

Version	Date	Remark
1.0	01.06.2020	First version of OSP interface specification

Contents

1 Introduction 4
1.1 About the Specification 4
1.2 What You Stand to Gain 4
1.3 License 5
1.4 Acknowledgements 5
1.5 OSP-IS Maintenance 6
1.6 Limitations 6
2 Definitions 7
2.1 General Definitions 7
2.2 Special Definitions 7
3 Connections Between Simulation Models 9
3.1 OspModelDescription 9
3.1.1 Example OspModelDescription.xml 31
4 Ontology 32
4.1 Classes 32
4.2 Individuals 32
4.3 Object Properties 35
4.4 Populating the OSP-IS Ontology 36
4.5 Model Validation Using OSP-IS Ontology 38
4.5.1 Model Validations 38
4.6 Configuration Validation Using OSP-IS Ontology 39
4.6.1 Configuration Validations 40
4.7 OSP-Validator: A Reference Implementation 40
5 Principles for Interfacing with Control Systems 41
5.1 The Information Layer: Sensor and Actuator FMUs 41
5.2 The Communication Layer: Network FMUs 42
5.3 Example System: DP Controller 42
5.4 An Interface for DP Controllers 44
5.5 Status, Challenges, and Outlook 61
5.5.1 A Note on Hardware-in-the-Loop Testing With Hardwired Signals 62
6 References 62
Appendix A TUTORIALS 63
A.1 How to Magically Solve Interfacing Issues 63

A.1.1 The Agony of Connecting Models 63
A.1.2 Saving Your Time and Your Nerves with OSP-IS 65
A.1.3 Your Effortless Transition to OSP-IS 67
A.1.4 Defining the Model Interfaces 67
A.1.5 Defining the Model Connections 69
A.2 How OSP-IS Makes Digital Collaboration Easier 70
A.3 What's Next? 71
A.4 A Message to the Simulation Tool Developer 71
Appendix A Form for New Variable Groups 74
A.1 Examples 75

1 INTRODUCTION

1.1 ABOUT THE SPECIFICATION

The OSP interface specification (OSP-IS) enables faster construction of co-simulation simulators by simplifying the connection process, and validation of semantically correct simulations. OSP-IS is an addition to the FMI for co-simulation standard [1] and provides a method for adding semantic meaning to model interface variables. The OSP interface specification consists of four main parts:

1. The present document.
2. An ontology which is described in Chapter 4 of this document and is available at https://data.dnvgl.com/osp/ontology/core
3. An xml schema described in Chapter 3 of this document and downloadable at https://opensimulationplatform.com/xsd/OspModelDescription-1.0.0.xsd
4. The osp-validator SW tool which is described in section 4.7 of this document and can be downloaded at https://github.com/open-simulation-platform/osp-validator/releases

The content of OSP-IS can be divided into two main categories: model-to-model (simulator-to-simulator) connections and simulator-to-control-system (or simulator-to-hardware) connections. Both categories are elaborated in this document. The basic idea is to group and classify the FMI variables in the connections into variable groups with high interpretability by adding semantic meaning to them. The method is based on an ontology (explained in Chapter 4) and a new xml schema (introduced in Chapter 3).

Connections between simulation models represents the physical interactions between them. Such interactions typically correspond to the energetic coupling of (sub)systems, for example in the mechanical or the electric domain. As we will see in Appendix A, even such simple connections can easily become difficult to handle due to various naming conventions, different units of measure, and developer habits. To support the wide range of complex and multi-domain connections found in typical maritime systems, OSP-IS greatly simplifies the interfaces and helps to ensure that they are set up correctly.

Connections between simulators and control systems and other equipment represent a fundamentally different type of coupling based on the exchange of information and a myriad of different protocols and formats. Add to this the issues of a lacking naming conventions to create a quite complex and challenging problem. To solve it, OSP-IS proposes a similar approach as for connections between simulator models by classifying FMI variables and adding semantic meaning matching the different protocols in use. An introduction to the topic and the proposed method is given in Chapter 5, along with a realistic example case involving a Dynamic Positioning (DP) controller.

1.2 WHAT YOU STAND TO GAIN

If you are already familiar with the OSP-IS method and are looking for **definitions**, Chapter 3 contains the connections between simulator models and Chapter 5 treats simulator-to-control-system interfaces.

If you would like to get a more hands-on feel for how the standard can help you solve your interfacing issues, you will find an **implementation guide** in Appendix A.

