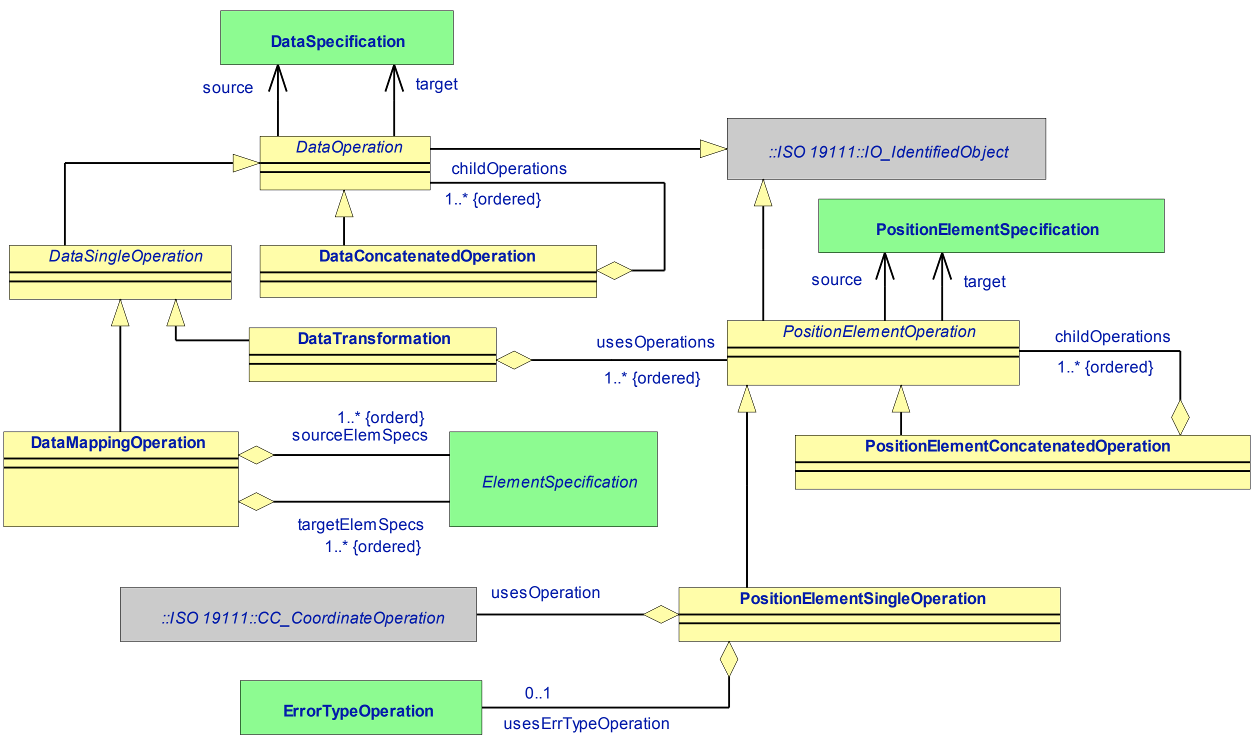


Figure 19: RoLo Data Operation

13313, 13441 ### Table 68 - PositionElementOperation class

Description: Base abstract class for representing operations for transforming data between different RoLo position elements. RoLo position elements are basically composed by RoLo position and RoLo error. As the value of RoLo error is also based on the coordinate reference system where the combined RoLo position is based on, both the main information and the error information shall be transformed at once.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes:				
source	PositionElementSpecification	M	1	Source RoLo position element specification.

target	PositionElementSpecification	M	1	Target RoLo position element specification.

Table 69 - PositionElementConcatenatedOperation class

Table 70 - RoLo Element class

Description: Base abstract class for holding each localization result element, combination of the main information and the accompanying error information.				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes:				
spec	RoLo Element Specification	O	1	Reference to RoLo Element Specification instance that this element is based on.

Table 71 - Position Element class

Description: Data container of each localization result element, combination of the main information and the accompanying error information. This subclass of RoLo Element holds the actual main information. The derived spec attribute shall be a reference to Position Element Specification.				
Derived From: RoLo Element				
Attributes:				
pos	RoLo Position	M	1	The main information.
err	RoLo Error	O	1	Error information related to the pos attribute of the same instance.If the Position Element Specification related with this Position Element instance does not contain errType attribute, this err attribute shall not be contained.

Table 72 - Error Element class

Description: Data container of error information that is related to multiple number of main information in the same RoLo Data. The elements related with this error information are defined in the accompanying Error Element Specification instance. The derived spec attribute shall be a reference to Error Element Specification.				
Derived From: RoLo Element				
Attributes:				
err	RoLo Error	M	1	Reference to RoLo Error Type instance. Specifies the type of err attribute in Position Element class.

Table 73 - NULL Element class

Description: Don't-care indicator. Used for indicating that this slot in RoLo Element Specification shall not be processed. The spec attribute value shall be ignored.
Derived From: RoLo Element

~~Table 74 - RoLo Architecture class~~

~~**Description:** Definition holder for the robotic localization result.~~				
~~**Derived From:** IO_IdentifiedObject [ISO19111]~~				
~~**Attributes:**~~				
~~elemSpec~~	~~RoLo Element Specification~~	~~M~~	~~N~~	~~Ordered list for the RoLo Element Specifications that defines the structure of the localization result data.~~

~~Table 75 - RoLo Data class~~

~~**Description:** Data container for the robotic localization result.~~				
~~**Derived From:** IO_IdentifiedObjectBase [ISO19111]~~				
~~**Attributes:**~~				
~~rla~~	~~RoLo Architecture~~	~~O~~	~~1~~	~~Reference to the correspoinding RoLo Architecture.~~
~~elem~~	~~RoLo Element~~	~~M~~	~~N~~	~~Ordered list for the RoLo Element Specifications that defines the structure of the localization result data. Numbers of the elements shall match that of corresponding RoLo Architecture.~~

~~Operations~~

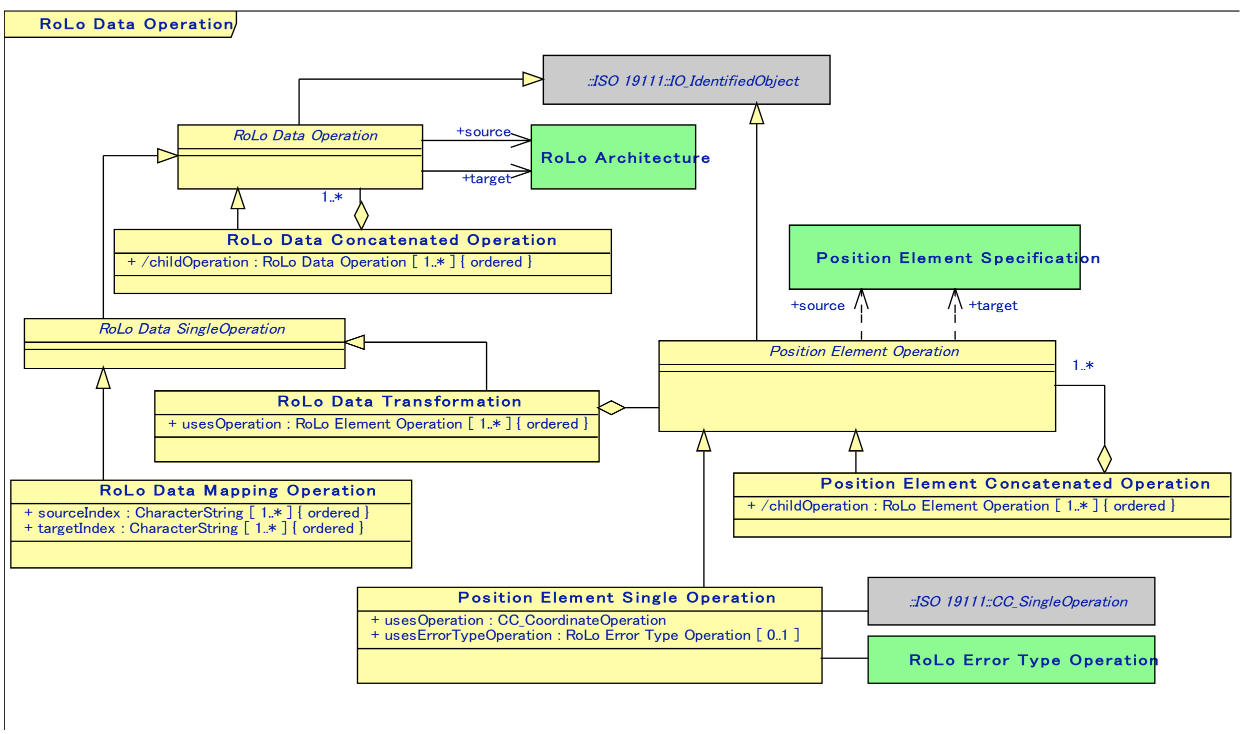


Figure 20 - RoLo Data Operation

Table 76 - Position Element Operation class

Description: Base abstract class for representing operations for transforming data between different RoLo Elements. RoLo Elements are basically composed by RoLo Position and RoLo Error. As the value of RoLo Error is also based on the coordinate reference system where the combined RoLo Position is based on, both the main information and the error information shall be transformed at once.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes:				
source	Position Element Specification	M	1	Source Position Element Specification.
target	Position Element Specification	M	1	Target Position Element Specification.

13441 Table 77 - Position Element Concatenated Operation class

Description: Concatenation of multiple PositionElementOperation instances.				
Derived From: PositionElementOperation				
Attributes:				
childOperations	PositionElementOperation	M	N ord	Ordered list of PositionElementOperation to be applied. Target RoLo position element specification and source RoLo position element specification for succeeding operations shall match.

13441 Table 78 - PositionElementSingle Element Single Operation class

Description: Definition of an operation for transforming or converting data between different RoLo position element specifications. The main information is processed by the CC_CoordinateOperation [ISO19111], and the error information should also be transformed.				
Derived From: PositionElementOperation				
Attributes:				
usesOperation	CC_CoordinateOperation [ISO19111]	M	1	Operation to be used for transforming the main localization data. This operation may also be utilized to transform the accompanying RoLo error.
usesErrTypeOperation	ErrorTypeOperation	O	1	Operation to be used for converting the type of the RoLo error part. If no error type conversion is necessary, this part may be omitted.

13116, 13441 Table 79 - DataRoLo Data Operation class

Description: Base abstract class for representing operations for transforming data between different RoLo data specifications. The main purpose of this operation is to transform or to convert RoLo data that contains RoLo error element. RoLo data which contains RoLo error element need to know about how other elements within the same RoLo data specification are operated. Instances of this class perform necessary operations for RoLo error elements, alongside the operations for RoLo position elements.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes:				
source	DataSpecification	M	1	Reference to the originate RoLo data specification.
target	DataSpecification	M	1	Reference to the target RoLo data specification.

~~**Description:** Base abstract class for representing operations for transforming data between different RoLo Architectures. The main purpose of this operation is to transform or to convert RoLo Data that contains Error Element. RoLo Data which contains Error Element need to know about how other elements within the same RoLo Architecture are operated. RoLo Data Operation performs necessary operations for Error Elements, alongside the operations for Position Elements.~~				
~~**Derived From:** IO_IdentifiedObject [ISO19111]~~				
~~**Attributes:**~~				
~~source~~	~~RoLo Architecture~~	~~M~~	~~1~~	~~Reference to the originate RoLo Architecture.~~
~~target~~	~~RoLo Architecture~~	~~M~~	~~1~~	~~Reference to the target RoLo Architecture.~~

12940, 13116, 13441

Table 80 - DataConcatenated~~RoLo Data Concatenated~~ Operation class

Description: Concatenation of multiple RoLo data operations.				
Derived From: DataOperation				
Attributes:				
childOperations	DataOperation	M	N ord	Ordered list of RoLo data operation to be applied. Target RoLo data specification and source RoLo data specification for succeeding operations shall match.

~~Table 81 - RoLo Data Single Operation class~~

~~**Description:** Abstract class for representing an operation for transforming data between different RoLo Architectures.~~
~~**Derived From:** RoLo Architecture Operation~~

12940, 13116, 13441

Table 82 - DataSingleOper~~RoLo Data Transform~~ation class

Description: Abstract class for representing an operation for transforming data between different RoLo data specifications.				
Derived From: DataOperation				
~~**Description:** Definition of an operation for transforming data between different RoLo Architecture.~~				
~~**Derived From:** RoLo Architecture Single Operation~~				
~~**Attributes:**~~				
~~usesOperation~~	~~RoLo Element Operation~~	~~M~~	~~N ord~~	~~Operations used for each of the RoLo Element Specification in this RoLo Architecture. The number of RoLo Element Specifications in this RoLo Architecture and that of usesOperation attribute shall match. The operation defined here is applied to each of the RoLo Elements in the order the corresponding RoLo Element Specifications are defined.~~

12940, 13116, 13320, 13441

Table 83 - DataTransform~~RoLo Data Mapping Oper~~ation class

Description: Definition of an operation for transforming data between different RoLo data specification.				
Derived From: DataSingleOperation				
Attributes:				
usesOperations	PositionElementOperation	M	N	Operations used for each of the RoLo position element specification in

12940, 13116, 13441

			ord	the RoLo data specification. The number of RoLo position element specifications in this RoLo data specification and that of ‘usesOperation’ attribute shall match. The operation defined here is applied to each of the RoLo position elements in the order the corresponding RoLo position element specifications are defined.
Description: Definition of an operation for transforming data between different RoLo Architecture, that simply maps elements in the source RoLo Architecture to elements in the target RoLo Architecture. Only the structures of the RoLo Elements are altered, and the data content itself are not changed. With Error Elements, the reference to the Position Elements shall be modified appropriately. The two attributes contained are lists of RoLo Element Specification indices in source and target RoLo Architectures that defines how the mapping is to be performed.				
Derived From: RoLo Architecture Single Operation				
Attributes:				
soruceIndex	CharacterString	M	N ord	Ordered list of RoLo Element Specification relative IDs within the source RoLo Architecture, which is to be mapped to the RoLo Element Specification in the target RoLo Architecture represented by the targetIndex attribute value at same position. The numbers of sourceIndex attribute shall match that of targetIndex attribute
targetIndex	CharacterString	M	N ord	Ordered list of RoLo Element Specification relative Ids within the target RoLo Architecture.

12940, 13116, 13129

Table 84 - DataMappingOperation class

Description: Definition of an operation for transforming data between different RoLo data specifications that simply maps elements in the source RoLo data specification to elements in the target RoLo data specification. Only the structures of the RoLo elements are altered, and the data content itself are not changed. With RoLo error elements, the reference to the RoLo position elements shall be modified appropriately. The two attributes contained are lists of references to RoLo element specifications in source and target RoLo data specifications that defines how the mapping is to be performed.				
Derived From: DataSingleOperation				
Attributes:				
sourceElemSpecs	ElementSpecification	M	N ord	Ordered list of RoLo element specification references within the source RoLo data specification which is to be mapped to the RoLo element specification in the target RoLo data specification represented by the ‘targetElemSpecs’ attribute value at the same position. The numbers of 'sourceElemSpecs’ attribute shall match that of ‘targetElemSpec’ attribute.
targetElemSpecs	ElementSpecification	M	N ord	Ordered list of RoLo element specification references within the target RoLo data specification.

12916

8.4 DataFormat Package

8.5 Data Formats

When exchanging information amongst modules, knowledge on data structures is not enough. We need to specify the actual data representation format exchanged amongst modules.

12931, 13116

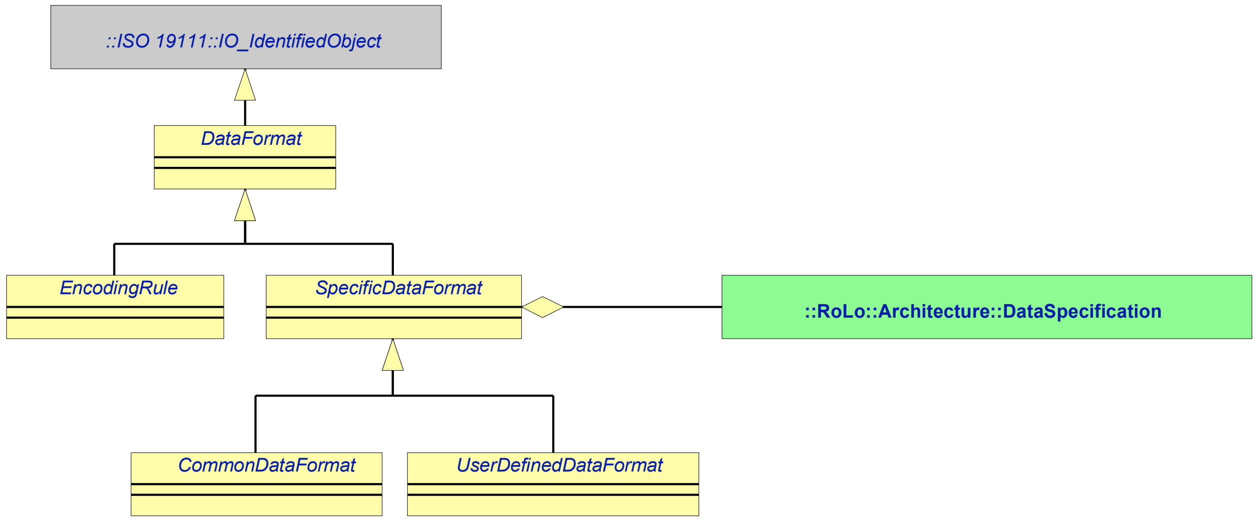


Figure 21: RoLo Data Format

Table 85 - DataFormat class

Description: Base abstract class for data format definitions.	
Derived From: IO_IdentifiedObject [ISO19111]	

12940, 13116, 13441

Table 86 - EncodingRule~~Specific Data Format~~ class

Description: Base abstract class for encoding rules. Encoding rule denotes some systematic mean that can determine the data format from corresponding data structure (i.e. RoLo data specification). Packed Encoding Rule [PER] is an example of encoding rule. This is a reserved class for future extension.				
Derived From: DataFormat				
~~**Description:** Abstract class for data formats where format description is tightly coupled with data structure. This is in contrast with the Encoding Rule class, where data formatting rules are independent to data structure definitions.~~				
~~**Derived From:** RoLo Data Format~~				
~~**Attributes:**~~				
~~rla~~	~~RoLo Architecture~~	~~M~~	~~1~~	~~Specifies data architecture that this data format can handle.~~

12916, 13116

Table 87 - SpecificDataFormat class

Description: Abstract class for data formats where format description is tightly coupled with data structure. This is in contrast with the EncodingRule class, where data formatting rules are independent to data structure definitions.				
Derived From: DataFormat				
Attributes:				
dataSpec	DataSpecification (RoLo::Architecture)	M	1	Specifies a RoLo data specification that this data format can handle.

Table 88 - UserDefinedData ~~Defined Data~~ Format class

Description: Abstract class for user-defined, non-common data formats.
Derived From: SpecificDataFormat
~~**Description:** Abstract class for user-defined, non-common data formats.~~
~~**Derived From:** Specific Data Format~~

Table 89 - CommonData ~~Data~~ Format class

Description: Abstract class for denoting Common Data Formats.
Derived From: SpecificDataFormat
~~**Description:** Abstract class for denoting Common Data Formats.~~
~~**Derived From:** Specific Data Format~~

13120 8.5.1 Common data format

This specification allows a wide range of data formats for keeping compatibility to widely used data formats. This specification, however, defines three common data formats each with two different RoLo data specifications, representing location information in order to provide interoperability between modules which have lack of computing resources. Every module in RoLo s~~, representing location information in order to provide interoperability between modules which have lack of computing resources. Every module in Robotic Localization S~~ervice shall support at least one of these common data formats in order to transmit location information to enhance inter-module connectability as much as possible.

In this specification, depending on the coordinate systems to refer the position and the methods to specify the orientation, the common data format is represented by one of the six types, Type I-1, I-2, II-1, II-2, III-1, and III-2 as bellows:

Type I-1

Table 90 - Common data format type I-1 (Cartesian Coordinate System, xyz-Euler Angle Representation)

Parameter	Format of value	Value type	Unit
Position	[*x*, *y*, *z*]	Real, Real, Real	meter, meter, meter
Orientation	[α, β, γ]	Real, Real, Real	radian, radian, radian
Timestamp	POSIX time	Integer, Integer	second, nanosecond
ID	--	Integer	--

13120 **Type I-2**

Table 91 - Common data format type I-2 (Cartesian Coordinate System, XYZ-Euler Angle Representation)

Parameter	Format of value	Value type	Unit
Position	[x, y, z]	Real, Real, Real	meter, meter, meter
Orientation	[yaw α, pitch β, roll γ]	Real, Real, Real	radian, radian, radian
Timestamp	POSIX time	Integer, Integer	second, nanosecond
ID	--	Integer	--

Type II-1

Table 92 - Common data format type II-1 (Spherical Coordinate System, xyz-Euler Angle Representation)

Parameter	Format of value	Value type	Unit
Position	[r, θ, φ]	Real, Real, Real	meter, radian, radian
Orientation	[α, β, γ]	Real, Real, Real	radian, radian, radian
Timestamp	POSIX time	Integer, Integer	second, nanosecond
ID	--	Integer	--

Type II-2

Table 93 - Common data format type II-2 (Spherical Coordinate System, XYZ-Euler Angle Representation)

Parameter	Format of value	Value type	Unit
Position	[r, θ, φ]	Real, Real, Real	meter, radian, radian
Orientation	[yaw α, pitch β, roll γ]	Real, Real, Real	radian, radian, radian
Timestamp	POSIX time	Integer, Integer	second, nanosecond
ID	--	Integer	--

Type III-1

Table 94 - Common data format type III-1 (Geodetic Coordinate System, xyz-Euler Angle Representation)

Parameter	Format of value	Value type	Unit
Position	[latitude φ, longitude λ, height h]	Real, Real, Real	degree, degree, meter
Orientation	[α, β, γ]	Real, Real, Real	radian, radian, radian
Timestamp	POSIX time	Integer, Integer	second, nanosecond
ID	--	Integer	--

Type III-2

Table 95 - Common data format type III-2 (Geodetic Coordinate System, XYZ-Euler Angle Representation)

Parameter	Format of value	Value type	Unit
Position	[latitude φ, longitude λ, height h]	Real, Real, Real	degree, degree, meter
Orientation	[yaw α, pitch β, roll γ]	Real, Real, Real	radian, radian, radian
Timestamp	POSIX time	Integer, Integer	second, nanosecond
ID	--	Integer	--

Each type of the common data formats includes four parameters as follows:

- ***Position*** – specifies the coordinate value in a Cartesian coordinate system for Type I-1 and I-2, in a

13120 spherical coordinate system for Type II-1 and II-2, and in a geodetic coordinate system for Type III-1 and III-2. (See Figure 22 and its explanation for details).