If you are interested in **developing validators**, check out our ontology in Chapter 4.

We also need **your help**. The OSP-IS is far from complete and we depend on your input and feedback to make connecting models and control systems as easy and hassle-free as possible for everyone!

- If you are missing connection types and would like to help improve the specification, go to Appendix B. There you will find a form for **suggesting new variable groups**. Simply fill it out and send it to us by e-mail using contribute-ospis@opensimulationplatform.com.
- Please send any other ideas or contributions to contribute-ospis@opensimulationplatform.com.

Let's make digital collaboration easier for everyone!

1.3 LICENSE

This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.

You are free to:

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

To view a copy of this license including all relevant information, visit http://creativecommons.org/licenses/by-sa/4.0/.

NOTE:

libcosim, osp-cli, osp-validator, OSPModelDescription.xsd and the OSP ontology are licensed under different licenses.

1.4 ACKNOWLEDGEMENTS

This specification has been developed as part of the OSP JIP project and the DTYard project (number 295918) funded by the Research Council of Norway.

More information about these projects can be found at www.open-simulation-platform.com.

1.5 OSP-IS MAINTENANCE

The specification will be maintained by the **OSP committee** and will follow the semantic versioning principle [2].

1.6 LIMITATIONS

The requirements within this specification can be applied to simulation models following the Functional Mock-up Interface (FMI) standard [1].

This specification does not pose requirements on model internals, nor does it attempt to serve as a manual for modelling and simulation. It is also not an alternative to the FMI standard; it is an *optional addition* that will simplify the process of connecting models.

2 DEFINITIONS

This chapter contains definitions of central terms used throughout the specification.

2.1 GENERAL DEFINITIONS

Term	Definition	Comments
model	A mathematical or logical representation of a system of entities, phenomena, or processes.	A model is a simplified abstract view of the complex reality.
sub-model	A model which is part of a larger model.	
computer model	A model which is implemented in the form of computer code.	We may simply use "model" when there is no ambiguity.
simulation	The computation of the behaviour of a model under specified conditions.	
computer simulation	The computation of the behaviour of a computer model under specified conditions, carried out by a computer.	This is what we usually mean by "simulation", and we will use the shorter form when there is no ambiguity.
simulated time	The time that passes in the simulation.	We may simply use "time" when there is no ambiguity.
modelling software, modelling program, modelling tool	Software for the purpose of creating computer models.	Most modelling and simulation software contain both classes of functionality, and the terms are therefore sometimes used interchangeably. In this document, we will use them precisely.
simulation software, simulation program, simulation tool	Software for the purpose of carrying out computer simulations.	
co-simulation	A simulation technique in which the computations associated with different sub-models are performed mostly independently from each other, the only interaction being the exchange of output values at certain discrete points in time (communication points).	

2.2 SPECIAL DEFINITIONS

The following terms are here defined specifically in the context of *co-simulation*. They may be defined differently when used generally or in other contexts (e.g. other simulation methods).

Term	Definition	Comments
sub-simulator	A simulation program that performs the computations associated with a sub-model in a co-simulation.	
communication point	A point in simulated time at which data is exchanged between sub-simulators.	

co-simulation software	Software for the purpose of carrying out co-simulations.	
connection	An equality relationship between two model variables that is enforced by the co-simulation software at communication points.	

3 CONNECTIONS BETWEEN SIMULATION MODELS

The OSP interface specification acts as an extension of the FMI standard by providing mechanisms for model developers to add more semantics to their models. By employing the specification, the model variables can be understood in an unambiguous way, both by a computer and a human.

This simplifies interoperability and co-operation between model providers and simulation configurators from different domains. The main feature of OSP-IS is that it provides the possibility of grouping FMI variables into *variable groups* of defined types. Each type is associated with content restrictions and can only be connected to variable groups of the same type. OSP-IS includes variable group types inspired by bond graph theory [3], which is useful and common when connecting simulation models that capture some physical phenomena.

As demonstrated in Appendix A, connecting FMI variables can be very challenging and time consuming because there are no conventions to describe what each variable actually is. The only information at the model developer's disposal is the variable name and description in modelDescription.xml, which is fragile in terms of an unambiguous definition of what a variable represents. This problem becomes even greater when models are exchanged across organizational borders, and different modelling domains - each with their own naming and modelling conventions. It is also challenging with FMI's modelDescription.xml alone, to effectively communicate that a set of variables might logically belong together in a connection (i.e. a voltage/current pair).