- ***Orientation*** – specifies sequential three rotations by each axis in a right-handed 3-dimensional Cartesian coordinate system defined by a so-called xyz-Euler Angle representation for Type I-1, II-1, and III-1 (See Figure 23 and its explanation for details), and a so-called XYZ-Euler Angle representation (commonly called yaw-pitch-roll rotation) for I-2, II-2, and III-2. (See Figure 24 and its explanation for details).

- ***Timestamp*** – specifies time at occurring measurement for current position and orientation. It is compatible to POSIX time which is the time elapsed since midnight Coordinated Universal Time (UTC) of January 1, 1970. A timestamp consists of two integers of elapsed seconds and nanoseconds which is compatible to standard UNIX C time_t data structure.

Parameter	Format of value	Value type	Unit
Position	[x,y,z]	Real	meter
Orientation	[θ,ϕ,ψ] (roll, pitch, yaw)	Real	degree
Timestamp	POSIX time	Real	millisecond
ID	--	Integer	--

~~The type I format delivers location information based on Cartesian coordinate system. It includes four parameters.~~

- ~~***Position*** – specifies the coordinate value in a right-handed 3-dimensional Cartesian coordinate system.~~

- ~~***Orientation*** – specifies the rotation along each axis in a right-handed 3-dimensional Cartesian coordinate system. The unit of this parameter is radians. The order of orientation is the rotation around x-axis, the rotation around y-axis, and the rotation around z-axis of 3-dimentional right-handed Cartesian coordinate system; roll, pitch, and yaw are conventional terms.~~

- ~~***Timestamp*** – specifies time at occurring measurement for current position and orientation. The format is compatible to POSIX time which is the number of second elapsed since midnight Coordinated Universal Time (UTC) of January 1, 1970. The resolution of timestamp is milliseconds.~~

- ***ID*** – specifies the identifier of current location information for robots and related entities.

The coordinate values of position information in the common data format in Table 90-95 are defined respectively by three different coordinate systems: Cartesian coordinate, spherical coordinate and geodetic coordinate system as shown in Figure 22.~~ype II~~

13120

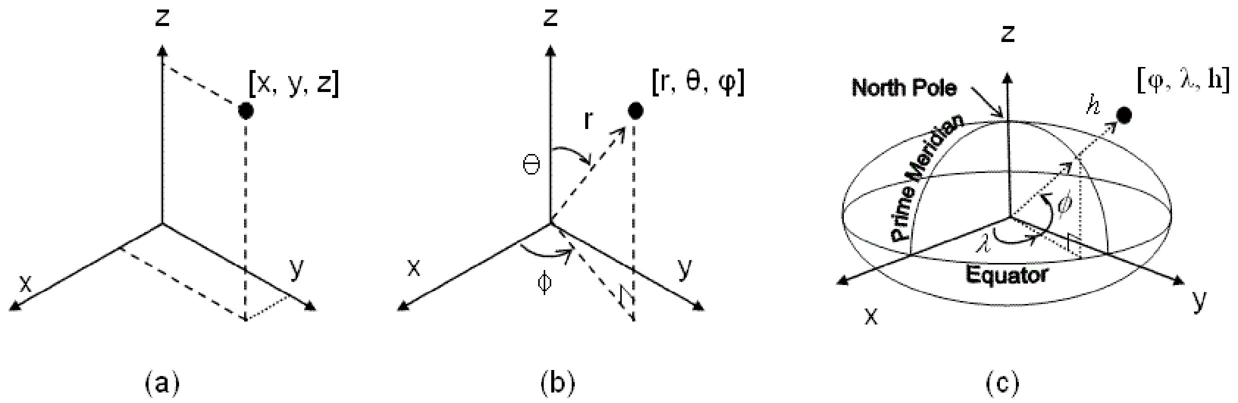


Figure 22: Definition of a position and reference coordinate systems used in the common data format: (a) Cartesian coordinate system for Type I-1 and I-2, (b) spherical coordinate system for Type II-1 and II-2, (c) geodetic coordinate system for Type III-1 and III-2.

Generally, a 3 by 3 matrix is commonly used in robotics to calculate consecutive rotations of a coordinate system or to specify the orientation of a coordinate system respective to a reference coordinate system. However, it is not easy for a human to interpret an orientation by the matrix that contains 9 numbers. Due to the reason, common data formats in this specification use so-called Euler angles that specify the orientation of a coordinate system by a sequence of three rotations that take place about an axis of the coordinate system.

To specify the rotation of the coordinate system, a fixed right-hand Cartesian coordinate system is denoted in lower case (x, y, z) and a rotating right-hand Cartesian coordinate system is denoted in upper case letters (X, Y, Z). Depending on the order of sequential rotations of two coordinate systems, the Euler angle representation can be defined in several ways. In this specification, two most popular Euler angle representations are used: in this specification, the first representation is called xyz-Euler angle representation and used for Common Data Format I-1, II-1, III-1 and the second representation is called XYZ-Euler angle representation and used for Common Data Format I-2, II-2, III-2.

The xyz-Euler angle representation is defined as follows:

1) Start with the rotating XYZ coordinate system coinciding with the fixed xyz coordinate system.

2) Rotate the XYZ coordinate system about the x-axis by α as shown in Figure 23(a).

3) Rotate the XYZ coordinate system about the fixed y-axis by β as shown in Figure 23(b).

4) Rotate the XYZ coordinate system about the fixed z-axis by γ as shown in Figure 23(c).

13120

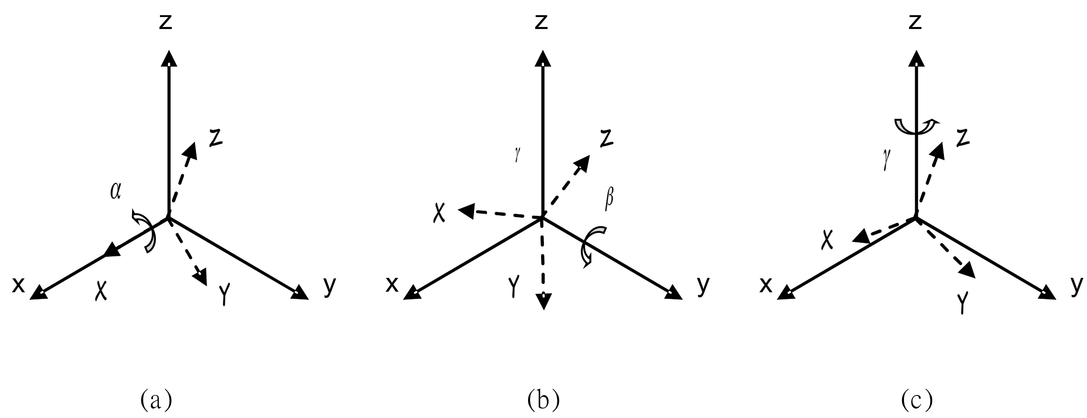


Figure 23: Three sequential rotations for the xyz-Euler angle representation used in the common data format type I-1, II-1, and III-1

The XYZ-Euler Angle representation is commonly called as yaw-pitch-roll rotation and defined as follows:

1) Start with the rotating XYZ coordinate system coinciding with the fixed xyz coordinate system. Most familiar case appears in the x-axis directed to north, the y-axis directed to east and the z-axis directed to the center of the globe. Practically, the X-axis is set to the forward motion direction of a vehicle and the origin is fixed at the rotation reference point of the vehicle.

2) Rotate the XYZ coordinate system about the Z-axis by α (yaw angle) as shown in Figure 24(a).

3) Rotate the XYZ coordinate system about the newly rotated Y-axis by β (pitch angle) as shown in Figure 24(b).

4) Rotate the XYZ coordinate system about the newly rotated X-axis by γ (roll angle) as shown in Figure 24(c).

13120

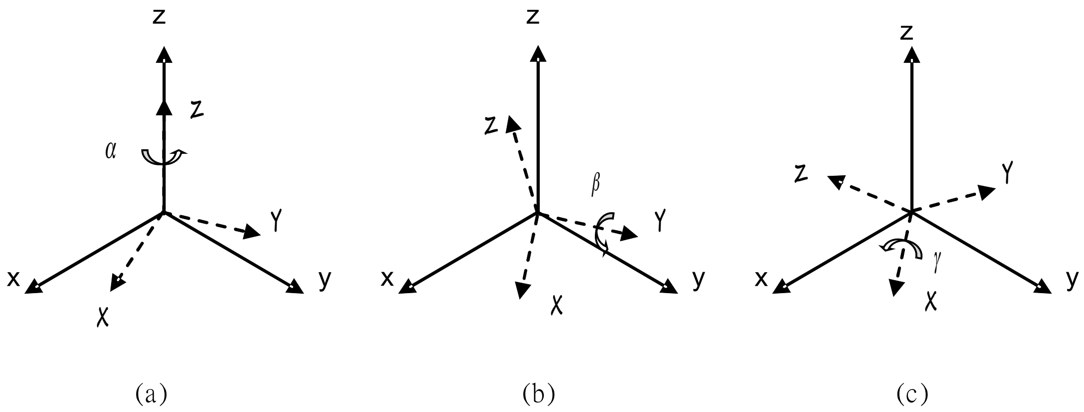


Figure 24: Three sequential rotations for the XYZ-Euler angle representation used in the common data format type I-2, II-2, and III-2

12916 8.6 Filter Condition Package

13184 When a location service is operated in a large scale and handles a large number of location information, it is useful that the service has a filtering functionality by which it limits outgoing RoLo data by a given condition. Without this functionality, service providers and receivers are required to have large capacities of output/input to process the whole data from large scaled systems. Suppose, as an example, that we implement a sensor system at a shopping center which detects thousands of guests at once and provides localization service for robots. In such case, it is not reasonable for each robot to receive localization data about the whole guests every time. Instead each robot is generally interested in specific guests identified by certain features, area, and/or time period.

The "Filter Condition" specified below is aimed to provide the functionality for localization services to specify a condition for filtering data sent to service receivers.

Filter Condition in RoLo stream

A RoLo output stream may have the functionality to filter localization results by a certain condition. We call this condition as "filter condition." When a filter condition is specified, each localization result is tested by the condition and passed to the output stream only when it satisfies the condition.

If no condition is given, or if the stream has no such functionality, the "True" condition is used as the default condition, in which all localization results are passed to the output stream.

To handle the filter condition functionality, ability descriptor for RoLo streams shall additionally have the following parameter:

13184 **Table 96 - Filter Condition parameter for RoLo streams**

filterCondition	Parameter<::ISO19143::NonIdOperator> (RoLo::Interface)	O	1	Filter condition to be used for output data. Default value is 'True'.

Users can set and get the content of this parameter through the 'setParameterValueSet' and 'getParameterValueSet' methods toward the stream or the service. When filter condition is not supported by the stream, UNSUPPORTED_PARAMETER will be returned.

Data Format of Filter Condition

In order to specify a filter condition, we follow the ISO 19143 specification [ISO/DIS19143] which is defined for ISO 19142 [ISO/DIS19142]. ISO 19143 specifies XML encoding and UML class charts of filter conditions and their operators.

While the UML charts provides general concepts of data format of the filter condition, it is generally useful and flexible enough to use the XML encoding for the localization service. (see Examples.)

12916
8.7 Interface Package

Several types of modules are commonly used in robotic localization services in general. The simplest form of module is that which receives data from sensors, calculates location and outputs the results. However, this type of interface strongly depends on sensor interfaces or sensor output formats. Strong dependency on specific products or vendors is not suitable for standardization. Moreover, when a location is calculated, many kinds of resources such as map data, specific to each sensing system, are required. It is impractical to include each of these resources into the standard specification. Thus, we decided to embed and hide the individual device or localization algorithm details inside the module structure (Figure 25).

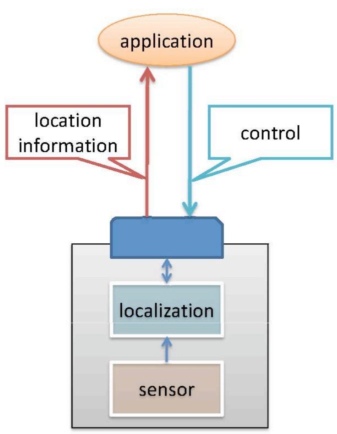


Figure 25: Basic robotic localization module

On the other hand, if we focus on functionality required to localization modules, we can classify them into roughly three classes (Figure 34):

A) Table 97 - Common data format type II

Parameter	Format of value	Value type	Unit
Position	[γ,α,β]	Real	radian
Orientation	[θ,ϕ,ψ] (roll, pitch, yaw)	Real	degree
Timestamp	POSIX time	Real	millisecond
ID	--	Integer	--

The type II format delivers location information based on Polar coordinate system. It includes four parameters.

- ***Position*** – specifies the coordinate value in a 3-dimensional Polar coordinate system. The unit of each value is radian.
- ***Orientation*** – specifies the rotation along each axis in a right-handed 3-dimensional Cartesian coordinate system. The unit of this parameter is radians. The order of orientation is the rotation around x-axis, the rotation around y-axis, and the rotation around z-axis of 3-dimentional right-handed Cartesian coordinate system; roll, pitch, and yaw are conventional terms.
- ***Timestamp*** – specifies time at occurring measurement for current position and orientation. The format is compatible to POSIX time which is the number of second elapsed since midnight Coordinated Universal Time (UTC) of January 1, 1970. The resolution of timestamp is milliseconds.
- ***ID*** – specifies the identifier of current location information for robots and related entities.

Type III

Table 98 - Common data format type III

Parameter	Format of value	Value type	Unit
Position	[longitude, latitude, height]	Real	degree, meter
Orientation	[θ,ϕ,ψ] (roll, pitch, yaw)	Real	degree
Timestamp	POSIX time	Real	millisecond
ID	--	Integer	--

The type III format delivers location information based on geodetic coordinate system. It includes four parameters.

- ***Position*** – specifies the coordinate value in a geodetic coordinate system. The value of this parameter represents longitude, latitude, and height in order. The unit of longitude and latitude is degree and that of height is meter.

- ***Orientation*** – specifies the rotation along each axis in a right-handed 3-dimensional Cartesian coordinate system. The unit of this parameter is radians. The order of orientation is the rotation around x-axis, the rotation around y-axis, and the rotation around z-axis of 3-dimentional right-handed Cartesian coordinate system; roll, pitch, and yaw are conventional terms.

- ***Timestamp*** – specifies time at occurring measurement for current position and orientation. The format is compatible to POSIX time which is the number of second elapsed since midnight Coordinated Universal Time (UTC) of January 1, 1970. The resolution of timestamp is milliseconds.

- ***ID*** – specifying the identifier of current location information for robots and related entities.

8.8 Filter Condition

RoLo Service (or RoLo Stream) may have the functionality to filter localization results by a certain condition. We call the condition as "filter condition." When a filter condition is specified, each localization result is tested by the condition, and passed to the output stream only when it satisfies the condition.

If no condition is given, or the service has no such functionality, the "True" condition is used as the default condition, in which all localization results are passed to the output stream.

To handle Filter Condition functionality, RoLo Service shall have the following operations.

Table 99 - Filter Condition Extension of RoLo Service class

Description: Interface for the Robotic Localization Service.					
Derived From: OutStream					
Attributes					
inStream		InStream	O	N ord	An ordered list of input stream objects.
Operations					
getNumInStream		Returns the number of input streams.			
out	num	Integer	Number of input streams equipped in this module.		
getInStream		Returns the input stream of given index.			
in	index	Integer	Index number indicating the target InStream.		
out	stream	InStream	Reference to the target InStream.		
adjust		Method for adjusting localization results. For elements not required for adjustment, NULL element should be specified.			
in	data	RoLo Data	Data to be used for initialization or adjustment. Adjusts every element at once.		
getChild		Return the service connected to the InStream of this service module.			

in	index	Integer	Index number indicating the target child service.
out	service	RoLo Service	Reference to the target child service
getAbility		Obtain ability description for this service.	
out	ability	Service Ability	Ability description for this service
setParameter		Set parameter values for this service.	
in	parameter	Service Parameter	Parameter values to be set.
getParameter		Obtain current parameter values for this service.	
out	parameter	Service Parameter	Current parameter settings.
setFilterCondition		**Set a filter condition**	
in	**condition**	**FC_FilterOperator**	**a filter condition to set**
getFilterCondition		**Return the filter condition set to the service**	
in	**condition**	**FC_FilterOperator**	**a filter condition set to the service.**

8.8.1 Filter Operators

In order to specify a filter condition, we follow the specification of "OpenGIS Filter Encoding Implementation Specification" (ISO 19143), which is defined for "OpenGIS Web Feature Service Implementation Speficication" (ISO 19142). While ISO 19143 only specifies XML encoding, we need object models for the condition. The following specification is re-written object model of ISO 19143.

Most of the non-abstracted operators described below have the same meanings as the XML elements in ISO 19143 that have the same names, respectively. However, there are two exceptions in this specification.

The first exception is FC_True and FC_False classes. They have no corresponding XML elements in ISO 19143. They denote simply logical true and false.

The second exception is FC_PropertyName class. There are corresponding XML element, `<propertyName>` in ISO 19143, which specifies the position of a focusing property in a geographical feature. On the other hand, in the RLS specification, FC_PropertyName specifies a RoLo Position or its component in a RoLo Data.

Top Level

FC_Node class is the top level abstracted class for all elements that is used to express filter conditions. FC_Node should have a class method called 'isInstanceOf' that receive a node (an instance of FC_Node) and a class. The method shall check the node is an instance of the class or its derived class.

FC_FilterOp class is the top level abstracted class for all operation used in filter conditions.

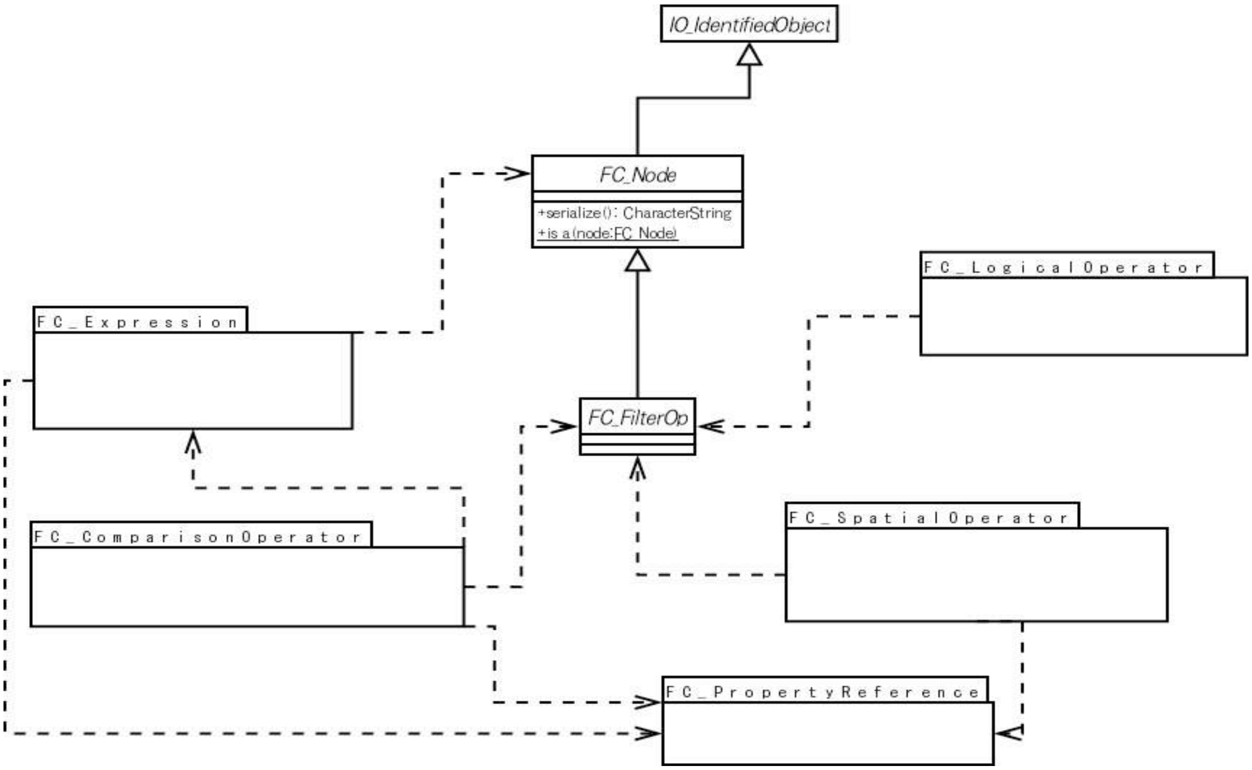


Figure 26 - Filter Condition Toplevel

Table 100 - Filter Condition Node class

FC_Node			
Description: abstracted class for tree node in filter operators			
Derived From:			
Operations:			
isInstanceOf	Returncode_t	Check a given node is an instance of klass class.	
in	node	NC_Node	A node to be checked.
in	klass	Class	A class that node may belong to
out	belongness	boolean	if node belongs to klass, return true. otherwise, return false.

Table 101 - Filter Op class

FC_FilterOp
Description: abstracted class for filter operator

FC_FilterOp
Derived From: FC_Node

Logical Operators

The Filter Condition facility should provide general logical operations like AND, OR, and NOT. FC_And, FC_OR, and FC_Not are classes to denote these operations respectively.

The Filter Condition facility also may provide two atomic logical operation, FC_True and FC_False, which means logical true and false value.

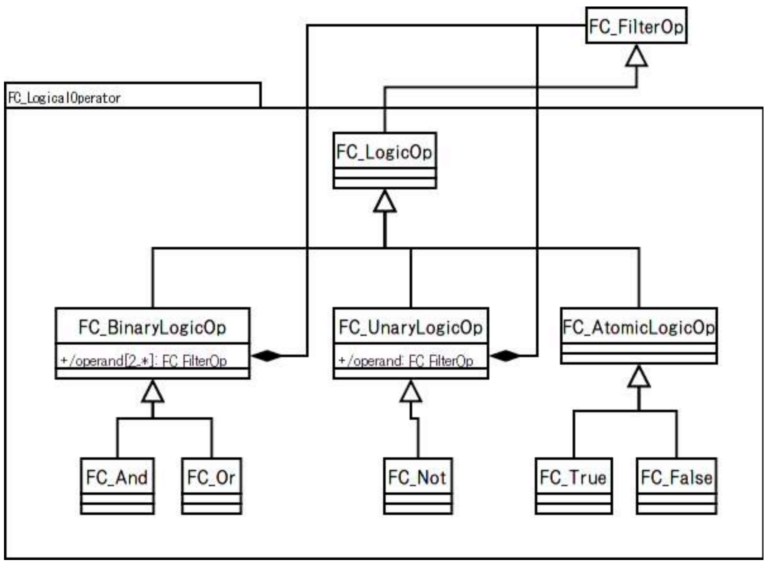


Figure 27 - Logical Operators

Table 102 - LogicOp class

FC_LogicOp
Description: abstracted class for logical operator node
Derived From: FC_FilterOp

Table 103 - BinaryLogicOp class

FC_BinaryLogicOp
Description: abstracted class for logical operator node that takes two or more operands
Derived From: FC_LogicOp
Attributes

FC_BinaryLogicOp				
/operand	FC_FilterOp	M	2-N	A list of children nodes of the logic operation

Table 104 - UnaryLogicOp class

FC_UnaryLogicOp		
Description: abstracted class for logical operator node that takes an operand.		
Derived From: FC_LogicOp		
Attributes		
/operand	FC_FilterOp	a child node of the logic operation

Table 105 - AtomicLogicOp class

FC_AtomicLogicOp
Description: abstracted class for atomic logical operator.
Derived From: FC_LogicOp

Table 106 - And class

FC_And
Description: denote logical AND operation
Derived From: FC_BinaryLogicOp

Table 107 - Or class

FC_Or
Description: denote logical OR operation
Derived From: FC_BinaryLogicOp

Table 108 - Not class

FC_Not
Description: denote logical negation operation
Derived From: FC_UnaryLogicOp

Table 109 - True class

FC_True
Description: denote logical trueth value
Derived From: FC_AtomicLogicOp

Table 110 - False class

FC_False
Description: denote logical false value
Derived From: FC_AtomicLogicOp

Spatial Operators

The Filter Condition facility shall provide whole or a part of spatial operators, which specify to check the spatial relationship between specified RoLo Position data and geometry.

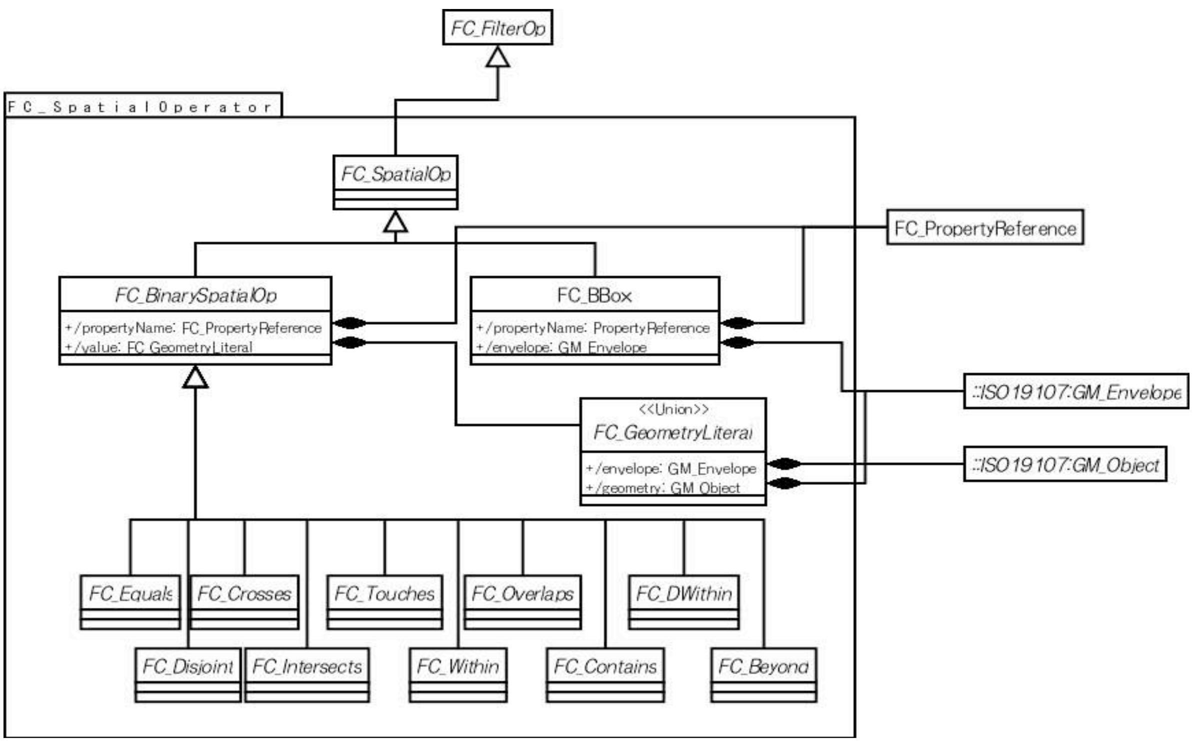


Figure 28 - Spatial Operators

Table 111 - SpatialOp class

FC_SpatialOp
Description: abstracted class for spatial operations
Derived From: FC_FilterOp

Table 112 - BBox class

FC_BBox		
Description: denote boundary box condition. It means TRUE when a RoLo Position of a localization result speficied by propertyName is overlapped with an envelope specified by geometry. Otherwise, it means FALSE.		
Derived From: FC_SpatialOp		
Attributes		
propertyName	FC_PropertyReference	Specify a reference to a RoLo position or its component.
Geometry	GML:Envelope	Specify an envelope the

Table 113 - FC_BinarySpatialOp class

FC_BinarySpatialOp		
Description: abstracted class for binary spatial operators		
Derived From: FC_SpatialOp		
Attributes		
propertyName	FC_PropertyReference	Specify a reference to a RoLo position or its component.
value	FC_GeometryLiteral	Specify a geometry value

Table 114 - FC_GeometryLiteral class

FC_GeometryLiteral		
Description:union of GML:Envelope and GML:Geometry.		
Derived From:		
Attributes		
envelope	GM_Envelope [ISO19107]	Specify an envelop (box) for a spatial comparison operation
geometry	GM_Object [ISO19107]	Specify a generic geometrical value for a spatial comparison operation

Table 115 - Equals class

FC_Equals
Description: denote a RoLo Position (specified by propertyName) of a localization result is equal to a geometry specified by value.
Derived From: FC_BinarySpatialOp

Table 116 - Disjoint class

FC_Disjoint
Description: denote a RoLo Position (specified by propertyName) of a localization result is disjoint to a geometry specified by value.
Derived From: FC_BinarySpatialOp

Table 117 - Crosses class

FC_Crosses
Description: denote a RoLo Position (specified by propertyName) of a localization result is crossing with a geometry specified by value.
Derived From: FC_BinarySpatialOp

Table 118 - Intersects class

FC_Intersects
Description: denote a RoLo Position (specified by propertyName) of a localization result is intersecting with a geometry specified by value.
Derived From: FC_BinarySpatialOp

Table 119 - Touches class

FC_Touches
Description: denote a RoLo Position (specified by propertyName) of a localization result is touching with a geometry specified by value.
Derived From: FC_BinarySpatialOp

Table 120 - Within class

FC_Within
Description: denote a RoLo Position (specified by propertyName) of a localization result is included by a geometry specified by value.

FC_Within
Derived From: FC_BinarySpatialOp

Table 121 - Overlaps class

FC_Overlaps
Description: denote a RoLo Position (specified by propertyName) of a localization result overlaps on a geometry specified by value.
Derived From: FC_BinarySpatialOp

Table 122 - Contains class

FC_Contains
Description: denote a RoLo Position (specified by propertyName) of a localization result contains a geometry specified by value.
Derived From: FC_BinarySpatialOp

Table 123 - DWithin class

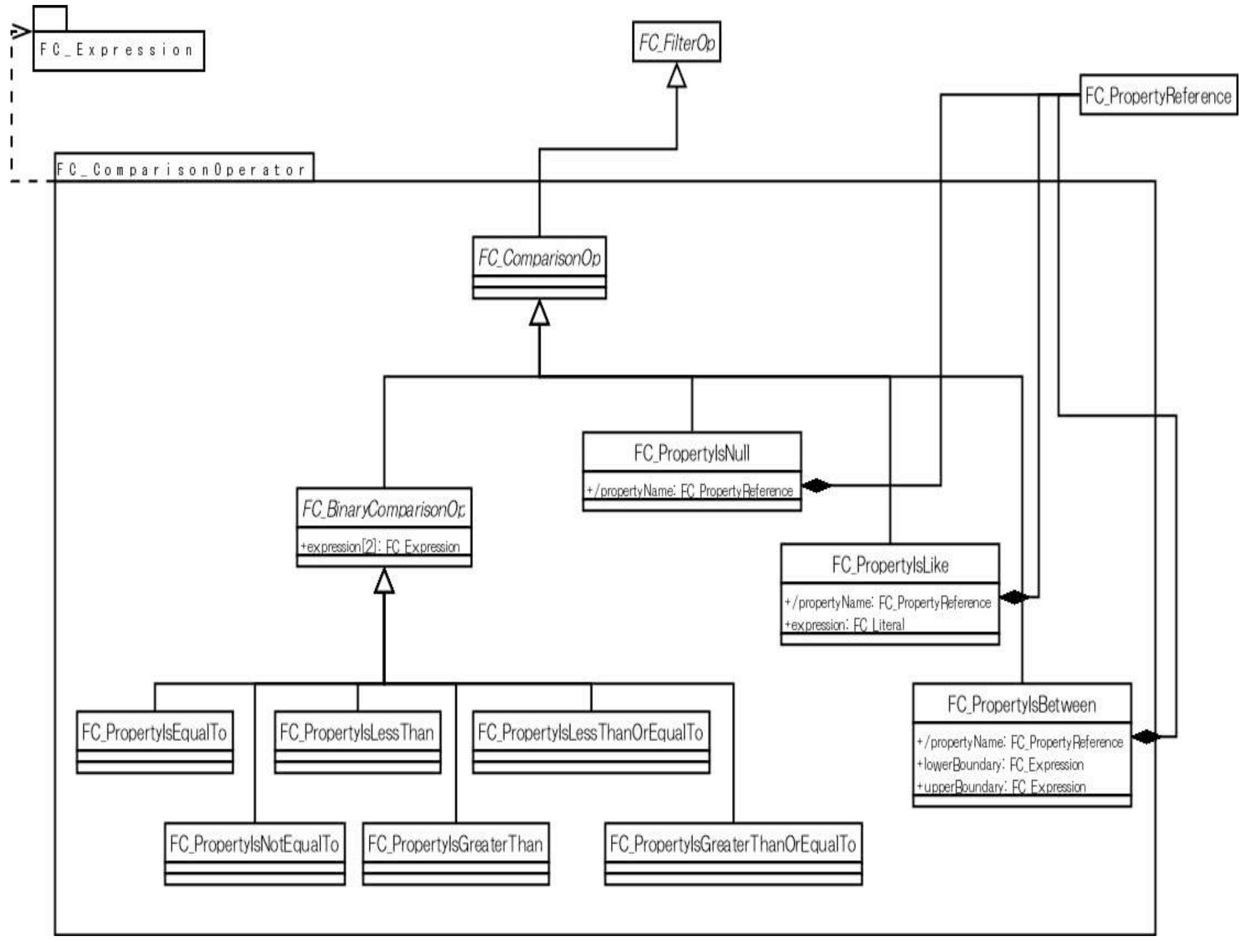
FC_Dwithin
Description:
Derived From: FC_BinarySpatialOp

Table 124 - Beyond class

FC_Beyond
Description: denote a RoLo Position (specified by propertyName) of a localization result is equal to a geometry specified by value.
Derived From: FC_BinarySpatialOp

Comparison Operators

The Filter Condition facility should provide whole or a part of comparison operations, which check general comparison operations like 'equal to,' 'less than,' 'greater than,' and so on.



~~Figure 29 - Comparison Operators~~

~~Table 125 - ComparisonOp class~~

~~FC_ComparisonOp~~
~~**Description:** abstract class for numerical and symbolic comparison operation in filter condition.~~
~~**Derived From:** FC_FilterOp~~

~~Table 126 - BinaryComparisonOp class~~

~~FC_BinaryComparisionOp~~				
~~**Description:** abstract class for comparison operations that takes two expressions as operands.~~				
~~**Derived From:** FC_FilterOp~~				
~~**Attributes**~~				
~~/operand~~	~~FC_Expression~~	~~M~~	~~2~~	~~Specify two expressions that are used for the comparison.~~

Table 127 - PropertyIsEqualTo class

FC_PropertyIsEqualTo
Description: denote that two expressions are equal to each other.
Derived From: FC_BinaryComparisonOp

Table 128 - PropertyIsNotEqualTo class

FC_PropertyIsNotEqualTo
Description: denote that two expressions are not equal to each other.
Derived From: FC_BinaryComparisonOp

Table 129 - PropertyIsLessThan class

FC_PropertyIsLessThan
Description: denote that the first expression is less than the second expression
Derived From: FC_BinaryComparisonOp

Table 130 - PropertyIsGreaterThan class

FC_PropertyIsGreaterThan
Description: denote that the first expression is greater than the second expression.
Derived From: FC_BinaryComparisonOp

Table 131 - PropertyIsLessThanOrEqualTo class

FC_PropertyIsLessThanOrEqualTo
Description: denote that the first expression is less than or equal to the second expression.
Derived From: FC_BinaryComparisonOp

Table 132 - PropertyIsGreaterThanOrEqualTo class

FC_PropertyIsGreaterThanOrEqualTo
Description: denote that the first expression is greater than or equal to the second expression.
Derived From: FC_BinaryComparisonOp

Table 133 - PropertyIsNull class

FC_PropertyIsNull		
Description: denote that a RoLo Position or its component specified by propertyName is null.		
Derived From: FC_FilterOp		
Attributes		
propertyName	FC_PropertyReference	Specify a reference to a RoLo Position or its component.

Table 134 - PropertyIsLike class

FC_PropertyIsLike		
Description: denote that a RoLo Position or its component specified by propertyName matches with pattern.		
Derived From: FC_FilterOp		
Attributes		
propertyName	FC_PropertyReference	Specify a reference to a RoLo Position's identifier or its component.
pattern	FC_Literal	Specify a pattern to match.

Table 135 - PropertyIsBetween class

FC_PropertyIsBetween		
Description: denote that a RoLo Position or its component specified by propertyName is between two expressions specified by lowerBoundary and upperBoundary.		
Derived From: FC_FilterOp		
Attributes		
PropertyName	FC_PropertyReference	Specify a reference to a RoLo Position's identifier or its component.
lowerBoundary	FC_Expression	Specify a lower boundary of the comparison.
upperBoundary	FC_Expression	Specify a upper boundary of the comparison..

Expressions

The expression class and its ancestors are used to represent numerical or symbolic (character string) operations for the comparison operations.

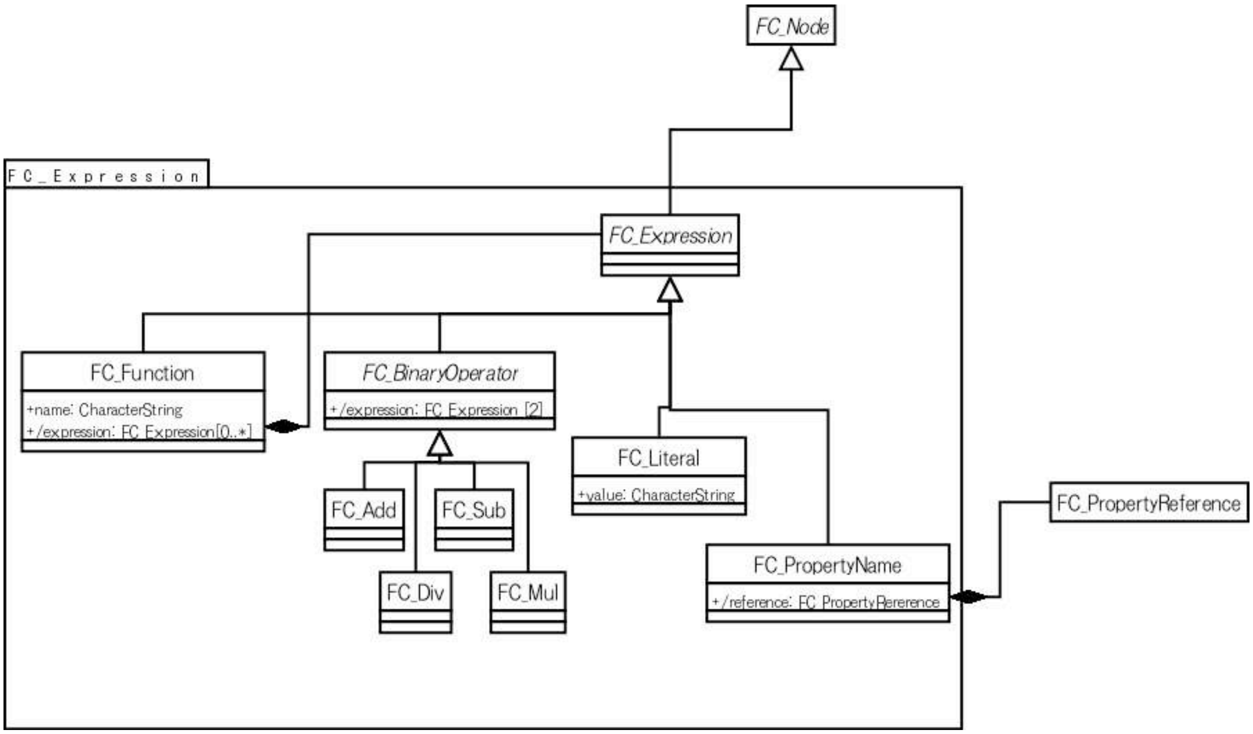


Figure 30 - Expressions

Table 136 - Expression class

FC_Expression
Description: abstract class for expression node that is used for expression attribute of FC_ComparisonOP. An expression denotes a numerial or symbolic value that is handled in comparison operation of filter conditions.
Derived From: FC_Node

Table 137 - BinaryOperator class

FC_BinaryOperator				
Description: abstract class for expressions that take two operands.				
Derived From: FC_Expression				
Attributes				
/expression	FC_Expression	M	2	Specify two expressions that are used for the operation.

Table 138 - Add class

FC_Add
Description: denote numerical addition of two expressions.
Derived From: FC_BinaryOperator

Table 139 - Sub class

FC_Sub
Description: denote numerical subtract of two expressions.
Derived From: FC_BinaryOperator

Table 140 - Mul class

FC_Mul
Description: denote numerical multiplication of two expressions.
Derived From: FC_BinaryOperator

Table 141 - Div class

FC_Div
Description: denote numerical division of two expressions.
Derived From: FC_BinaryOperator

Table 142 - Literal class

FC_Literal		
Description: denote literal value of numbers, symbols, and character strings.		
Derived From: FC_Expression		
Attributes		
value	CharacterString	Specify a literal data in character string.

Table 143 - PropertyName class

FC_PropertyName		
Description: denote reference to RoLo position value or its component of a RoLo Data as a localization result.		
Derived From: FC_Expression		
Attributes		
Reference	FC_PropertyReference	Specify a reference to a RoLo position value or its component.