OSP-IS attempts to address both issues with a OspModelDescription.xml file that allows a model developer to add more semantic meaning to variables, and group them together in a way that makes logical sense in terms of connections. The mechanism that enables this is the *variable group*. OSP-IS defines a range of variable group types that the model developer has at their disposal, like Force, Voltage, AngularVelocity, etc. This clarifies communication, simplifies connections, and enables improved validation of simulation configurations.

The following section describes the OspModelDescription xml schema and all the variable groups useful in model-to-model connections.

In Section 5.3, we present the part of the xml schema which defines variable groups useful in a case where simulation models interface towards a real DP control system. We define variable groups closely representing the NMEA protocol, making it simple to connect hardware to the simulation. This case is intended as a demonstration to show how OSP-IS can expand with more variable group definitions.

OSP-IS seeks to be community-driven and we strongly encourage everyone to contribute with suggestions for variable groups that are **useful for you**. If you find yourself unable to fully describe your model with the available variable groups, fill out the form in Appendix B and send it to contribute-ospis@opensimulationplatform.com.

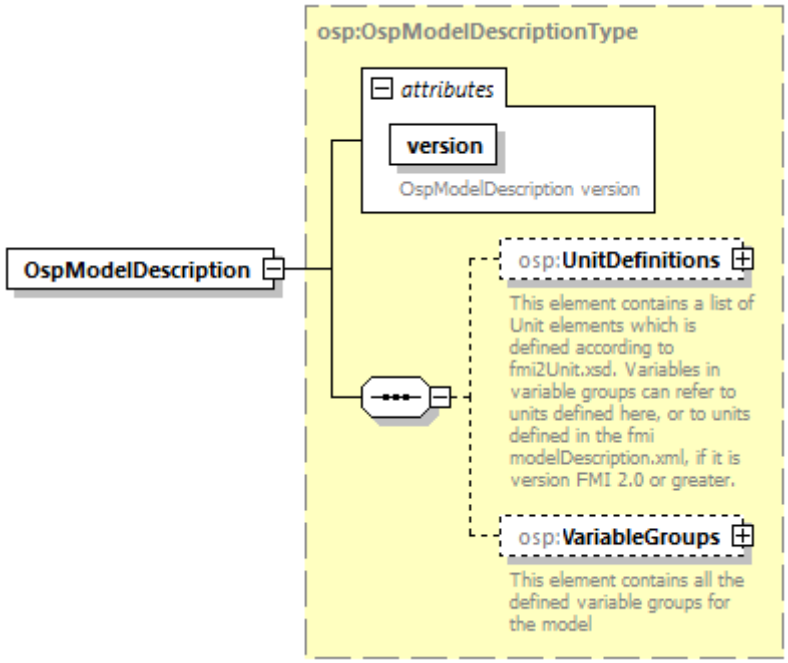
3.1 OSPMODELDESCRIPTION

This section acts as a reference for looking up element and attribute definitions for the part of OspModelDescription.xsd with variable group types intended for model-to-model connections. In addition to defining variable groups, you can also define variable units in OspModelDescription.xml. There, we are using FMI 2.0 standard's fmi2Units.xsd and extend it with 'percent' as a base unit.

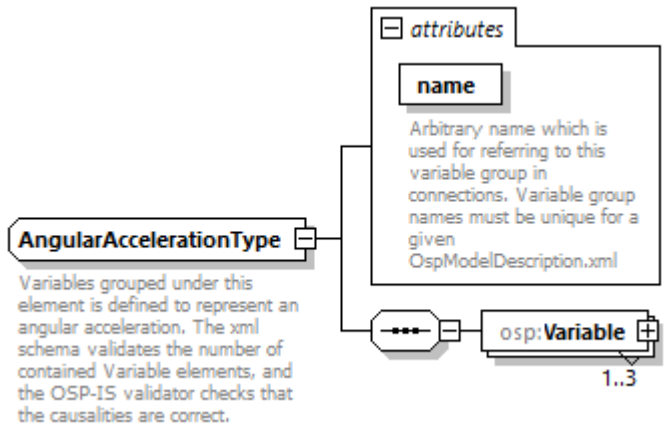
Section 5.3 shows xml schema documentation for the variable groups used for connections between simulation models and control systems in the DP case.

The full documentation for OspModelDescription.xsd can also be found at https://opensimulationplatform.com/xsd/OspModelDescription-1.0.0-doc/OspModelDescription-1.0.0.html.

element **OspModelDescription**

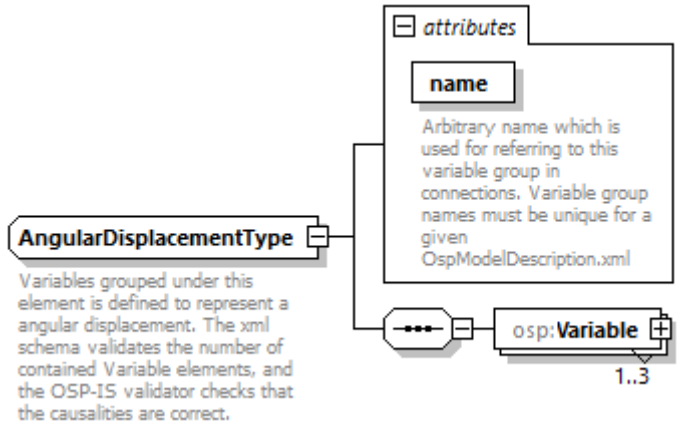
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
type	**osp:OspModelDescriptionType**
properties	content complex
children	**osp:UnitDefinitions osp:VariableGroups**
attributes	Name: version; Type: **xs:string**; Use: required; Default: ; Fixed: 1.0; Annotation: documentation OspModelDescription version

complexType **AngularAccelerationType**

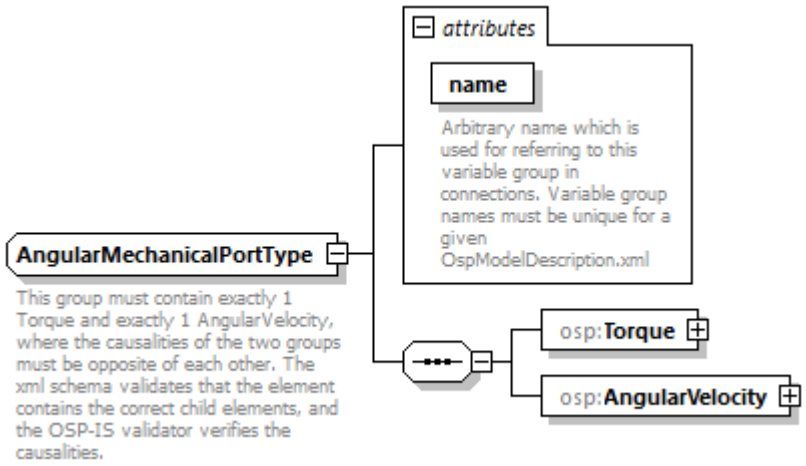
diagram	

namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **GenericType/AngularAcceleration VariableGroupsType/AngularAcceleration**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: Variables grouped under this element is defined to represent an angular acceleration. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

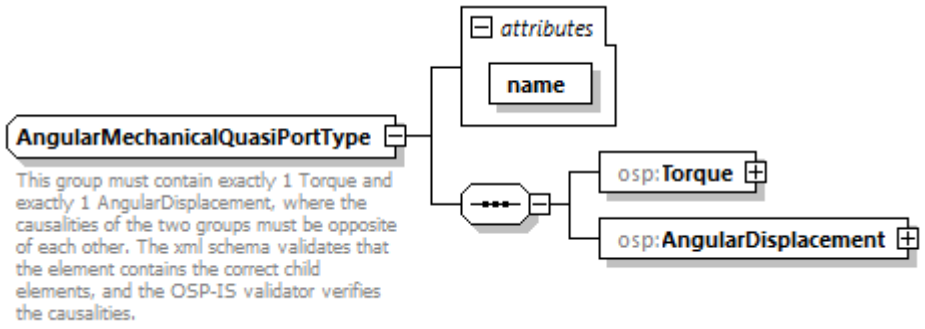
complexType **AngularDisplacementType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **AngularMechanicalQuasiPortType/AngularDisplacement NmeaSxnType/AngularDisplacement GenericType/AngularDisplacement VariableGroupsType/AngularDisplacement** complexTypes **AzimuthAngleType BladePitchType**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: Variables grouped under this element is defined to represent a angular displacement. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