Table 144 - Function class

FC_Function				
Description: denote a named procedure that performs a distinct calculation. A function may accept zero or more expressions as input and generate a single scalar value.				
Derived From: FC_Expression				
Attributes				
name	CharacterString	M	1	Specify the name of the function.
/expression	FC_Expression	M	N	Specify zero or more arguments for the function.

8.8.2 PropertyReference

FC_PropertyReference indicates a reference to a RoLo element or one of its components. This is used in comparison operators, spatial operators, and expressions. Generally, it specifies which part in a RoLo data is utilized for a filter condition.

The specification of the reference consists of three attributes, elementSpec, elementComponentName, and componentAxis. The elementSpec attribute specifies which RoLo element in a RoLo data is referred to. The elementComponentName attribute determines which of `pos' and `err' attributes in a position element is referred. The component Axis specifies which axis component in a RoLo position is referred if necessary.

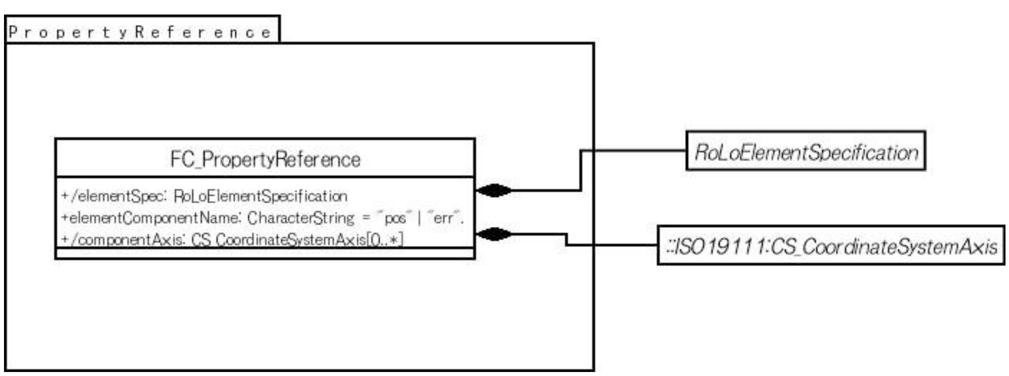


Figure 31 - Property Reference

Table 145 - FC_PropertyReference

FC_PropertyReference		
Description: denote a reference to a RoLo position or its component		
Derived From: FC_Node		
Attributes		
/elementSpec	RoLoElementSpecification	Specify a RoLo Element Specification to refer in a RoLo Archicacture.
elementComponentName	CharacterString	Specify which of position or error component is referred. The value shall be one of "pos" and "err".
componentAxis	CS_CoordinateSystemAxis [0..*]	Specify an axis in the referred position element.

8.8.3 Capability of Filter Condition

In order to inform user modules of the ability of filter conditioning of a RoLo Service, the RoLo Service module shall indicate its abilities in its "ability" attribute. The conceptual format of the ability of the filter condition is the same as the capability specification in ISO 19143 (OpenGIS Filter Encoding Implemenatation).

8.9 Service Interface

Several types of modules are commonly used in robotic localization services in general. The simplest form of module is that which receives data from sensors, calculates location and outputs the results. However, this type of interface strongly depends on sensor interfaces or sensor output formats. Strong dependency on specific products or vendors is not suitable for standardization. Moreover, when a location is calculated, many kinds of resources such as map data, specific to each sensing system, are required. It is impractical to include each of these resources into the standard specification. Thus, we decided to embed and hide the individual device or localization algorithm details inside the module structure (Figure 32).

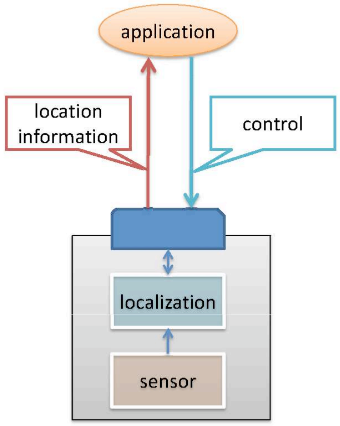


Figure 32 - Basic robotic localization module

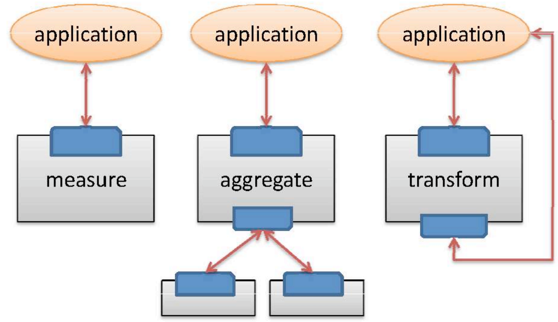


Figure 33 - Structures of robotic localization modules with different functionalities

On the other hand, if we focus on functionality required to localization modules, we can classify them into roughly three classes (Figure 33):

B) Calculate localization results based on sensor outputs (measurement)

C) Aggregate or integrate multiple localization results (aggregation)

D) Transform localization results into different coordinate reference systems (transformation)

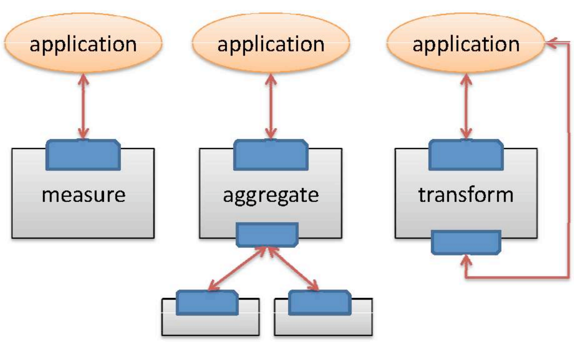


Figure 34: Structures of robotic localization module with different functionalities

13439 These functionalities differ in their internal algorithms or the number of input / output streams. However, in all of these, the main data to be exchanged is localization results. As we are focusing on the interface of RLS modules, and not on their functionalities, we decided to abstract these different types of modules into a single form of module. This abstract module holds n (>=0) input streams and a uniformone output stream. By abstracting various types of modules and assuming a uniform interface, complex module compositions such as hierarchical or recursive module connections can be easily realized (Figure 35).

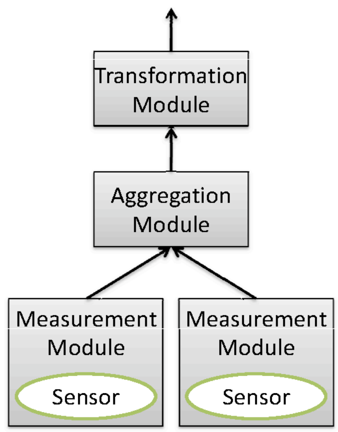


Figure 35: Example of a cascading module connection

13439 A RoLo service (implemented as a Service class) may have one or more RoLo output streams (OutStream class) and zero or more RoLo input streams (InStream class). Typically, the number of RoLo inputs a service owns is predetermined and the number of RoLo outputs a service owns changes dynamically based on requests from service users. This is similar to typical server systems such as database or Web servers where the number of established output connection increases as requests arrive until it reaches a predefined maximum number.

13322 If each module can represent what or how it can perform, or provide information on available configurable parameters, a large amount of development efforts can be reduced. Thus, each service or stream is modeled to own an ***ability*** description (Ability class) which contains a set of ***attributes*** (Attribute class) and ***parameters*** (Parameter class). Attributes show some static nature of a module and parameters indicate its configurable parameters. For example, an ability description for a service (ServiceAblity class) includes an attribute describing expected value of latency. And an ability description for a stream (StreamAbility class) includes parameters denoted by lists of DataSpecification and DataFormat objects which shows what type of data structure or data format a stream can handle, respectively. Attributes or parameters specific to each implementation, such as vendor-specific parameters, can be described by extending the respective classes. As such, attributes may be used to describe fixed nature (catalogue specs) of modules, while parameters define configurable settings for modules. Note that, some parameters may not be configurable on some implementations. For example, if an module implementation can output data only by a single data format, the aforementioned parameter for DataFormat may show only a single candidate, and be marked as non-configurable (Parameter.isConfigurable = false).

Often, parameters are defined over some limited value domains. As in the example given above, data specifications or data formats that a stream is able to pass data are likely to be limited to sets with a small number of choices. Or some parameters, such as output frequency, may be restricted under a limited range of values. The attribute 'domain' in ParameterOverDomain class is aimed to denote these limitations. As the value domain required may take variations of forms such as finite set or interval (or range). The

13322 ParameterOverDomain class is defined as an template class which allows a type argument for indicating what sort of value domain shall be specified.

By defining the “meaning” of attributes and parameters, the ambiguity in functional definition or parameters can be eliminated which can be expected to increase developing efficiency. For example, what does the value 0.23 given as an 'expectedError' attribute for a RoLo Service mean? These ambiguities can often been seen in sensor products such as GPS receivers, making it difficult to design a reusable system applicable to devices or modules from different vendors. The AttributeDefinition class is aimed to clarify the meaning of attributes and parameters. Although this is out of scope for this specification, by providing a repository of AttributeDefinition objects that can be referred on demand, RoLo service users and developers can always make sure what each ability description means or on which unit they are defined on.

Moreover, advanced features can be implemented such as verification of inter-module connection, automatic search of specific modules or semi-automatic parameter negotiation between modules. In cases where sensors or robots distributed in the environment cooperate with each other, namely the Network Robot environment, it becomes essential to register each module‘s capabilities in repositories and make them searchable.

13322

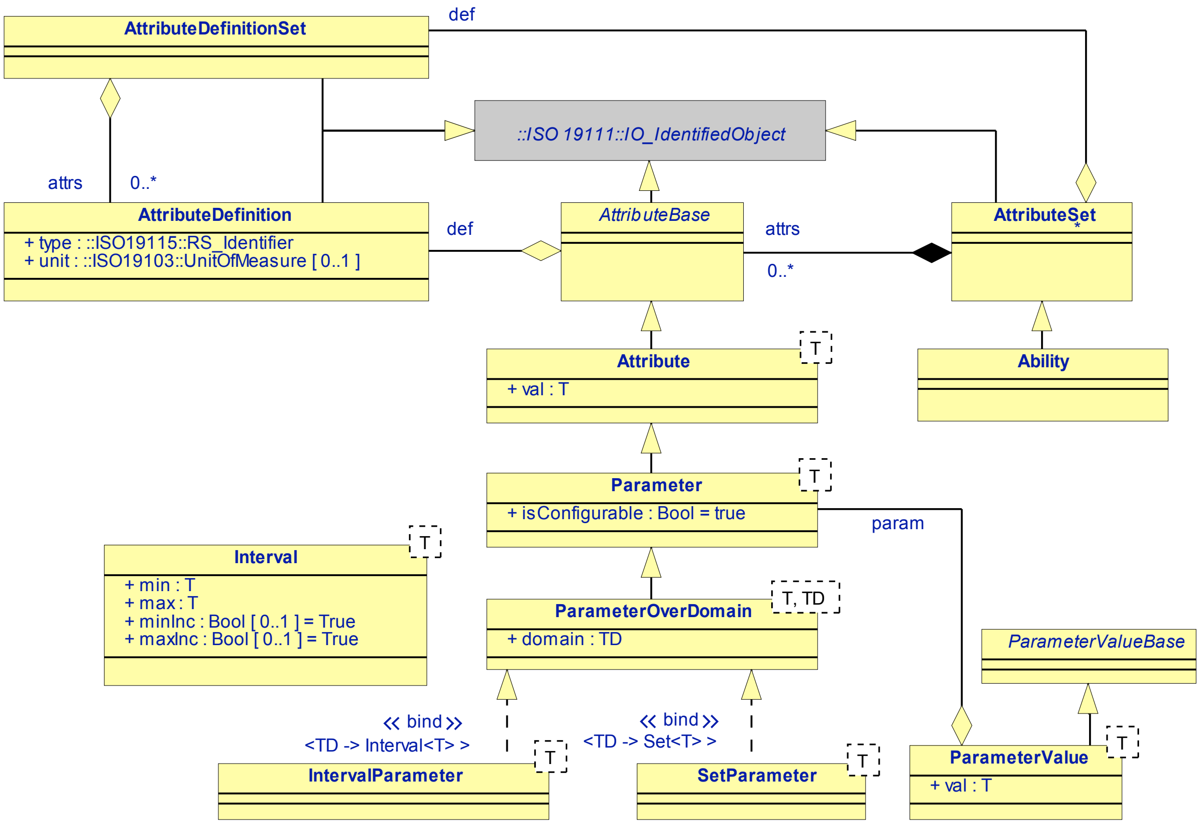


Figure 36 - RoLo Ability

13322 **Table 146 - AttributeDefinition class**

Description: Definition of a single attribute.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes				
type	RS_Identifier [ISO19115]	M	1	Type descriptor for this attribute.
unit	UnitOfMeasure [ISO19103]	O	1	Unit of the target attribute. If no unit is required, this may be omitted.

~~**Figure 37 - Example of a cascading module connection**~~

~~The Robotic Localization Service (RLS) module has one output ***stream*** and n (>=0) input streams. As the functionality of the output stream is the most important to RLS, the service module itself is modeled as an output stream. Each stream owns an ***ability*** description, which shows how this stream can perform and be configured. This includes the list of RoLo Architecture or data formats that this stream can handle, or other configurable parameters specific to the module. Each service module also owns an ability description for the service it provides, besides the ability description for its output stream. The configurable ***parameters*** defined in the ability description can be specified values via the module interface, restricted by the ability description held by the service module or the belonging stream modules.~~

If each module can represent what it can perform, or provide information on available configurable parameters, a large amount of development efforts can be reduced. By defining the "meaning" of parameters, the ambiguity in functional definition or parameters can be eliminated, resulting in increase of developing efficiency. Moreover, advanced features can be implemented such as verification of inter-module connection, automatic search of specific modules or semi-automatic parameter negotiation between modules. In cases where sensors or robots distributed in the environment cooperate with each other, namely the Network Robot environment, it becomes essential to register each module's capabilities in repositories and make them searchable.

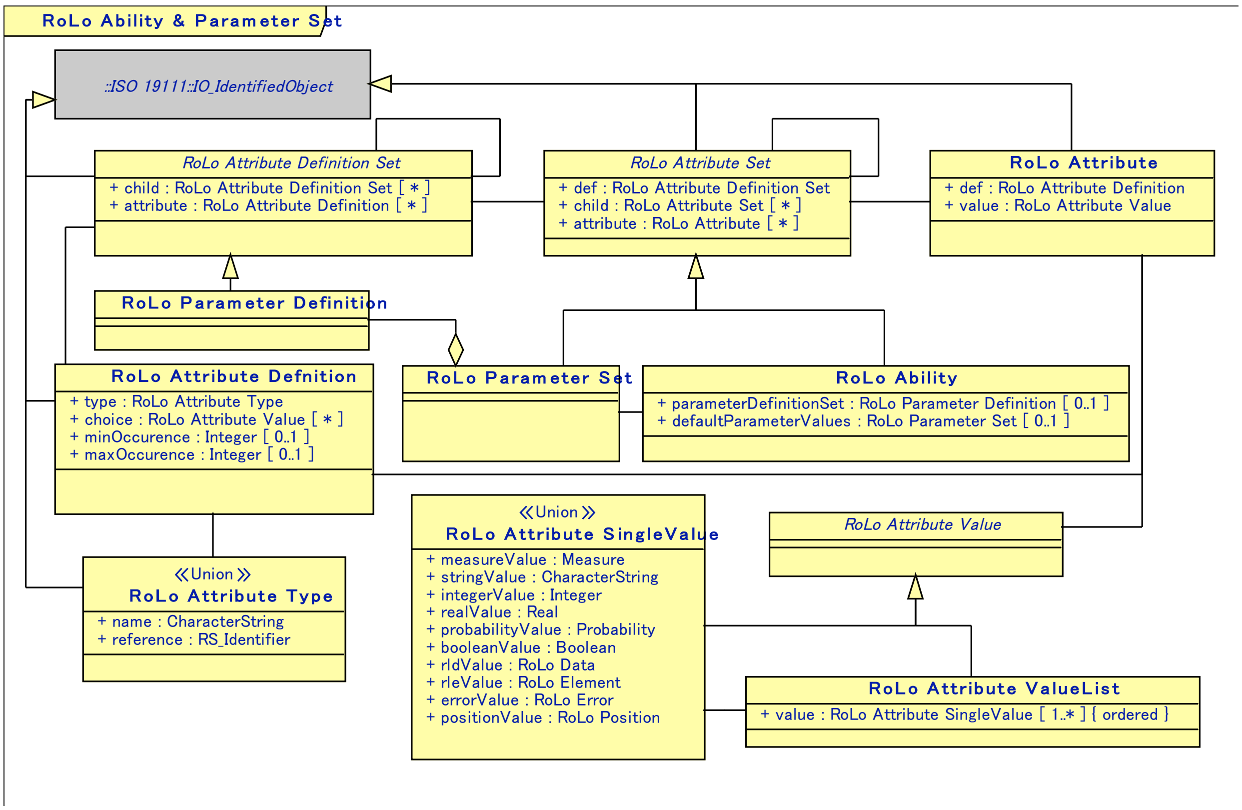


Figure 38 - RoLo Ability and RoLo Parameter Set

Table 147 - RoLo Attribute Type class

Description: Type reference for describing RoLo Attribute Definition.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes				
name	CharacterString	C	1	Name of the Type.
reference	RS_Identifier [ISO19115]	C	1	Identifier of the Type.

Condition: One and only one of the attributes shall be specified.

Table 148 - RoLo Attribute Definition class

Description: Definition of a single attribute. If both minOccurence and maxOccurence are specified as 1, then this attribute is mandatory. Of minOccurence is specified as 0, this attribute is optional.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes				
type	RoLo Attribute Type	M	1	Type descriptor for this attribute.
choice	RoLo Attribute Value	O	N	List of the available choices for this attribute's value.
minOccurence	Integer	O	1	Minimal occurrence of this attribute. If omitted, the defaulting value is 0.
maxOccurence	Integer	O	1	Maximal occurrence of this attribute. If omitted, the defaulting value is 1.

Table 149 - RoLo Attribute Definition Set class

Description: Abstract class that represents a hierarchical set of attribute definitions.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes				
child	RoLo Attribute Definition Set	O	N	Child attribute definition sets included.
attribute	RoLo Attribute Definition	M	N	Attribute definitions.

Table 150 - RoLo Parameter Definition class

Description: Represents an parameter definition for modules. Users may extend this class to describe natures of their own module implementations.
Derived From: RoLo Attribute Definition Set

Table 151 - RoLo Attribute Value class

Description: Abstract class for holding values for RoLo Attribute. May be a single value or a list of values.
Derived From: (none)

Table 152 - RoLo Attribute SingleValue class

Description: Value holder for RoLo Attribute.
Derived From: RoLo Attribute Value

Attributes				
measureValue	Measure [ISO19103]	C	1	Measurement value with unit.
stringValue	CharacterString	C	1	String value.
integerValue	Integer	C	1	Integer value.
realValue	Real	C	1	Real value.
probabilityValue	Probability [ISO19103]	C	1	Probability value.
booleanValue	Boolean	C	1	Boolean value.
rldValue	RoLo Data	C	1	RoLo Data value.
rleValue	RoLo Element	C	1	RoLo Element value.
errorValue	RoLo Error	C	1	RoLo Error value.
positionValue	RoLo Position	C	1	RoLo Position value.
Condition: One and only one of the choices shall be specified.				

Table 153 - RoLo Attribute ValueList class

Description: Represents a list of RoLo Attribute SingleValue.				
Derived From: RoLo Attribute Value				
Attributes				
value	RoLo Attribute SingleValue	M ord	N	Ordered list of values.

Table 154 - RoLo Attribute class

Description: Represents a single attribute.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes				
def	RoLo Attribute Definition	M	1	Definition of this attribute.
value	RoLo Attribute Value	M	1	Value of this attribute

Table 155 - RoLo Attribute Set class

Description: Represents a hierarchical set of attributes.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes				
def	RoLo Attribute Definition Set	M	1	Definition of this attribute set.
child	RoLo Attribute Set	O	N	Child attribute sets included.