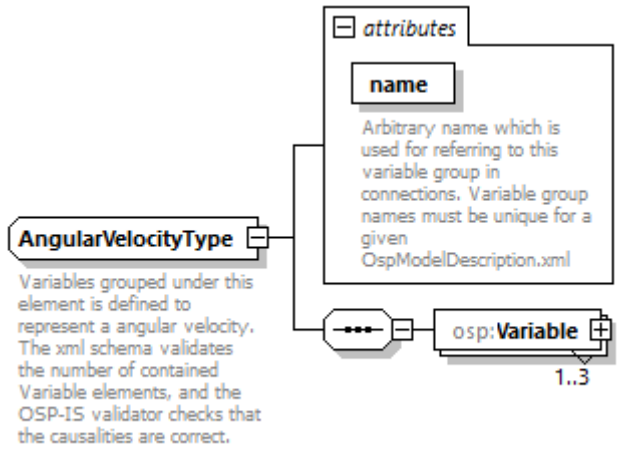
complexType **AngularMechanicalPortType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Torque osp:AngularVelocity**
used by	elements **GenericType/AngularMechanicalPort VariableGroupsType/AngularMechanicalPort**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group must contain exactly 1 Torque and exactly 1 AngularVelocity, where the causalities of the two groups must be opposite of each other. The xml schema validates that the element contains the correct child elements, and the OSP-IS validator verifies the causalities.

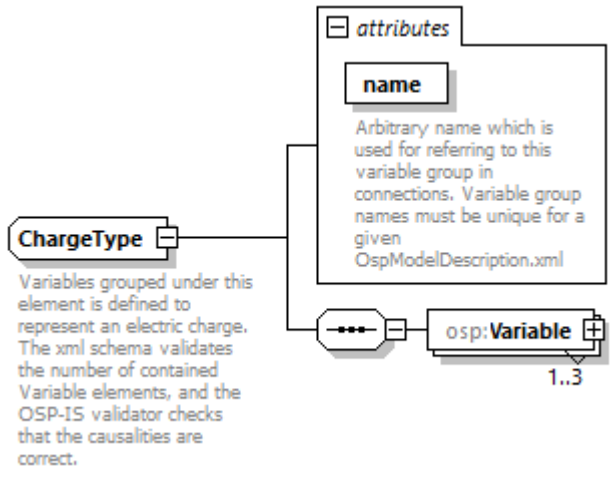
complexType **AngularMechanicalQuasiPortType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Torque osp:AngularDisplacement**
used by	elements **GenericType/AngularMechanicalQuasiPort VariableGroupsType/AngularMechanicalQuasiPort**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation:
annotation	documentation: This group must contain exactly 1 Torque and exactly 1 AngularDisplacement, where the causalities of the two groups must be opposite of each other. The xml schema validates that the element contains the correct child elements, and the OSP-IS validator verifies the causalities.

complexType **AngularVelocityType**

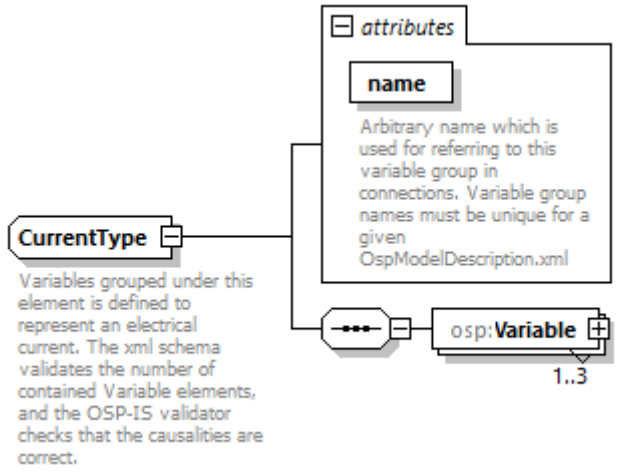
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **AngularMechanicalPortType/AngularVelocity GenericType/AngularVelocity VariableGroupsType/AngularVelocity** complexType **ShaftSpeedType**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation Variables grouped under this element is defined to represent a angular velocity. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