~~attribute~~	~~RoLo Attribute~~	~~M~~	~~N~~	~~Attributes.~~

~~Table 156 - RoLo Parameter Set class~~

~~**Description:** Parameter value sets.~~
~~**Derived From:** RoLo Attribute Set~~

~~Table 157 - RoLo Ability class~~

~~**Description:** Describes module ability.~~				
~~**Derived From:** RoLo Attribute Set~~				
~~**Attributes**~~				
~~parameterDefinitionSet~~	~~RoLo Parameter Definition Set~~	~~O~~	~~1~~	~~Definitions for the parameters, that describes the parameters for which module this ability description stands for.~~
~~defaultParameterVals~~	~~RoLo Parameter Set~~	~~O~~	~~1~~	~~Default values for the parameter value. If the attribute parameterDefinitionSet is specified, this should be specified and shall contain default values for all the parameters.~~

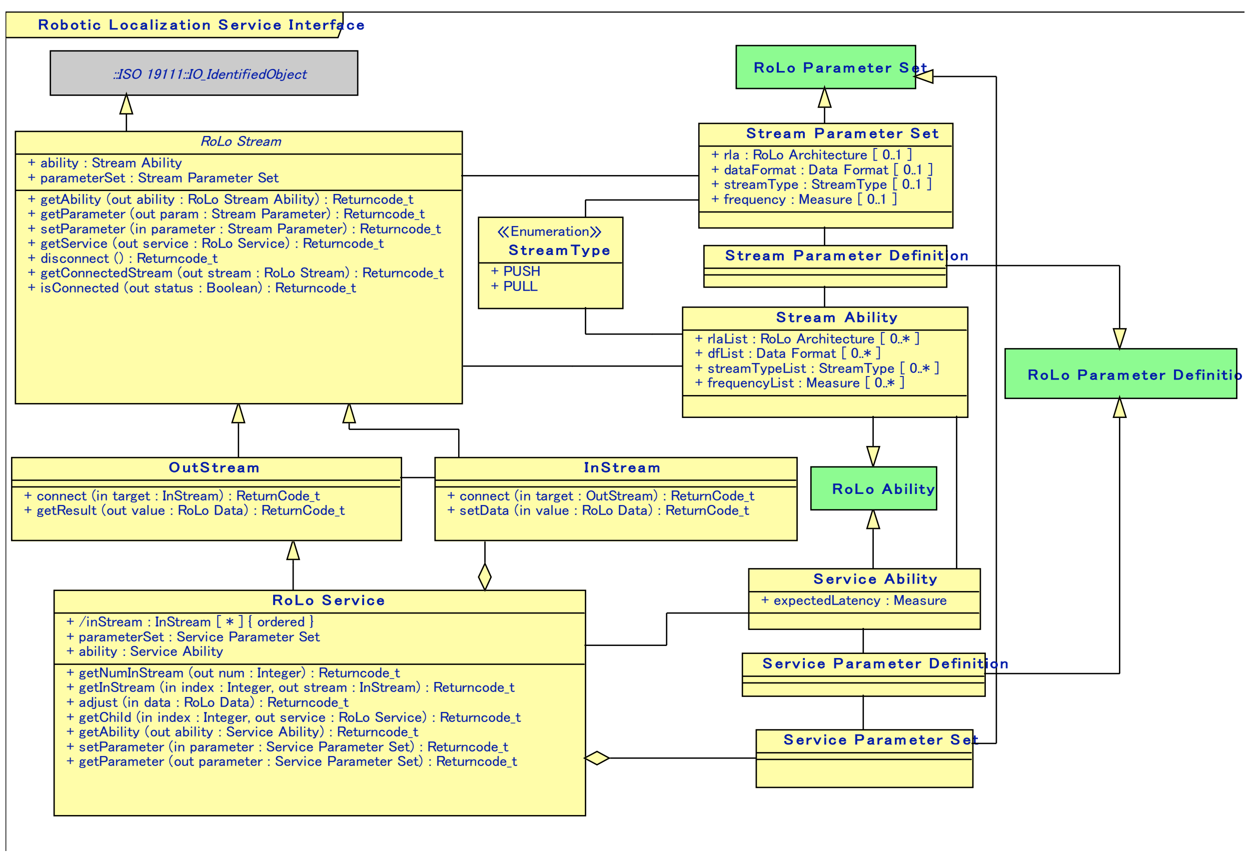


Figure 39 - RoLo Service

Table 158 - StreamType enumeration

PUSH	Indicates that the stream is in data pushing mode, i.e., the stream triggers data passing.
PULL	Indicates that the stream is in data receiving mode, i.e. the counterpart of the stream triggers data passing.

Table 159 - Stream Parameter Definition class

Description: Definitions for the Stream parameter set.
Derived From: RoLo Parameter Definition

Table 160 - Stream Parameter Set class

Description: Represents setting parameters for the stream. If each specific implementation has special parameters, this class may be extended to be added the necessary descriptions.				
Derived From: RoLo Parameter				
Attributes				
rla	RoLo Architecture	C	1	RoLo Architecture to be used.
df	RoLo Data Format	M	1	Data format.
streamType	StreamType	M	1	Stream type.
frequency	Measure [ISO19103]	O	1	Data passing frequency.
Condition: If the RoLo Architecture can be identified by the specified data format, the rlaList attribute may not necessary be specified explicitly.				

Table 161 - Stream Ability class

Description: Ability description for RoLo Stream. The 'parameter' attribute derived from RoLo Ability class shall be based on Stream Parameter class. If each specific implementation has special functionalities, this class may be extended to be added the necessary descriptions.				
Derived From: RoLo Ability				
Attributes				
rlaList	RoLo Architecture	C	N	List of reference to RoLo Architectures supported by this stream
dfList	RoLo Data Format	M	N	List of reference data formats supported by this stream. When every RoLo Architecture in rlaList owns a fixed RoLo Data Format, this attribute can be omitted. Otherwise, this shall be specified. The stream object shall determine whether the combination of specified RoLo Architecture and RoLo Data Format can be used.
streamTypeList	StreamType	M	N	List of supported stream types.

~~frequencyList~~	~~Measure [ISO19103]~~	~~O~~	~~N~~	~~List of supported data passing frequencies.~~
~~Condition: If the RoLo Architecture instances for each of the supported data format can be identified, then rlaList may not necessary be specified explicitly.~~				

13322

Table 162 - AttributeDefinitionSet~~RoLo Stream~~ class

Description: Class that represents a set of attribute definitions.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes				
attrs	AttributeDefinition	M	N	References to AttributeDefinition objects .

~~**Description:** Base abstract class for representing streams.~~				
~~**Derived From:** IO_IdentifiedObject [ISO19111]~~				
~~**Attributes**~~				
~~ability~~	~~Stream Ability~~	~~M~~	~~1~~	~~Holds ability description for this stream.~~
~~parameterSet~~	~~Stream Parameter Set~~	~~M~~	~~1~~	~~Holds the current parameter settings.~~
~~**Operations**~~				
~~getAbility~~	~~Operation for obtaining the ability description for this stream~~			
~~out~~	~~ability~~	~~Stream Ability~~	~~Ability description of this stream.~~	
~~setParameter~~	~~Operation for setting values to the configurable parameters.~~			
~~in~~	~~parameter~~	~~Stream Parameter~~	~~Parameter-value pair to be set. Users may specify values for only a subset of configurable parameters.~~	
~~getParameter~~	~~Operation for obtaining status of configurable parameters.~~			
~~out~~	~~parameter~~	~~RoLo Parameter~~	~~Current status of parameter values.~~	
~~getService~~	~~Returns the service using this stream.~~			
~~out~~	~~service~~	~~RoLo Service~~	~~Reference to the service using this stream.~~	
~~getConnectedStream~~	~~Obtain currently connected stream.~~			
~~out~~	~~stream~~	~~RoLo Stream~~	~~Stream that is currently connected to this stream. If no stream is connected, ERROR is returned. Otherwise, OK is returned.~~	
~~isConnected~~	~~Check whether this stream is connected to other stream.~~			
~~out~~	~~status~~	~~Boolean~~	~~If true, connected. Otherwise, not connected.~~	
~~disconnect~~	~~Disconnects this stream from the currently connected stream.~~			

13322

Table 163 - AttributeBase~~OutStream~~ class

Description: Base abstract class for different types of Attribute classes.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes				
def	AttributeDefinition	M	1	Reference to an AttributeDefinition object indicating definition for this attribute.
~~**Description:** Represents output streams.~~				

~~**Derived From:** RoLo Stream~~			
~~**Operations**~~			
~~connect~~	~~Connect to given stream.~~		
~~in~~	~~target~~	~~InStream~~	~~Target stream to connect.~~
~~getResult~~	~~Obtain localization result.~~		
~~out~~	~~value~~	~~RoLo Data Set~~	~~Resulting localization data.~~

13322

Table 164 - Attribute~~InStream~~ class

Description: Represents a single attribute. This is a template class with type argument T which denotes the type of attribute value. The type argument T shall be consistent with the value of 'type' attribute in AttributeDefinition object referred by the 'def' attribute derived from AttributeBase class.				
Derived From: AttributeBase				
Attributes				
val	T	M	1	Value of this attribute.
~~**Description:** Represents input streams.~~				
~~**Derived From:** RoLo Stream~~				
~~**Operations**~~				
~~connect~~	~~Connect to given stream.~~			
~~in~~	~~target~~	~~OutStream~~	~~Target stream to connect.~~	
~~setData~~	~~Set data to this stream.~~			
~~in~~	~~value~~	~~RoLo Data Set~~	~~Localization data to be set to this stream.~~	

13322

Table 165 - Parameter~~Service Parameter Definition~~ class

Description: Represents a single parameter. A parameter is an attribute that may be configurable. This is a template class with type arguments T which denotes the type of parameter value.				
Derived From: Attribute <T>				
Attributes				
isConfigurable	Bool	O	1	Flag to show whether this parameter is configurable or not. If omitted, assumed to True. When this value is set to False, this parameter is not configurable.
~~**Description:** Definitions for the Service parameter set.~~				
~~**Derived From:** RoLo Parameter Definition~~				

Table 166 - ParameterOverDomain~~Service Parameter Set~~ class

Description: Represents a parameter whose value domain is defined. This is a template class with type arguments T and TD, where T denotes the type of parameter value and TD denotes the type to show domain of the parameter value.
Derived From: Parameter <T>
Attributes

13322

domain	TD	M	1	Domain of parameter value.
~~**Description:** Base class for holding parameters for RoLo Service. If each specific implementation has special parameters, this class may be extended to be added the necessary descriptions.~~				
~~**Derived From:** RoLo Parameter Set~~				

Table 167 - Interval~~Service Ability~~ class

Description: Class for indicating an interval. Note that an interval is sometimes referred as a 'range'.				
Derived From: (none)				
Attributes				
val	T	M	1	Value of this attribute.
~~**Description:** Ability description for RoLo Service. The 'parameter' attribute derived from RoLo Ability class shall be based on Service Parameter class. If each specific service implementation has special functionalities, this class may be extended to be added the necessary descriptions.~~				
~~**Derived From:** RoLo Ability~~				
~~**Attributes**~~				
~~expectedLatency~~	~~Measure [ISO19103]~~	~~M~~	~~1~~	~~Expected latency. This ability descriptor is especially useful for Robotic Localization Service users.~~
~~inStreamAbility~~	~~Stream Ability~~	~~O~~	~~N~~	~~Ability descriptions for the input stream in this service.~~
~~outStreamAbililty~~	~~Stream Ability~~	~~M~~	~~1~~	~~Ability descriptions for the output stream in this service.~~

Table 168 - IntervalParameter~~RoLo Service~~ class

Description: A parameter whose value domain is defined as an interval. This is a template class with type argument T. The type argument TD from ParameterOverDomain is deduced to be class Interval <T>.					
Derived From: ParameterOverDomain<T, Interval>					
~~**Description:** Interface for the Robotic Localization Service.~~					
~~**Derived From:** OutStream~~					
~~**Attributes**~~					
~~inStream~~		~~InStream~~	~~O~~	~~N ord~~	~~An ordered list of input stream objects.~~
~~**Operations**~~					
~~getNumInStream~~		~~Returns the number of input streams.~~			
~~out~~	~~num~~	~~Integer~~	~~Number of input streams equipped in this module.~~		
~~getInStream~~		~~Returns the input stream of given index.~~			
~~in~~	~~index~~	~~Integer~~	~~Index number indicating the target InStream.~~		
~~out~~	~~stream~~	~~InStream~~	~~Reference to the target InStream.~~		
~~adjust~~		~~Method for adjusting localization results. For elements not required for adjustment, NULL element should be specified.~~			
~~in~~	~~data~~	~~RoLo Data~~	~~Data to be used for initialization or adjustment. Adjusts every element at once.~~		
~~getChild~~		~~Return the service connected to the InStream of this service module.~~			
~~in~~	~~index~~	~~Integer~~	~~Index number indicating the target child service.~~		

13322

~~out~~	~~service~~	~~RoLo Service~~	~~Reference to the target child service~~
~~getAbility~~		~~Obtain ability description for this service.~~	
~~out~~	~~ability~~	~~Service Ability~~	~~Ability description for this service~~
~~setParameter~~		~~Set parameter values for this service.~~	
~~in~~	~~parameter~~	~~Service Parameter~~	~~Parameter values to be set.~~
~~getParameter~~		~~Obtain current parameter values for this service.~~	
~~out~~	~~parameter~~	~~Service Parameter~~	~~Current parameter settings.~~

Table 169 - SetParameter class

Description: A parameter whose value domain is defined as a set of values. This is a template class with type argument T. The type argument TD from class ParameterOverDomain is deduced to be a set.
Derived From: ParameterOverDomain<T, Set<T>>

Table 170 - ParameterValueBase class

Description: Base abstract class for different types of ParameterValue class.
Derived From: (none)

Table 171 - ParameterValue class

Description: A Class that represents values for parameters. This is a template class with type argument T which denotes the type of the parameter value.				
Derived From: AttributeBase				
Attributes				
val	T	M	1	Value of the parameter.
param	Parameter	M	1	Reference to a Parameter object this parameter value is for. The template argument T for this class shall match the template argument of the referred Parameter object.

13322

Table 172 - AttributeSet class

Description: Represents a set of attributes or parameters.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes				
def	AttributeDefinitionSet	M	1	Definition of this attribute set.
attrs	Attribute	O	N	Set of attributes that is contained in this attribute set.

13322 **Table 173 - Ability class**

Description: Describes module ability.	
Derived From: AttributeSet	

12916, 13116, 13322, 13439, 13998, 14003, 14013, 14015

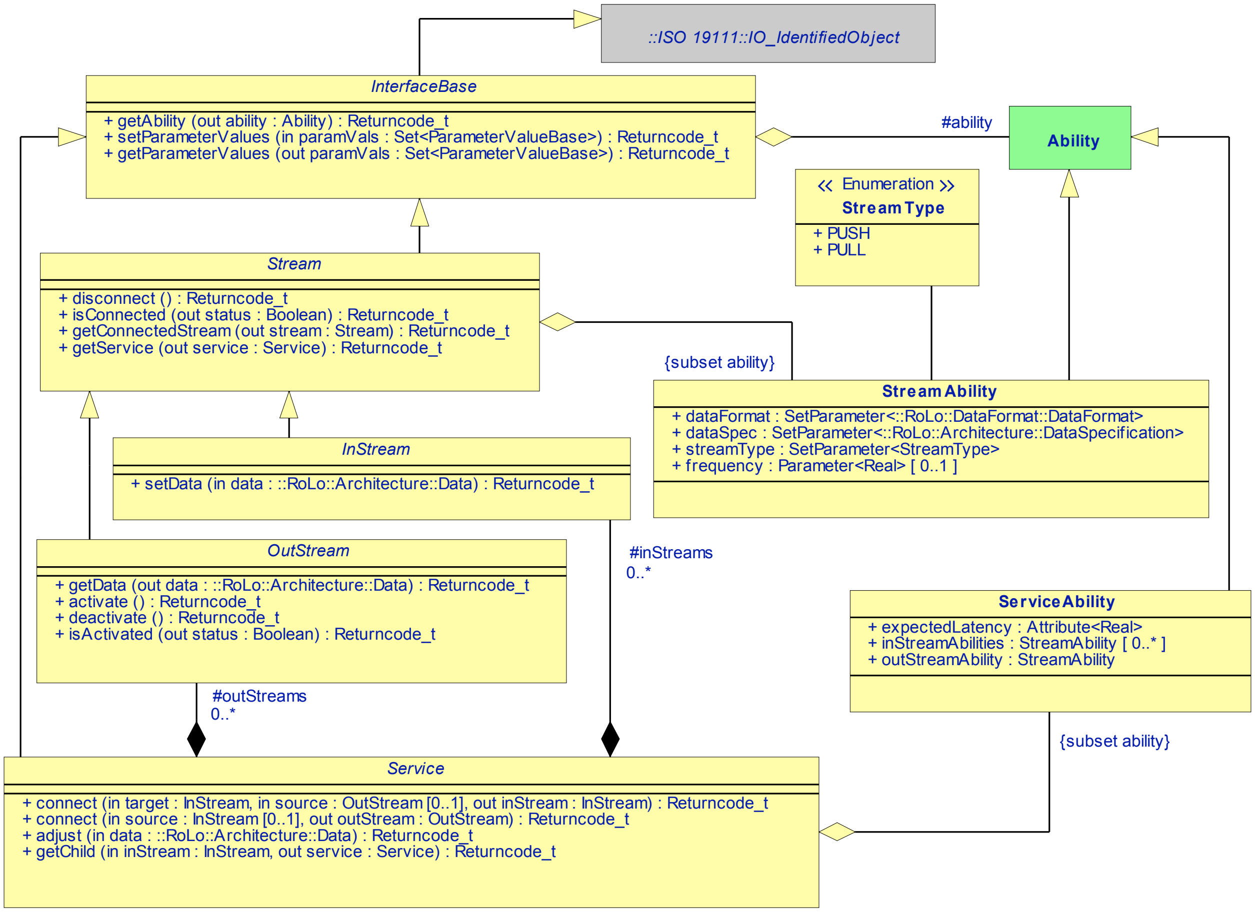


Figure 40 - RoLo Service

13439, 14011

Table 174 - InterfaceBase class

Description: Abstract class for interfacing objects.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes				
ability (protected)	Ability	M	1	Reference to an ability description for this object. The referred RoLo ability's attribute 'target' shall refer to this object.
Operations				
getAbility	Operation for obtaining the ability description for this stream			
out	ability	Ability	Ability description of this stream.	
setParameterValues	Operation for setting values to the configurable parameters.			
in	paramVals	Set<ParameterValueBase>	Set of parameter values to be set. If some nonexistent or inconfigurable parameters were specified, UNSUPPORTED_PARAMETER or BAD_PARAMETER will be returned respectively.	
getParameterValues	Operation for obtaining status of configurable parameters.			
out	paramVals	Set<ParameterValueBase>	Current status of parameter values.	

14012 **Table 175 - StreamType enumeration**

PUSH	Indicates that data passing is performed in PUSH mode, i.e. OUT side triggers data passing.
PULL	Indicates that data passing is performed in PULL mode, i.e. IN side triggers data passing.

12916, 13116, 13322 **Table 176 - StreamAbility class**

Description: Ability description for RoLo streams. If each RoLo stream has special functionalities, this class may be extended to be added necessary descriptions.				
Derived From: Ability				
Attributes				
dataSpec	SetParameter<RoLo::Architecture::DataSpecification>	M	1	Parameter for DataSpecification supported by this stream
dataFormat	SetParameter<RoLo::DataFormat::DataFormat>	M	1	Parameter for data formats supported by this stream.
streamType	SetParameter<StreamType>	M	1	Parameter for supported stream types.
frequency	SetParameter<Real>	O	1	Parameter for data passing frequency in PUSH mode. The unit for this attribute is Hz. If unnecessary (for example, a RoLo out stream which only supports PULL type data passing), this parameter may be omitted.

13322, 13439, 13998, 14011, 14015 **Table 177 - Stream class**

Description: Abstract class for representing RoLo streams.			
Derived From: InterfaceBase			
Operations			
getService	Returns the service owning this stream.		
out	service	Service	Reference to the service owning this stream.
getConnectedStream	Obtain currently connected stream, if any.		
out	streams	Stream	Reference to the stream that is currently connected to this stream. If no stream is connected, ERROR is returned. Otherwise, OK is returned. When the connection is performed without 'source' argument, this may not work (See description on Service class for details).
isConnected	Check whether this stream is connected to other stream.		
out	status	Boolean	If connected true, otherwise false.
disconnect	Disconnects this stream from the currently connected stream.		
Note: Values for the attribute 'ability' which is derived from parent class shall be limited to instances of StreamAbility or its inherited classes.			

12916, 12940, 13439, 13998 **Table 178 - OutStream class**

Description: Represents output streams.

12916, 12940, 13439, 13998, 14003, 14013

Derived From: Stream			
Operations			
getData	Obtain localization result.		
out	data	Data (RoLo::Architecture)	Resulting localization data.
activate	Activate stream output. Only meaningful on PUSH mode.		
deactivate	Deactivate stream output. Only meaningful on PUSH mode.		
isActivated	Query whether this stream is activated or not.		
out	status	Boolean	If activated true, otherwise false.

12916, 12940, 13439

Table 179 - InStream class

Description: Represents input streams.			
Derived From: Stream			
Operations			
setData	Set data to this stream.		
in	val	Data (RoLo::Architecture)	Localization data to be set to this stream.

13322

Table 180 - ServiceAbility class

Description: Ability description for RoLo Service. If each specific service implementation has special functionalities, this class may be extended to be added the necessary descriptions.				
Derived From: Ability				
Attributes				
expectedLatency	Attribute<Real>	M	1	Expected latency. This ability descriptor is especially useful for Robotic Localization Service users. The unit for this attribute is milliseconds.
inStreamAbilities	StreamAbility	O	N	Ability descriptions for the input streams in this service.
outStreamAbililty	StreamAbility	M	1	Ability descriptions for the output stream in this service.

12916, 13439, 14003, 14015

Table 181 - Service class

Description: Interface for the robotic localization service.				
Derived From: InterfaceBase				
Attributes				
inStreams (protected)	InStream	O	N	An ordered list of RoLo input streams owned by this service.
outStreams (protected)	OutStream	M	N	An ordered list of RoLo output streams owned by this service.
Operations				
connect	Establish connection from output stream to input stream. (OUT service initiates the connection)			

12916, 13439, 14003, 14015

in	target	InStream	Reference to a RoLo input stream to be connected. This target reference shall be obtained through getAbility method.
in	source [0..1]	OutStream	Reference to the RoLo output stream that is connecting. This argument is optional. When this argument is omitted, 'getChildren' method may not work
out	inStream	InStream	Reference to a RoLo input stream to be used for further manipulation of the established connection. Note that, this reference may be pointing to a different object as the one given as input argument. Users shall use the returned reference, not the one obtained through getAbility method.
connect	Establish connection from input stream to output stream. (IN service initiates the connection)		
in	source [0..1]	InStream	Reference to the RoLo input stream that is connecting. This argument is optional. However, when data passing is to be done in PUSH mode, this argument cannot be omitted. Also, when this argument is omitted, 'getChildren' method may not work.
out	outStream	OutStream	Reference to a RoLo output stream object to be used for further manipulation of the established connection.
adjust	Method for adjusting localization results. For elements not required for adjustment, don't-care element should be specified.		
in	data	Data (RoLo::Architecture)	Data to be used for initialization or adjustment. Adjusts every element at once.
getChildren	Obtain services connected to input streams of this service.		
out	services	List<Service>	Ordered list of services connected to the input streams of this service.
Note: When 'getAbility' method is called, RoLo stream shall return an ability description that contains ability descriptors for the service and also the descriptors for the RoLo streams that this service holds. This shall include the descriptors for each of the input streams. For the out stream, only a single descriptor is sufficient. Values for the attribute 'ability' which is derived from parent class shall be limited to instances of ServiceAbility or its inherited classes.			