complexType **ChargeType**

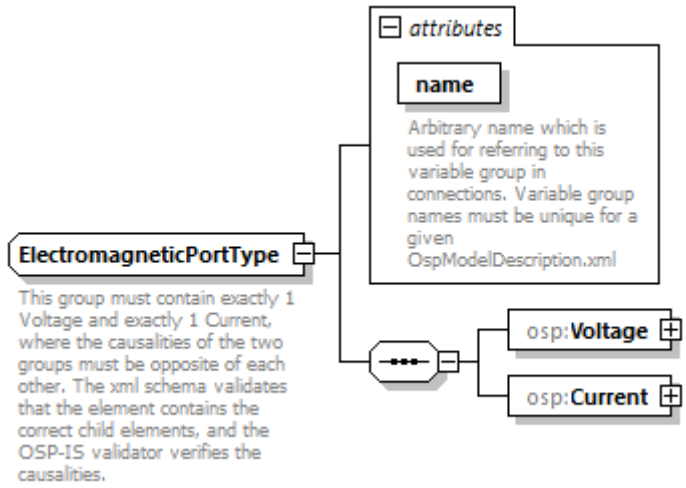
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **ElectromagneticQuasiPortType/Charge GenericType/Charge VariableGroupsType/Charge**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation documentation: Arbitrary name which is used for referring to this

	variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation Variables grouped under this element is defined to represent an electric charge. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

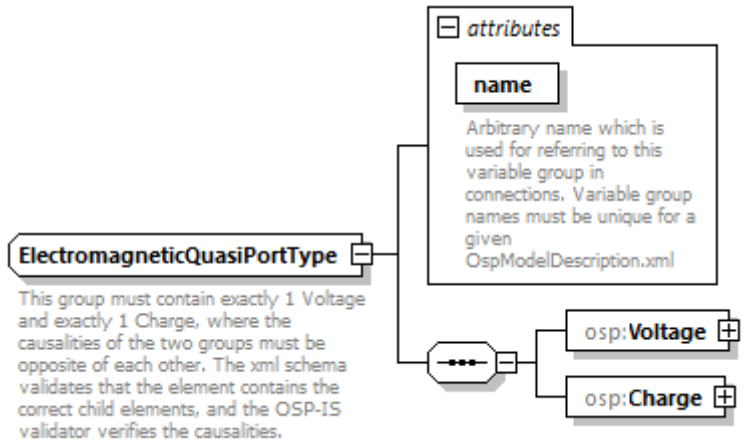
complexType **CurrentType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **ElectromagneticPortType/Current GenericType/Current VariableGroupsType/Current**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation Variables grouped under this element is defined to represent an electrical current. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

complexType **ElectromagneticPortType**

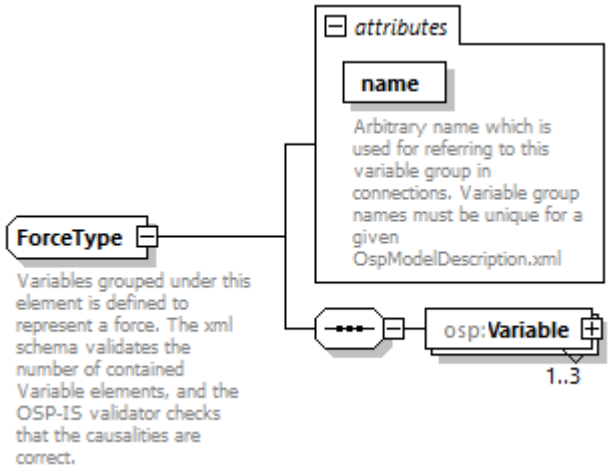
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Voltage osp:Current**
used by	elements **GenericType/ElectromagneticPort VariableGroupsType/ElectromagneticPort**
attributes	Name: **name** — Type: **xs:string** — Use: required — Default: — Fixed: — Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group must contain exactly 1 Voltage and exactly 1 Current, where the causalities of the two groups must be opposite of each other. The xml schema validates that the element contains the correct child elements, and the OSP-IS validator verifies the causalities.

complexType **ElectromagneticQuasiPortType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Voltage osp:Charge**
used by	elements **GenericType/ElectromagneticQuasiPort VariableGroupsType/ElectromagneticQuasiPort**
attributes	Name: **name** — Type: **xs:string** — Use: required — Default: — Fixed: — Annotation: documentation: Arbitrary name which is used for referring to this

		variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This group must contain exactly 1 Voltage and exactly 1 Charge, where the causalities of the two groups must be opposite of each other. The xml schema validates that the element contains the correct child elements, and the OSP-IS validator verifies the causalities.	

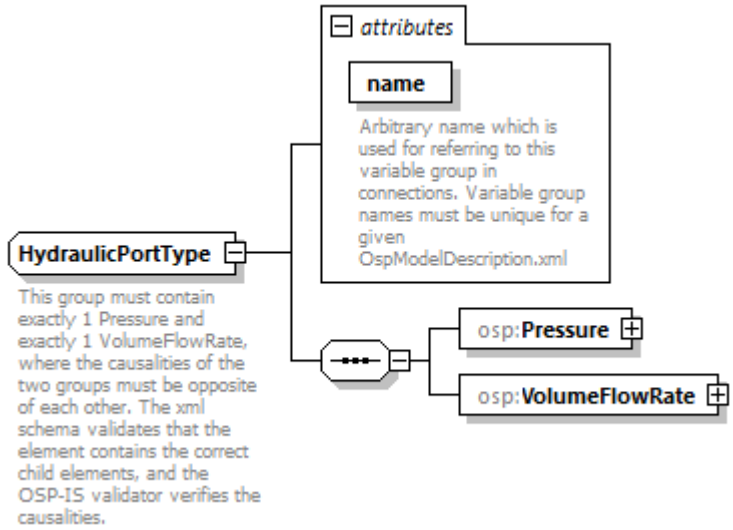
complexType **ForceType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **LinearMechanicalPortType/Force LinearMechanicalQuasiPortType/Force FixedThrusterFeedbackType/Force AzimuthThrusterFeedbackType/Force GenericType/Force VariableGroupsType/Force**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation Variables grouped under this element is defined to represent a force. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

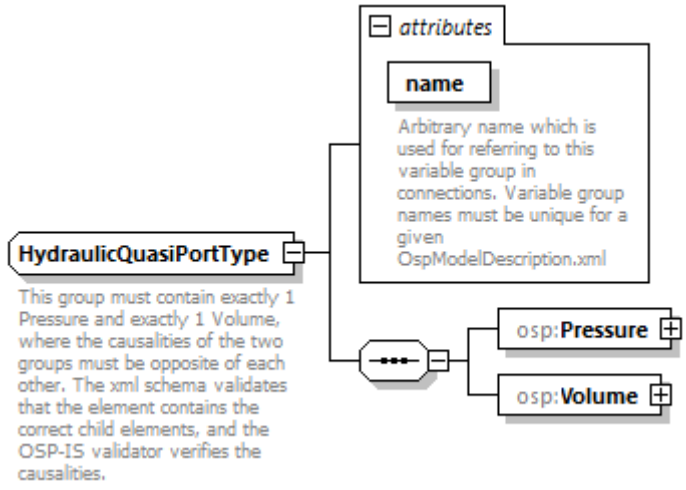
complexType **GenericType**

namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**all defined variable group types including GenericType**
used by	elements **GenericType/Generic VariableGroupsType/Generic**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This variable group has few restrictions on content. The intention with this group is to enable users to create their own custom groups of variables, which does not have a defined type in OSP-IS.

complexType **HydraulicPortType**

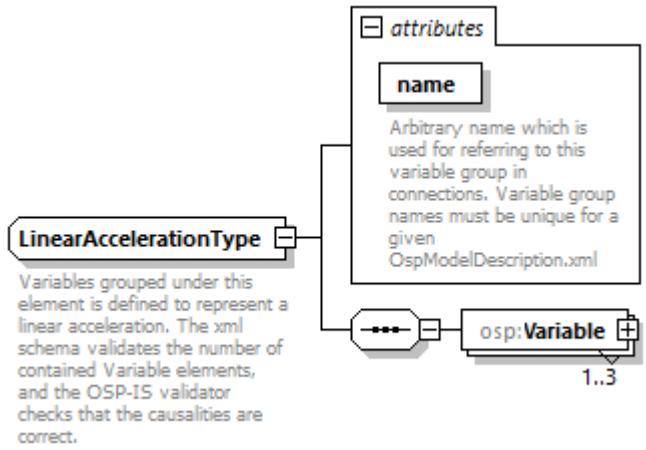
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Pressure osp:VolumeFlowRate**
used by	elements **GenericType/HydraulicPort VariableGroupsType/HydraulicPort**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group must contain exactly 1 Pressure and exactly 1 VolumeFlowRate, where the causalities of the two groups must be opposite of each other. The xml schema validates that the element contains the correct child elements, and the OSP-IS validator verifies the causalities.