12917

Using RoLo Service

Here we show several non-mandatory steps and sequence diagrams as examples. Typical steps of using RoLo Services can be listed as following:

1. (optional) Obtain ability description by calling 'getAbility' method toward RoLo service. An ability description obtained from RoLo service also includes descriptions on its streams. This step can be omitted if users already have sufficient information such by reading reference manuals.
2. (optional) Set up service and/or stream parameters through calling 'setParameterValues' method. If the default settings are sufficient or if there exists no parameter to be configured, this step can be omitted. In complicated cases, users may need to repeatedly call 'setParameterValues' and 'getParameterValues' to set and to confirm parameter changes.
3. Establish connection.
4. (optional) Set up initial position data by calling 'adjust' method with necessary data.
5. Perform data passing.
6. (optional) Occasionally, perform adjustment if necessary. Adjustment is an act to provide auxiliary information to the target module for improving the localization process..

12917 7. Disconnect the connection.

Figure 41 to 45 show sequences of typical steps on using RoLo service. Note that in step 3, connection establishment can be initiated from two side; either from the service that outputs data (OUT service) or from the service that accepts data inputs (IN service). Figure 42 and Figure 43 show typical connection sequences in both cases. Note that, disconnection of the established connection (step 7) can be performed from both sides regardless of which side initiated the connection (Figure 45).

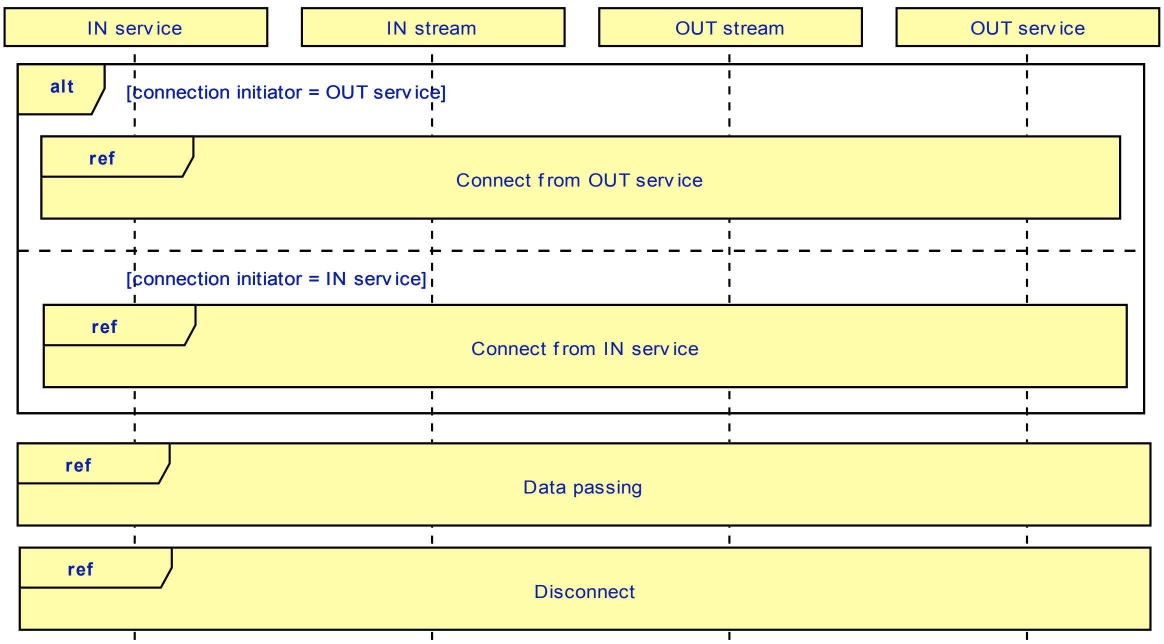


Figure 41 - Sequence Diagram of Typical RoLo Service Usage

12917

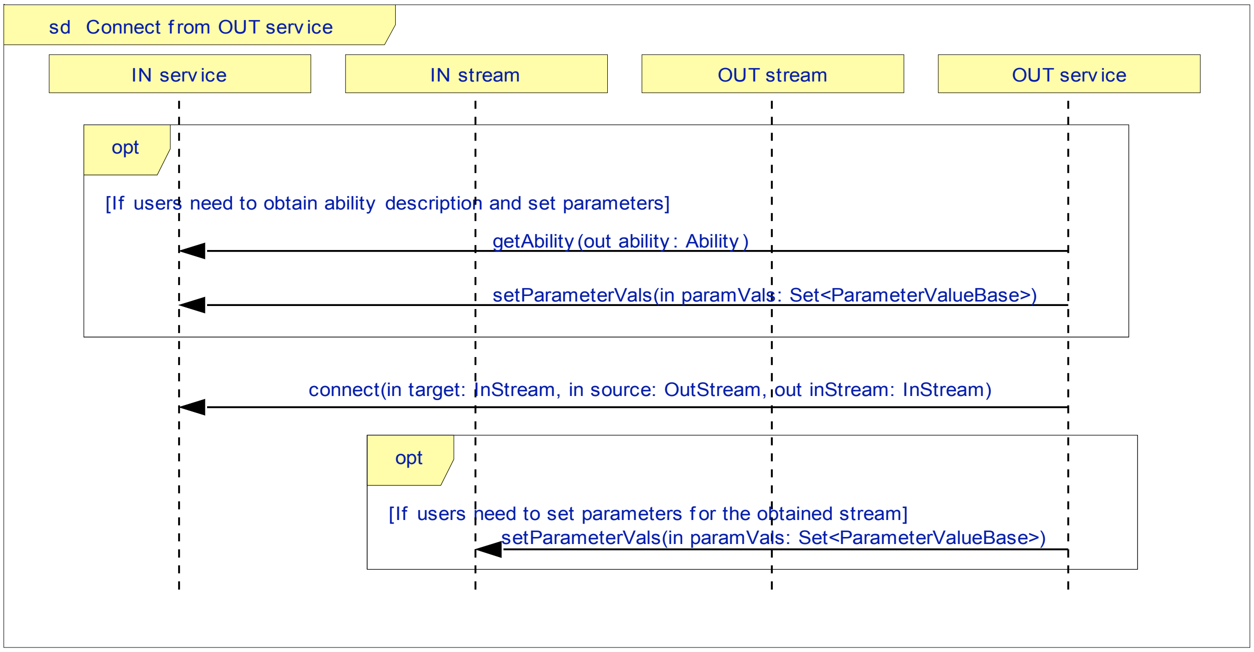


Figure 42 - Sequence Diagram of Connection Establishment from OUT Service

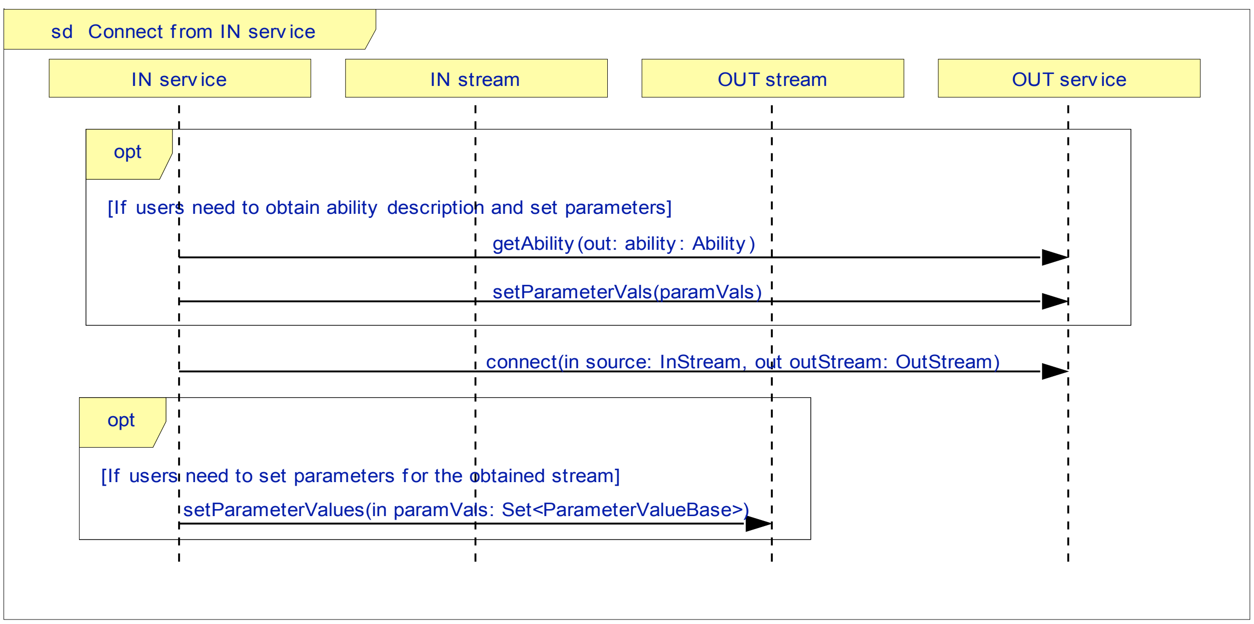


Figure 43 - Sequence Diagram of Connection Establishment from IN Service

12917

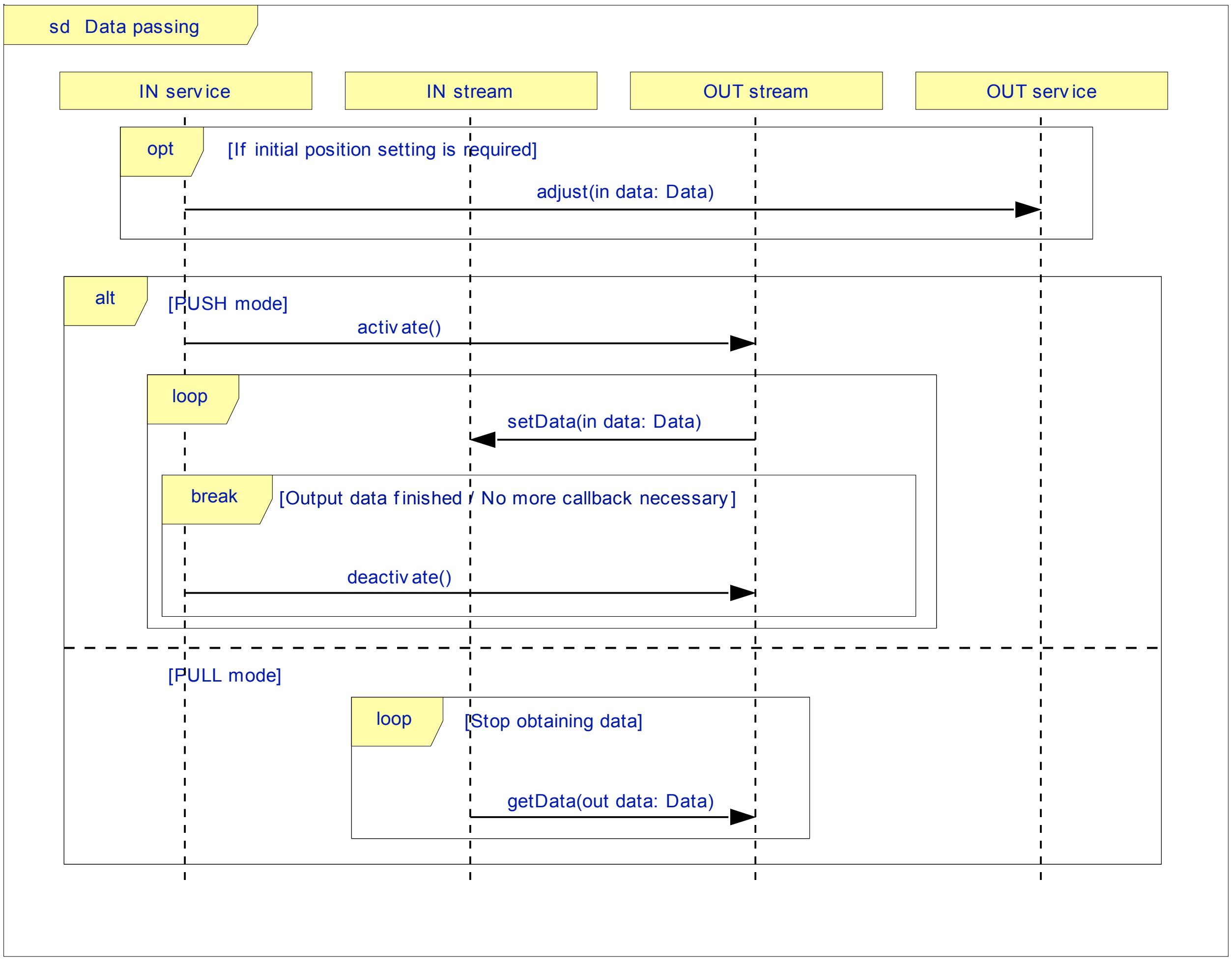


Figure 44 - Sequence Diagram of Data Passing

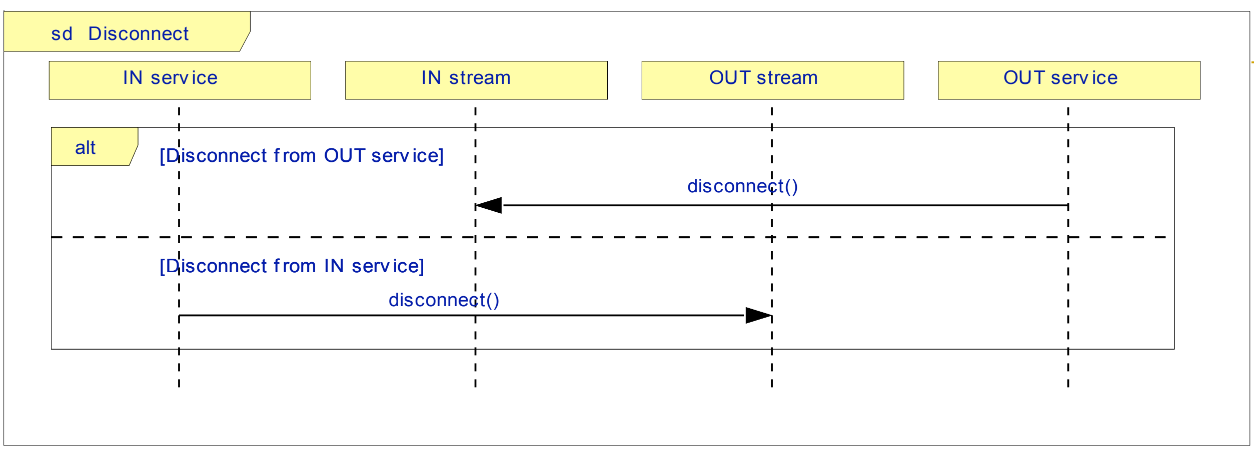


Figure 45 - Sequence Diagram of Disconnecting Connection

12917 Another factor that needs consideration is the type of data passing. In this specification, two data passing types are provided as elements of StreamType enumeration: PUSH mode (OUT side triggers data passing) and PULL mode (IN side triggers data passing). For example, most GPS receivers output data in PUSH mode, that is, measurement results are outputted continuously in some frequency. These two types of data passing can be performed regardless of which side initiates connection, as far as both modules have the ability to perform data passing in the specified type. Figure 44 shows typical steps for performing data passing for the two directions. As can be seen from the sequence, in PULL mode, the IN service triggers data passing by calling 'getData' method. And in PUSH mode, the OUT service triggers data passing by 'setData' method.

PUSH type data passing can also be understood as a callback from OUT side to IN side. Thus, when using PUSH mode and when connection is established from IN side, the 'source' argument cannot be omitted. Without this, the RoLo output stream on OUT side cannot know where to make callbacks for data passing. However, when connection is established from OUT side, this 'source' argument is not required for the sake of making callbacks, as the RoLo input stream is given back as an 'inStream' argument.

13115, 14000

9. Platform Specific Model

9.1 C++ PSM

In this section, we show a PSM in C++ language based on the PIM described in section 7. This PIM-PSM mapping is based on the following rules:

- The return values of methods are assumed to be mapped as exceptions. Thus, in this PSM, no explicit description is given.
- When methods had only a single 'out' argument, it was mapped as return value of the corresponding function.
- The 'in' arguments to methods were mapped as method arguments with the 'const' modifier.
- Arguments which were based on non-primitive types are passed by reference.
- An attribute or an argument that is marked to occur more than once and is marked as unordered is mapped to '::std::list'. If marked as ordered, it is mapped to '::std::vector'.
- When an attribute is shown as an aggregation or as a derived attribute, or when an argument indicates a reference to other object, it is mapped as a pointer.
- CharacterString is mapped as '::std::string'.

The following shows the resulting C++ header files.

```
// $Id: Returncode_t.hpp,v 1.3 2009/06/20 06:18:43 nishio Exp $

#pragma once

namespace RoLo
{
  enum Returncode_t {
    OK,
    ERROR,
    BAD_PARAMETER,
    UNSUPPORTED_PARAMETER,
    UNSUPPORTED_OPERATION,
    TIMEOUT
  };

}
// $Id: Architecture.hpp,v 1.3 2009/06/20 06:18:42 nishio Exp $
```

12916

```
#pragma once

#include <RLS/RelativeCRS.hpp>
#include <RLS/MobileCRS.hpp>
#include <RLS/MobileOperation.hpp>
#include <RLS/Identity.hpp>
#include <RLS/ErrorType.hpp>
#include <RLS/Error.hpp>
#include <RLS/RoLoArchitecture.hpp>
#include <RLS/RoLoDataOperation.hpp>
// $Id: RelativeCRS.hpp,v 1.8 2009/06/20 17:51:30 nishio Exp $

#pragma once
#include <ISO19111/SC_CoordinateReferenceSystem.hpp>
#include <ISO19111/CD_Datum.hpp>
#include <RLS/RoLoArchitecture.hpp>

namespace RoLo
{
```

12916

```
  namespace Architecture
  {
    class RelativeCRS
      : public ::ISO19111::SC_EngineeringCRS
    {
    };
```

13106, 14016

```
    class RelativeDatum
      : public ::ISO19111::CD_EngineeringDatum
    {
    };
```

13106

```
    class StaticRelativeCRS
      : public RelativeCRS
    {
    };
```

13106

```
    class StaticRelativeCartesianCRS
      : public StaticRelativeCRS
    {
    };
```

13106

```
    class StaticRelativePolarCRS
      : public StaticRelativeCRS
    {
    };
```

13106

```
    class StaticRelativeDatum
```

```
13106, 13116, 13131          : public RelativeDatum
        {
        public:
            DataSpecification* dataSpec;
            Data base;
        };

13106       class DynamicRelativeCRS
            : public RelativeCRS
        {
        };

13106       class DynamicRelativeDatum
            : public RelativeDatum
        {
        };

    }
}
// $Id: MobileCRS.hpp,v 1.5 2009/06/20 06:52:40 nishio Exp $

#pragma once
#include <ISO19111/CS_CoordinateSystem.hpp>
#include <ISO19111/CD_Datum.hpp>
#include <RLS/RelativeCRS.hpp>
#include <RLS/Service.hpp>

namespace RoLo
{
12916   namespace Architecture
    {
13106       class MobileCRS
            : public DynamicRelativeCRS
        {
        };

        class MobileCartesianCRS
            : public MobileCRS
        {
        };

        class MobilePolarCRS
            : public MobileCRS
        {
        };

12916       class MobileDatum
```

12916, 13106, 13436, 14003, 14014

```
      : public DynamicRelativeDatum
    {
    public:
      const ::RoLo::Interface::InStream& getInStream();
    protected:
      ::RoLo::Interface::InStream inStream;
    };

  }
}
// $Id: MobileOperation.hpp,v 1.5 2009/06/20 06:18:43 nishio Exp $

#pragma once
#include <ISO19111/CC_Operation.hpp>
#include <ISO19111/SC_CoordinateReferenceSystem.hpp>
#include <ISO19111/CD_Datum.hpp>

namespace RoLo
{
```

12916

```
  namespace Architecture
  {
    class MobileOperation
      : public ::ISO19111::CC_Transformation
    {
    };

    class Mobile2StaticOperation
      : public MobileOperation
    {
    public:
      MobileCRS *source;
      ISO19111::SC_CRS *target;
    };

    class Staic2MobileOperation
      : public MobileOperation
    {
    public:
      ISO19111::SC_CRS *source;
      MobileCRS *target;
    };

    class Mobile2MobileOperation
      : public MobileOperation
    {
    public:
      MobileCRS *source, *target;
```

```
    };

  }
}
// $Id: Identity.hpp,v 1.8 2009/06/20 06:18:43 nishio Exp $

#pragma once
#include <string>
#include <vector>
#include <ISO19111/IO_IdentifiedObject.hpp>
#include <ISO19111/CS_CoordinateSystem.hpp>
#include <ISO19111/SC_CoordinateReferenceSystem.hpp>
#include <ISO19111/CD_Datum.hpp>

namespace RoLo
{
12916
  namespace Architecture
  {
    class IdentityCS
      : public ::ISO19111::CS_CoordinateSystem
    {
    };

    class NumericIdentityCS
      : public IdentityCS
    {
    };

13108
    class SymbolicIdentityCS
      : public IdentityCS
    {
    };

    class IdentityDatum
      : public ::ISO19111::CD_Datum
    {
    };

    class IdentityCRS
      : public ::ISO19111::SC_SingleCRS
    {
    };

    class NumericIdentityCRS
      : public IdentityCRS
    {
    };
```

13108

```
    class SymbolicIdentityCRS
      : public IdentityCRS
    {
    };
```

13108

```
    class DirectSymbol
      : public ::ISO19111::IO_IdentifiedObjectBase
    {
    public:
      ::std::vector<std::string> coords;
      SymbolicIdentityCRS *crs;
    };

    class SymbolRef
      : public ::ISO19111::IO_IdentifiedObjectBase
    {
    public:
      DirectSymbol *point;
    };
```

14004

```
    class SymbolicPosition
      : public ::ISO19111::IO_IdentifiedObjectBase
    {
    public:
      DirectSymbol *direct;
      SymbolRef *indirect;
    };

  }
}
// $Id: ErrorType.hpp,v 1.4 2009/06/20 06:18:43 nishio Exp $
#pragma once

#pragma once
#include <ISO19111/IO_IdentifiedObject.hpp>

namespace RoLo
{
```

12916

```
  namespace Architecture
  {
```

13317

```
    class ErrorType
      : public ::ISO19111::IO_IdentifiedObject
    {
      ErrorType *baseType;
    };
```

```
    class ErrorTypeOperation
      : public ::ISO19111::IO_IdentifiedObject
    {
    public:
      ErrorType *source, *target;
    };

  }
}
```

12916, 13129

```
// $Id: ErrorBase.hpp,v 1.1 2009/06/20 06:18:42 nishio Exp $

#pragma once
#include <RLS/ErrorType.hpp>

namespace RoLo
{
  namespace Architecture
  {
    class ErrorType;

    class Error
      : public ::ISO19111::IO_IdentifiedObjectBase
    {
    public:
      ErrorType *errType;
    };

  }
}
// $Id: Error.hpp,v 1.7 2009/06/20 06:18:42 nishio Exp $

#pragma once
#include <ISO19103/Primitive.hpp>
#include <ISO19111/IO_IdentifiedObject.hpp>
#include <RLS/ErrorType.hpp>
#include <RLS/ErrorBase.hpp>

namespace RoLo
{
```

12916

```
  namespace Architecture
  {
    class Reliability
      : public Error, public ::ISO19103::Probability
    {
    };

    class ErrorDistribution
```

```
        : public Error
    {
    };

    class Matrix
    {
    public:
        int nRow, nCol;
        ::std::vector< ::ISO19103::Number > vals;
    };

    class CovarianceMatrix
        : public Matrix
    {
    };

    class Gaussian
        : public ErrorDistribution
    {
    public:
        CovarianceMatrix cov;
    };

13998    class UniformGaussian
        : public Gaussian
    {
    };

    class MixtureModel
        : public ErrorDistribution
    {
    };

13185    class WeightedModel
        : public ::ISO19111::IO_IdentifiedObjectBase
    {
    public:
        PositionElement posElem;
        ::ISO19103::Probability weight;
    };

13185, 13437    class LinearMixtureModel
        : public MixtureModel
    {
    public:
        ::std::vector<WeightedModel> models;
    };
```

```
    class MixtureOfGaussian
      : public LinearMixtureModel
    {
    };
```

13185

```
    class ParticleSet
      : public LinearMixtureModel
    {
    };

  }
}
// $Id: RoLoArchitecture.hpp,v 1.8 2009/06/20 06:18:43 nishio Exp $

#pragma once
#include <vector>
#include <ISO19107/CoordinateGeometry.hpp>
#include <ISO19111/IO_IdentifiedObject.hpp>
#include <ISO19111/CS_CoordinateSystem.hpp>
#include <ISO19111/SC_CoordinateReferenceSystem.hpp>
#include <RLS/ErrorType.hpp>
#include <RLS/ErrorBase.hpp>
#include <RLS/Identity.hpp>

namespace RoLo
{
```

12916

```
  namespace Architecture
  {
```

14004

```
    class Position
      : public ISO19111::IO_IdentifiedObjectBase
    {
    public:
      SymbolicPosition* symbolic;
      ISO19107::GM_Position* numeric;
    };

    class ElementSpecification
      : public ::ISO19111::IO_IdentifiedObject
    {
    };
```

12929, 13187

```
    class PositionElementSpecification
      : public ElementSpecification
    {
    public:
      ::ISO19111::SC_CRS *crs;
```

12929, 13187

```
        ErrorType *errType;
    };
```

12939, 13129

```
    class ErrorElementSpecification
        : public ElementSpecification
    {
    public:
        ::std::vector<PositionElementSpecification*> posSpecRefs;
        ErrorType *errType;
    };

    class Element
        : public ::ISO19111::IO_IdentifiedObjectBase
    {
    public:
        ElementSpecification *spec;
    };

    class PositionElement
        : public Element
    {
    public:
        Position pos;
        Error err;
    };

    class ErrorElement
        : public Element
    {
    public:
        Error err;
    };
```

13116

```
    class DataSpecification
        : public ::ISO19111::IO_IdentifiedObject
    {
    public:
        ::std::vector<ElementSpecification> elemSpecs;
    };
```

13116

```
    class Data
        : public ::ISO19111::IO_IdentifiedObjectBase
    {
    public:
        DataSpecification *spec;
        ::std::vector<Element> elems;
    };
```

```
13187     class DontCare
    {
    };

    class NULLCS
      : public DontCare, public ::ISO19111::CS_CoordinateSystem
    {
    };

    class NULLCRS
      : public DontCare, public ::ISO19111::SC_CRS
    {
    };

    class NULLDatum
      : public DontCare, public ::ISO19111::CD_Datum
    {
    };

    class NULLErrorType
      : public DontCare, public ErrorType
    {
    };

    class NULLElementSpecification
      : public DontCare, ElementSpecification
    {
    };

  }
}
// $Id: RoLoDataOperation.hpp,v 1.8 2009/06/20 17:46:15 nishio Exp $
#pragma once

#include <vector>
#include <ISO19111/IO_IdentifiedObject.hpp>
#include <ISO19111/CC_Operation.hpp>
#include <RLS/RoLoArchitecture.hpp>

namespace RoLo
{
12916   namespace Architecture
  {
    class PositionElementOperation
      : public ::ISO19111::IO_IdentifiedObject
    {
```

```
    public:
      PositionElementSpecification *source, *target;
    };

    class PositionElementConcatenatedOperation
      : public PositionElementOperation
    {
    public:
      ::std::vector<PositionElementOperation*> childOperations;
    };

    class PositionElementSingleOperation
      : public PositionElementOperation
    {
    public:
      ::ISO19111::CC_CoordinateOperation *usesOperation;
      ErrorTypeOperation *usesErrTypeOperation;
    };
```

13116

```
    class DataOperation
      : public ::ISO19111::IO_IdentifiedObject
    {
    public:
      DataSpecification *source, *target;
    };

    class DataConcatenatedOperation
      : public DataOperation
    {
    public:
      ::std::vector<DataOperation*> childOperations;
    };

    class DataSingleOperation
      : public DataOperation
    {
    };

    class DataTransformation
      : public DataSingleOperation
    {
    public:
      ::std::vector<PositionElementOperation*> usesOperations;
    };

    class DataMappingOperation
      : public DataSingleOperation
```

13129

```
    {
    public:
      ::std::vector<ElementSpecification*> sourceElemSpecs, targetElemSpecs;
    };

  }
}
// $Id: DataFormat.hpp,v 1.5 2009/06/20 06:18:42 nishio Exp $

#pragma once
#include <ISO19111/IO_IdentifiedObject.hpp>
#include <RLS/RoLoArchitecture.hpp>

namespace RoLo
{
```

12916

```
  namespace DataFormat
  {
    class DataFormat
      : public ISO19111::IO_IdentifiedObject
    {
    };

    class EncodingRule
      : public DataFormat
    {
    };
```

12916, 13116

```
    class SpecificDataFormat
      : public DataFormat
    {
    public:
      ::RoLo::Architecture::DataSpecification *dataSpec;
    };

    class UserDefinedDataFormat
      : public SpecificDataFormat
    {
    };

    class CommonDataFormat
      : public SpecificDataFormat
    {
    };

  }
}
// $Id: Interface.hpp,v 1.2 2009/06/20 06:18:43 nishio Exp $
```

```
12916 #pragma once

#include <RLS/Ability.hpp>
#include <RLS/Service.hpp>
// $Id: Ability.hpp,v 1.9 2009/06/21 16:51:56 nishio Exp $

#pragma once
#include <list>
#include <ISO19103/Primitive.hpp>
#include <ISO19111/IO_IdentifiedObject.hpp>
#include <ISO19115.hpp>
#include <RLS/RoLoArchitecture.hpp>
#include <RLS/Error.hpp>

namespace RoLo
{
12916   namespace Interface
  {
13322     class AttributeDefinition
      : public ::ISO19111::IO_IdentifiedObject
    {
    public:
      ::ISO19115::RS_Identifier type;
      ::ISO19103::UnitOfMeasure unit;
    };

13322     class AttributeDefinitionSet
      : public ::ISO19111::IO_IdentifiedObject
    {
      ::std::list<AttributeDefinition*> attrs;
    };

13322     class AttributeBase
      : public ::ISO19111::IO_IdentifiedObject
    {
    public:
      const AttributeDefinition *def;
    };

13322     template <typename T>
    class Attribute
      : public AttributeBase
    {
    public:
      T val;
    };
```

13322

```
    template <typename T>
    class Parameter
      : public Attribute<T>
    {
    public:
      bool isConfigurable;
    };

    template <typename T, typename TD>
    class ParameterOverDomain
      : public Attribute<T>
    {
    public:
      TD domain;
    };

    template <typename T>
    class Interval
    {
    public:
      T min, max;
      bool minInc, maxInc;
    };

    template <typename T>
    class IntervalParameter
      : public ParameterOverDomain< T, Interval<T> >
    {};

    template <typename T>
    class SetParameter
      : public ParameterOverDomain< T, ::std::list<T> >
    {};
```

13322

```
    class AttributeSet
      : public ::ISO19111::IO_IdentifiedObject
    {
    public:
      AttributeDefinitionSet *def;
      ::std::list<AttributeBase> attrs;
    };

    class Ability
      : public AttributeSet
    {
    public:
      const class InterfaceBase* targetID;
```

```
    };

    class ParameterValueBase
    {
    };

    template <typename T>
    class ParameterValue
      : public ParameterValueBase
    {
    public:
      T val;
    };

  }
}
// $Id: Service.hpp,v 1.10 2009/06/20 06:48:15 nishio Exp $
```

13106

```
#pragma once
#include <vector>
#include <list>
#include <ISO19111/IO_IdentifiedObject.hpp>
#include <ISO19115.hpp>
#include <RLS/Ability.hpp>
#include <RLS/DataFormat.hpp>
#include <RLS/RoLoArchitecture.hpp>

namespace RoLo
{
  namespace Interface
  {
    enum StreamType {
      PUSH,
      PULL
    };
```

13116, 13322

```
    class StreamAbility
      : public Ability
    {
    public:
      SetParameter< ::RoLo::DataFormat::DataFormat> dataFormat;
      SetParameter< ::RoLo::Architecture::DataSpecification > dataSpec;
      SetParameter<StreamType> streamType;
      SetParameter<double> frequency;
    };
```

13439

```
    class InterfaceBase
```

13439, 14003, 14014

```
        : public ::ISO19111::IO_IdentifiedObject
    {
    public:
        const Ability& getAbility();
        void setParameterValues(const ::std::list<ParameterValueBase>& paramVals);
        const ::std::list<ParameterValueBase>& getParameterValues();
    protected:
        Ability ability;
    };
```

13322, 13439, 13998, 14003, 14014, 14015

```
    class Stream
        : public InterfaceBase
    {
    public:
        void disconnect();
        bool isConnected();
        const Stream& getConnectedStream();
        const class Service& getService();
    };
```

13439, 13998, 14013, 14014

```
    class OutStream
        : public Stream
    {
    public:
        const ::RoLo::Architecture::Data& getData();
        void activate();
        void deactivate();
        bool isActivated();
    };
```

13439, 14014

```
    class InStream
        : public Stream
    {
    public:
        void setData(const ::RoLo::Architecture::Data& data);
    };
```

13322

```
    class ServiceAbility
        : public Ability
    {
    public:
        Attribute<int> maxOutStreamNum;
        Attribute<double> expectedLatency;
        ::std::list<StreamAbility> inStreamAbilities;
        StreamAbility outStreamAbility;
    };
```

12916, 13322, 13439, 14003, 14014, 14015

```
    class Service
      : public OutStream
    {
    public:
      InStream& connect(const InStream& target, const OutStream* source = 
NULL);
      OutStream& connect(const InStream* source = NULL);
      void adjust(const ::RoLo::Architecture::Data& data);
      const ::std::list<const Service*> getChildren();

    protected:
      ::std::list<InStream> inStreams;
      ::std::list<OutStream> outStreams;
    };

  }
}
```

A sample implementation in C++ language of the proposed platform independent model specified in section 6 is described in the set of C++ header files that accompany this document.

Annex A

13184 **PSM for XML**

Sample C++ Headers

(normative)

(informative)

// $Id: Ability.hpp,v 1.2 $

1. Overview

This annex provides a platform specific model of RoLo Data for XML.

PSM of RoLo data for XML has two variations, *generic model* and *architecture-specific model*. The *generic model* is derived by mapping naively from UML model of RoLo data to XML, and is able to represent any RoLo data for any RoLo architecture. But, it is impossible to restrict structures syntactically for a specification of certain architecture even if the architecture of the data is known.

On the other hand, the *architecture-specific* model is generated for each RoLo specification in a pragmatic way, and is able to restrict its syntax strictly according to the specification. But, the XML schema for the representation should be given for each RoLo data specification.

Hereafter, the target namespace of the given XML schemas is assumed to be “http://www.omg.org/rls/1.0”. Also, the prefix “rls” indicates the same namespace.

2. Generic Model

Specified Structure Type

Specified Structure Type is an abstract type to represent structured data used in RoLo architectures, each of which has correspondence to a specification of its structure.

An instance of this type shall have a spec attribute that indicates an identifier of its specification.

The schema of the *Specified Structure Type* is given as follows:

```
<xsd:complexType name="SpecifiedStructureType" abstract="true">
  <xsd:attribute name="spec" type="ID" use="required"/>
</xsd:complexType>
```

Data

XML schema for Data is given as follows:

```
<xsd:element name="Data" type="rls:DataType"/>
<xsd:complexType name="DataType">
  <xsd:complexContent>
    <xsd:extension base="rls:SpecifiedStructureType">
      <xsd:sequence>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element ref="rls:PositionElement" />
          <xsd:element ref="rls:ErrorElement" />
```

13184

```
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Example

```
<rls:Data spec="#myDataSpec0001">
  <rls:PositionElement spec="#myPosSpec0002">
    <rls:pos>
      <rls:SymbolicIdentity srsName="#myCRS0003">
        <rls:coordinate axisName="type">human</rls:coordinate>
        <rls:coordinate axisName="color">red</rls:coordinate>
        <rls:coordinate axisName="seqNum">0253</rls:coordinate>
      </rls:SymbolicIdentity>
    </rls:pos>
    <rls:err>
      <rls:Reliability>0.6</rls:Reliability>
    </rls:err>
  </rls:PositionElement>
  <rls:PositionElement spec="#myPosSpec0004">
    <rls:pos>
      <gml:Point srsName="#myCRS0005">
        <gml:pos>3.25 2.21</gml:pos>
      </gml:Point>
    </rls:pos>
    <rls:err>
      <rls:UniformGaussian>
        <rls:cov nRow="1" nCol="1">
          2.13
        </rls:cov>
      </rls:UniformGaussian>
    </rls:err>
  </rls:PositionElement>
  <rls:PositionElement spec="#myPosSpec0006">
    <rls:pos>
      <gml:TimeInstant frame="#myCRS0007">
        <gml:TimePosition>2009-01-01T00:40:00+09:00</gml:TimePosition>
      </gml:TimeInstant>
    </rls:pos>
  </rls:PositionElement>
</rls:Data>
```

Element

```
<xsd:complexType name="ElementType" abstract="true">
  <xsd:complexContent>
    <xsd:extension base="rls:SpecifiedStructureType">
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

PositionElement

```
<xsd:element name="PositionElement" type="rls:PositionElementType"/>
<xsd:complexType name="PositionElementType">
```

```
    <xsd:complexContent>
        <xsd:extension base="rls:ElementType">
            <xsd:sequence>
                <xsd:element name="pos"
                                          type="rls:PositionType"/>
                <xsd:element name="err" minOccurs="0" maxOccurs="1"
                                          type="rls:ErrorType"/>
            </xsd:sequence>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
```

Example 1

```
<rls:PositionElement spec="#myPosSpec0002">
    <rls:pos>
        <rls:SymbolPoint srsName="#myCRS0003">
            <rls:coordinate axisName="type">human</rls:coordinate>
            <rls:coordinate axisName="color">red</rls:coordinate>
            <rls:coordinate axisName="seqNum">0253</rls:coordinate>
        </rls:SymbolPoint>
    </rls:pos>
    <rls:err>
        <rls:Reliability>0.6</rls:Reliability>
    </rls:err>
</rls:PositionElement>
```

Example 2

```
<rls:PositionElement spec="#myPosSpec0004">
    <rls:pos>
        <gml:Point srsName="#myCRS0005">
            <gml:pos>3.25 2.21</gml:pos>
        </gml:Point>
    </rls:pos>
    <rls:err>
        <rls:UniformGaussian>
            <rls:cov nRow="1" nCol="1">
                  2.13
            </rls:cov>
        </rls:UniformGaussian>
    </rls:err>
</rls:PositionElement>
```

Example 3

```
<rls:PositionElement spec="#myPosSpec0006">
    <rls:pos>
        <gml:TimeInstant frame="#myCRS0007">
            <gml:TimePosition>2009-01-01T00:40:00+09:00</gml:TimePosition>
        </gml:TimeInstant>
    </rls:pos>
</rls:PositionElement>
```

ErrorElement

XML schema for ErrorElement is given as follows:

```
<xsd:element name="ErrorElement" type="rls:ErrorElementType"/>
<xsd:complexType name="ErrorElementType">
```

13184

```
  <xsd:complexContent>
    <xsd:extension base="rls:ElementType">
      <xsd:sequence>
        <xsd:element name="err"
                     type="rls:ErrorComponentType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Example

```
<rls:ErrorElement spec="#myErrorSpec1234">
  <rls:err>
    <rls:Gaussian>
      <rls:cov nRow="3" nCol="1">
         2.31 -0.32  1.23
        -0.32  1.50  0.01
         1.23  0.01 10.31
      </rls:cov>
    </rls:Gaussian>
  </rls:err>
</rls:ErrorElement>
```

Position

Position is a union of classes SymbolicPosition in the Architecture package, GM_Position in ISO 19107 and TM_Position in ISO 19108. So, its XML expression is a choice of their corresponding XMLs as follows:

```
<xsd:complexType name="PositionType">
  <xsd:choice>
    <xsd:element ref="rls:SymbolicPosition"/>
    <xsd:element ref="gml:Point"/>
    <xsd:element ref="gml:TimeInstant"/>
  </xsd:choice>
</xsd:complexType>
```

SymbolicPosition

```
<xsd:element name="SymbolicPosition"
             type="rls:SymbolicPositionType" />
<xsd:complexType name="SymbolicPositionType">
  <xsd:sequence>
    <xsd:element ref="rls:coords" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="coords"
             type="rls:SymbolicCoordinateType" />
<xsd:complexType name="SymbolicCoordinateType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="axisName" type="xsd:string" use="required" />
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Example

```
<rls:SymbolicIdentity srsName="#myCRS0003">
```

13184

```
  <rls:coordinate axisName="type">human</rls:coordinate>
  <rls:coordinate axisName="color">red</rls:coordinate>
  <rls:coordinate axisName="seqNum">0253</rls:coordinate>
</rls:SymbolicIdentity>
```

Error (Base)

XML schema for Error is given as follows:

```
<xsd:complexType name="ErrorComponentType">
  <xsd:sequence>
    <xsd:element ref="rls:AbstractError"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="AbstractError" 
             type="AbstractErrorType" 
             abstract="true"/>
<xsd:complexType name="AbstractErrorType">
  <xsd:attribute name="errorType" type="ID" use="optional"/>
</xsd:complexType>
```

Error (Variations)

Reliability

```
<!-- Reliability -->
<xsd:element name="Reliability"
             type="rls:ProbabilityType"
             substitutionGroup="rls:AbstractError" />
<xsd:simpleType name="ProbabilityType">
  <xsd:restriction base="float"/>
</xsd:simpleType>
```

Example

```
<rls:Reliability>0.7</rls:Reliability>
```

ErrorDisribution

```
<!-- ErrorDistribution -->
<xsd:element name="ErrorDistribution"
             type="rls:ErrorDistributionType"
             substitutionGroup="rls:AbstractError"
             abstract="true" />
<xsd:complexType name="ErrorDistributionType">
  <xsd:complexContent>
    <xsd:extension base="rls:ErrorType"/>
  </xsd:complexContent>
</xsd:complexType>
```

Gaussian

```
<!-- Gaussian -->
<xsd:element name="Gaussian"
             type="rls:GaussianType"
             substitutionGroup="rls:ErrorDistrubition" />

<xsd:complexType name="GaussianType">
  <xsd:complexContent>
    <xsd:extension base="rls:ErrorDistributionType">
      <xsd:sequence>
        <xsd:element name="cov" 
```

13184

```
                     type="rls:CovarianceMatrixType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="MatrixType">
  <xsd:simpleContent>
    <xsd:extension base="gml:doubleList">
      <xsd:attribute name="nRow" type="integer"/>
      <xsd:attribute name="nCol" type="integer"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="CovarianceMatrixType">
  <xsd:restriction base="rls:MatrixType">
    <xsd:annotation>
      Attributes "nRow" should be equal to "nCol"
    </xsd:annotation>
  </xsd:restriction>
</xsd:complexType>
```

Example

```
<rls:Gaussian>
  <rls:cov nRow="3" nCol="3">
    3.20  0.53  0.02
    0.53  9.21 -3.05
    0.02 -3.05 12.00
  </rls:cov>
</rls:Gaussian>
```

UniformGaussian

```
<!-- Uniform Gaussian -->
<xsd:element name="UniformGaussian"
             type="rls:UniformGaussianType"
             substitutionGroup="rls:Gaussian" />

<xsd:complexType name="UniformGaussianType">
  <xsd:complexContent>
    <xsd:extension base="rls:GaussianType">
      <xsd:annotation>
        Attributes "nRow" and "nCol" should be "1".
      </xsd:annotation>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Example

```
<rls:UniformGaussian>
  <rls:cov nRow="1" nCol="1">
    2.43
  </rls:cov>
</rls:UniformGaussian>
```

13184

MixtureModel

```
<!-- Mixture Model -->
<xsd:element name="AbstractMixtureModel"
             type="rls:AbstractMixtureModelType"
             substitutionGroup="rls:ErrorDistribution"
             abstract="true" />

<xsd:complexType name="AbstractMixtureModelType"
                 abstract="true">
  <xsd:complexContent>
    <xsd:extension base="rls:ErrorDistributionType"/>
  </xsd:complexContent>
</xsd:complexType>
```

LinearMixtureModel

```
<!-- Linear Mixture Model -->
<xsd:element name="LinearMixtureModel"
             type="rls:LinearMixtureModelType"
             substitutionGroup="rls:AbstractMixtureModel"
             abstract="true" />

<xsd:complexType name="LinearMixtureModelType">
  <xsd:complexContent>
    <xsd:extension base="rls:AbstractMixtureModelType">
      <xsd:sequence>
        <xsd:element name="model" type="rls:WeightedModelType"
                     minOccurs=”1” maxOccurs="unbounded" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="WeightedModelType">
  <xsd:sequence>
    <xsd:element name="posElem" type="rls:PitionElementType"/>
    <xsd:element name="weight" type="rls:ProbabilityType"/>
  </xsd:sequence>
</xsd:complexType>
```

ParticleSet

```
<!-- Particle Set -->
<xsd:element name="ParticleSet"
             type="rls:ParticleSetType"
             substitutionGroup="rls:LinearMixtureModel" />

<xsd:complexType name="ParticleSetType">
  <xsd:complexContent>
    <xsd:extension base="rls:LinearMixtureModelType">
      <xsd:annotation>
        Each "model" element shall contain
        without "err" element.
        This is interpreted that the error is
        a Gaussian distribution with
        an all-zero covariance matrix.
      </xsd:annotation>
```

13184

```
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Example

```
<rls:ParticleSet>
  <rls:model>
    <rls:posElem>
      <rls:pos>
        <gml:Point srsName="#myCRS0001">
          <gml:pos>20.34 -2.59</gml:pos>
        </gml:Point>
      </rls:pos>
    </rls:posElem>
    <rls:weight>0.8</rls:weight>
  </rls:model>
  <rls:model>
    <rls:posElem>
      <rls:pos>
        <gml:Point srsName="#myCRS0001">
          <gml:pos>17.25 -3.01</gml:pos>
        </gml:Point>
      </rls:pos>
    </rls:posElem>
    <rls:weight>0.3</rls:weight>
  </rls:model>
  <rls:model>
    <rls:posElem>
      <rls:pos>
        <gml:Point srsName="#myCRS0001">
          <gml:pos>21.99 -1.51</gml:pos>
        </gml:Point>
      </rls:pos>
    </rls:posElem>
    <rls:weight>0.2</rls:weight>
  </rls:model>
</rls:ParticleSet>
```

MixtureOfGaussian

```
<!-- MixtureOfGaussian -->
<xsd:element name="MixtureOfGaussian"
             type="rls:MixtureOfGaussianType"
             substitutionGroup="rls:LinearMixtureModel" />

<xsd:complexType name="MixtureOfGaussianType">
  <xsd:complexContent>
    <xsd:extension base="rls:LinearMixtureModelType">
      <xsd:annotation>
        Each "model" element shall contain
        an error information of Gaussian distribution.
      </xsd:annotation>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

13184 **Example**

```
<rls:MixtureOfGaussian>
  <rls:model>
    <rls:posElem>
      <rls:pos>
        <gml:Point srsName="#myCRS0001">
          <gml:pos>20.34 -2.59</gml:pos>
        </gml:Point>
      </rls:pos>
      <rls:err>
        <rls:Gaussian>
          <rls:cov nRow="2" nCol="2">
             0.92 -0.07
            -0.07  0.30
          </rls:cov>
        </rls:Gaussian>
      </rls:err>
    </rls:posElem>
    <rls:weight>0.6</rls:weight>
  </rls:model>
  <rls:model>
    <rls:posElem>
      <rls:pos>
        <gml:Point srsName="#myCRS0001">
          <gml:pos>19.55 -1.30</gml:pos>
        </gml:Point>
      </rls:pos>
      <rls:err>
        <rls:UniformGaussian>
          <rls:cov nRow="1" nCol="1">
             0.7
          </rls:cov>
        </rls:UniformGaussian>
      </rls:err>
    </rls:posElem>
    <rls:weight>0.6</rls:weight>
  </rls:model>
</rls:MixtureOfGaussian>
```

3. Architecture-specific Model

While the generic model shown above can represent any RoLo data, it is redundant and over generalized so that it is difficult to check validity of the data syntactically according to the corresponding specifications. The architecture-specific model will provides another mapping of a RoLo data to XML that is tightly restricted for the corresponding RoLo architecture specifications.

Identifier and Tag Naming

In order to provide unique name of each component of RoLo data in a systematic way, we suppose that the following restrictions are applied to each related instance of RoLo architectures:

- Each instance of DataSpecification, ElementSpecification, ErrorType, SymbolicIdentityCS, and ::ISO19111:CS_CoordianteSystemAxis shall have an identifier attribute that follow the following syntax: (In the following BNF, we use “<<” and “>>” instead of “<” and “>” to avoid confusion of XML's tags and nonterminal symbols.)

13184

```
<<identifier>> ::= <<namespace>> <<separator>> <<localname>>
 <<namespace>> ::= <<xsd:anyURI>>
 <<separator>> ::= "/" | ":" | "#"
 <<localname>> ::= <<xsd::NCName>>
```

Here, <<xsd:anyURI>> and <<xsd:NCName>> are the restricted character strings that are defined in "W3C XML Schema Definition Language". From a given identifier that follows above syntax, we extract a namespace and a localname for a corresponding instance of Data, Element, Error, and SymbolicPosition and its axis name of coordinates using <<namespace>> and <<localname>>, respectively, part in <<identifier>>.

- ::ISO19111:CS_CoordianteSystemAxis's axisAbbrev attribute shall identical to the <<localname>> part of its identifier attribute.

RoLo Data

Suppose that a DataSpecification has an identifier attribute, whose <<namespace>> and <<localname>> part are ``#myNamespace000'' and ``myRoLoData'', respectively. We also suppose that the specification consists of a list of RoLo element specifications whose qualified names are ``myElement0'', ``myElement1'', ``myElement2'', and so on. Then, the XML schema of corresponding Data instance shall be as following. Here we assume that the target namespace of the following schema is "#myNamespace000" that corresponds to "app" prefix.

```
<xsd:element name="myRoLoData" type="myRoLoDataType"/>
<xsd:complexType name="myRoLoDataType">
  <xsd:sequence>
    <xsd:element ref="app:myElement0"/>
    <xsd:element ref="app:myElement1"/>
    <xsd:element ref="app:myElement2"/>
    ...
  </xsd:sequence>
</xsd:complexType>
```

Syntax of the contents of each Element is declared according to specifications of each ElementSpecification as describe below.

Example

```
<app:SensedObjectInfo xmlns:app="#myApplication000">
  <app:id>
    <app:pos>
      <app:SensedObjectId srsName="#myCRS0003">
        <app:type>human</app:type>
        <app:color>red</app:color>
        <app:seqNum>0253</app:seqNum>
      </app:SensedObjectId>
    </app:pos>
    <app:err>
      <rls:Reliability>0.6</rls:Reliability>
    </app:err>
  </app:id>
  <app:location>
    <app:pos>
      <gml:Point srsName="#myCRS0005">
        <gml:pos>3.25 2.21</gml:pos>
      </gml:Point>
    </app:pos>
    <app:err>
```

13184

```
        <rls:UniformGaussian>
          <rls:cov nRow="1" nCol="1">
            2.13
          </rls:cov>
        </rls:UniformGaussian>
      </app:err>
    </app:location>
    <app:time>
      <app:pos>
        <gml:TimeInstant frame="#myCRS0007">
          <gml:TimePosition>2009-01-01T00:40:00+09:00</gml:TimePosition>
        </gml:TimeInstant>
      </app:pos>
    </app:time>
</app:SensedObjectInfo>
```

PositionElement

Suppose that a PositionElementSpecification has an identifier attribute, whose namespace and localname part are ``#myNamespace000'' and ``myPosElement'', respectively. Then, the XML schema of a corresponding PositionElement shall be:

```
<xsd:element name="myPosElement" type="myPosElementType"/>
<xsd:complexType name="myPosElementType">
  <xsd:sequence>
    <xsd:element name="pos" type="<<myPosType>>"/>
    {<xsd:element name="err" type="rls:ErrorComponentType"/>}?
  </xsd:sequence>
</xsd:complexType>
```

, where <<myPosType>> is:

- an application specific SymbolicPosition type describe below if the CS_CoordinateSystem of PositionElementSpecification refers an identityCS,
- gml:TimeInstantPropertyType, if the cs is a temporal coordiante system,
- or, gml:PointPropertyType otherwise.

The “err” element part can be omitted according to the specification.

Example 1

```
<app:id xmlns:app="#myApplication000">
  <app:pos>
    <app:SensedObjectID srsName="#myCRS0003">
      <app:type>human</app:type>
      <app:color>red</app:color>
      <app:seqNum>0253</app:seqNum>
    </app:SensedObjectID>
  </app:pos>
  <app:err>
    <rls:Reliability>0.6</rls:Reliability>
  </app:err>
</app:id>
```

Example 2

```
<app:location xmlns:app="#myApplication000">
  <app:pos>
    <gml:Point srsName="#myCRS0005">
      <gml:pos>3.25 2.21</gml:pos>
```

13184

```
    </gml:Point>
  </app:pos>
  <app:err>
    <rls:UniformGaussian>
      <rls:cov nRow="1" nCol="1">
         2.13
      </rls:cov>
    </rls:UniformGaussian>
  </app:err>
</app:location>
```

Example 3

```
<app:time xmlns:app="#myApplication000">
  <app:pos>
    <gml:TimeInstant frame="#myCRS0007">
      <gml:TimePosition>2009-01-01T00:40:00+09:00</gml:TimePosition>
    </gml:TimeInstant>
  </app:pos>
</app:time>
```

ErrorElement

Suppose that a RoLo error element specification has an identifier attribute, whose namespace and localname parts are ``#myNamespace000'' and ``myErrElement'', respectively. Then, the XML expression of a corresponding *Error Element* shall be:

```
<xsd:element name="myErrElement" type="myErrElementType"/>
<xsd:complexType name="myErrElementType>
  <xsd:sequence>
    <xsd:element name="err" type="rls:ErrorComponentType"/>
  </xsd:sequence>
</xsd:complexType>
```

Example

```
<app:myError xmlns:app="#myApplication000">
  <app:err>
    <rls:Gaussian>
      <rls:cov nRow="3" nCol="1">
         2.31 -0.32  1.23
        -0.32  1.50  0.01
         1.23  0.01 10.31
      </rls:cov>
    </rls:Gaussian>
  </app:err>
</app:myError>
```

Symbolic Position

Suppose that an IdentityCS has an identifier attribute, whose namespace and localname parts are ``#myNamespace000'' and ``myIdCS'', respectively. We also suppose that the usesAxis attribute of the IdentityCS consists of a list of CoordinateSystemAxis [ISO19111] whose axisAbbrev (that is identical to the localname part in the identifier attribute of the axis) are ``myAxis0'', ``myAxis1'', ``myAxis2'', and so on. Then, the XML schema of a corresponding SymbolicPosition shall be as follows:

```
<xsd:element name="myIdCS" type="myIdCSType"/>
<xsd:complexType name="myIdCSType>
```

13184

```
  <xsd:sequence>
    <xsd:element name=”myAxis0” type=”xsd:string”/>
    <xsd:element name=”myAxis1” type=”xsd:string”/>
    <xsd:element name=”myAxis2” type=”xsd:string”/>
    ...
  </xsd:sequence>
</xsd:complexType>
```

Example

```
<app:id xmlns:app="#myApplication000" srsName="#myCRS0003">
  <app:type>human</app:type>
  <app:color>red</app:color>
  <app:seqNum>0253</app:seqNum>
</app:id>
```

13115

```
#pragma once
#include <string>
#include <set>
#include <ISO19103/Primitive.hpp>
#include <ISO19111/IO_IdentifiedObject.hpp>
#include <ISO19115.hpp>
#include <RLS/RoLoArchitecture.hpp>
#include <RLS/Error.hpp>

namespace RLS
{
  class RoLoAttributeValue
  {
  };

  class RoLoAttributeSingleValue
    : public RoLoAttributeValue
  {
  public:
    union {
      ::ISO19103::Measure measureValue;
      std::string stringValue;
      int integerValue;
      double realValue;
      ::ISO19103::Probability probabilityValue;
      RoLoData rldValue;
      RoLoElement rleValue;
      RoLoError errorValue;
      RoLoPosition positionValue;
    } value;
  };

  class RoLoAttributeValueSet
    : public RoLoAttributeValue
  {
  public:
    std::set<RoLoAttributeSingleValue> value;
  };
  
  // actually union
  class RoLoAttributeType {
    std::string name;
    ::ISO19115::RS_Identifier reference;
  };

  class RoLoAttributeDefinition
    : public ISO19111::IO_IdentifiedObject
  {
  public:
```

Note: pages omitted for removed C++ PSM

13115

```
    Returncode_t getInStream(int index, InStream*& inStream);
    Returncode_t adjust(const RoLoData* data);
    Returncode_t getChild(int index, RoLoService*& service);
    Returncode_t getAbility(ServiceAbility*& ability);
    Returncode_t setParameter(const ServiceParameterSet* parameter);
    Returncode_t getParameter(ServiceParameterSet*& parameter);
  };

}
```

Annex B

Naming of RoLo Architecture Components for Filter Condition

Sample XMI

12918

(normative)


```
<?xml version="1.0" encoding="UTF-8" ?>
<xmi:XMI xmi:version="2.1" xmlns:uml="http://schema.omg.org/spec/UML/2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmlns:thecustomprofile="http://www.sparxsystems.com/profiles/thecustomprofile/1.0">
<xmi:Documentation exporter="Enterprise Architect" exporterVersion="6.5" />
<uml:Model xmi:type="uml:Model" name="EA_Model" visibility="vis_public">
<packagedElement xmi:type="uml:Profile" xmi:id="EAPK_A0351711_2971_4024_9ECB_A15C58CF83EE" name="RLS" visibility="vis_public">
<packagedElement xmi:type="uml:Package" xmi:id="EAPK_304A773B_0A3F_4bd0_855C_7E6F2CC5BEC7" name="RLS" visibility="vis_public">
<ownedComment xmi:type="uml:Comment" xmi:id="EAID_C5D002E6_4ACB_45cf_A9EB_D803FACA1D0A" body="the mean value is the combined RoLo Position">
<annotatedElement xmi:idref="EAID_A1E70738_C83F_44d4_8C30_CD85F7C2464B" />
<annotatedElement xmi:idref="EAID_D247A86B_94D7_4bda_B144_381D8AAF6234" />
</ownedComment>
<ownedComment xmi:type="uml:Comment" xmi:id="EAID_99376F43_E4A7_406b_AFAC_AE54C14B9BBE" body="RoLo Architecture.elements.size == RoLo Data.elements.size">
<annotatedElement xmi:idref="EAID_1D9788D0_AE52_4031_819B_BCDBE41C5719" />
<annotatedElement xmi:idref="EAID_9F6F2265_1219_4f8a_8AFB_D14E5C4F6348" />
</ownedComment>
<packagedElement xmi:type="uml:Class" xmi:id="EAID_60459B3D_9CAC_4a38_BE36_941A62DCBC40" name="Common Data Format" visibility="vis_public" isAbstract="true">
<generalization xmi:type="uml:Generalization" xmi:id="EAID_CCC91D54_C36F_45c7_B668_649C213D3E9C" general="EAID_DBA6B7D9_FF39_4695_97D7_4D2ADA30FDDD" />
</packagedElement>
```


Note: pages omitted for removed XMI

12918 <element geometry="SX=0;SY=0;EX=0;EY=0;EDGE=1;$LLB=;LLT=;LMT=;LMB=;LRT=;LRB=;IRHS=;ILHS=;Path=;" subject="EAID_0835C9C2_17EF_4ce9_8AA5_22E0EDE32CAD" style="Mode=3;EOID=E7404CFA;SOID=DAB6CD41;Hidden=0;" />
</elements>
</diagram>
</diagrams>
</xmi:Extension>
</xmi:XMI>

(informative)

13184 This annex provides a naming rule of RoLo architecture components for use with filter condition. In order to utilize the filter condition, we need a way to specify components in a RoLo data to test each condition. For this purpose, we suppose that each RoLo data can be expressed by XML-PSM (see annex A) and use XPath to indicate each part of RoLo data as

13184 same as the original filter encoding in ISO 19143.

Example 1

This example is an XML encoding of a filter condition that requires only localization data in a certain area.

```
<fes:Filter
  xmlns:fes="http://www.opengis.net/fes/2.0"
   xmlns:gml="http://www.opengis.net/gml/3.1"
  xmlns:myapp="http://my.localhost.localnet/myapp"
  xmlns:rls="http://www.omg.org/rls/1.0">
  <fes:Intersects>
    <fes:PropertyName>myapp:location/rls:pos</fes:PropertyName>
    <gml:Envelope
      srsName="http://my.localhost.localnet/myapp/crs000">
          <gml:lowerCorner>10.25 15.33</gml:lowerCorner>
          <gml:upperCorner>17.73 25.03</gml:upperCorner>
    </gml:Envelope>
  </fes:Intersects>
</fes:Filter>
```

Example 2

This example is an XML encoding of a filter condition that requires only localization data of a certain ID.

```
<fes:Filter
  xmlns:fes="http://www.opengis.net/fes/2.0"
  xmlns:gml="http://www.opengis.net/gml/3.1"
  xmlns:myapp="http://my.localhost.localnet/myapp"
  xmlns:rls="http://www.omg.org/rls/1.0">
  <fes:PropertyIsEqualTo>
    <fes:PropertyName>myapp:id/rls:pos</fes:PropertyName>
    <fes:Literal>myID:3429:abcd</fes:Literal>
  </fes:PropertyIsEqualTo>
</fes:Filter>
```

Example 3

This example is an XML encoding of a filter condition that requires only localization data in a certain area and a certain time period.

```
  <fes:Filter
      xmlns:fes="http://www.opengis.net/fes/2.0"
               xmlns:gml="http://www.opengis.net/gml/3.1"
      xmlns:myapp="http://my.localhost.localnet/myapp"
      xmlns:rls="http://www.omg.org/rls/1.0">
    <fes:And>
```

13184

```
          <fes:Intersects>
            <fes:PropertyName>myapp:location/rls:pos</fes:PropertyName>
            <gml:Polygon
                srsName="http://my.localhost.localnet/myapp/crs000">
              <gml:exterior>
                <gml:LinearRing>
                  <gml:posList dimension="2">
                    23.02 34.21
                    11.56 23.14
                    90.43 23.19
                    33.23 29.00
                    23.02 34.21
                  </gml:posList>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </fes:Intersects>
          <fes:PropertyIsBetween>
            <fes:PropertyName>myapp:time/rls:pos</fes:PropertyName>
            <fes:LowerBoundary>
              <fes:Literal>2008-12-08T09:00:00.000-08:00</fes:Literal>
            </fes:LowerBoundary>
            <fes:UpperBoundary>
              <fes:Literal>2008-12-10T17:30:00.000-08:00</fes:Literal>
            </fes:UpperBoundary>
          </fes:PropertyIsBetween>
        </fes:And>
    </fes:Filter>
```