complexType **HydraulicQuasiPortType**

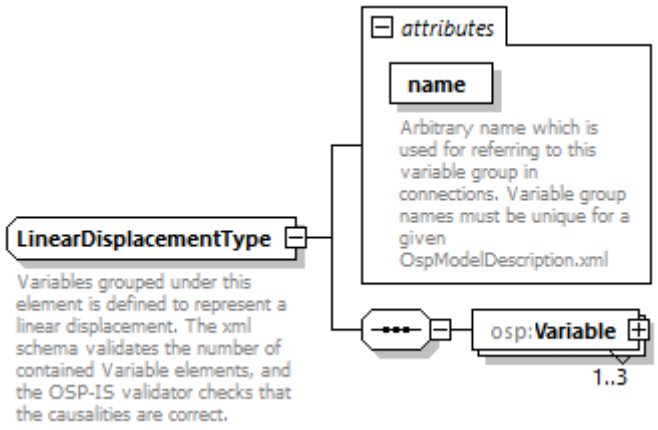
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0

children	**osp:Pressure osp:Volume**
used by	elements **GenericType/HydraulicQuasiPort VariableGroupsType/HydraulicQuasiPort**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group must contain exactly 1 Pressure and exactly 1 Volume, where the causalities of the two groups must be opposite of each other. The xml schema validates that the element contains the correct child elements, and the OSP-IS validator verifies the causalities.

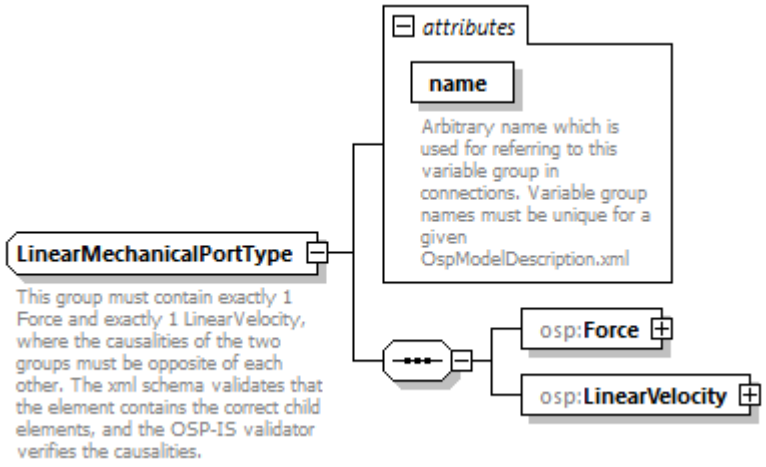
complexType **LinearAccelerationType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **GenericType/LinearAcceleration VariableGroupsType/LinearAcceleration**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: Variables grouped under this element is defined to represent a linear acceleration. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

complexType **LinearDisplacementType**

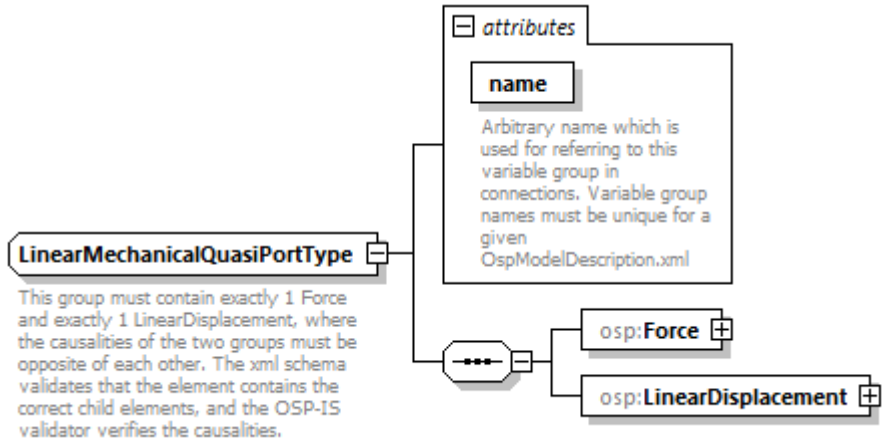
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **LinearMechanicalQuasiPortType/LinearDisplacement GenericType/LinearDisplacement VariableGroupsType/LinearDisplacement**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: Variables grouped under this element is defined to represent a linear displacement. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

complexType **LinearMechanicalPortType**

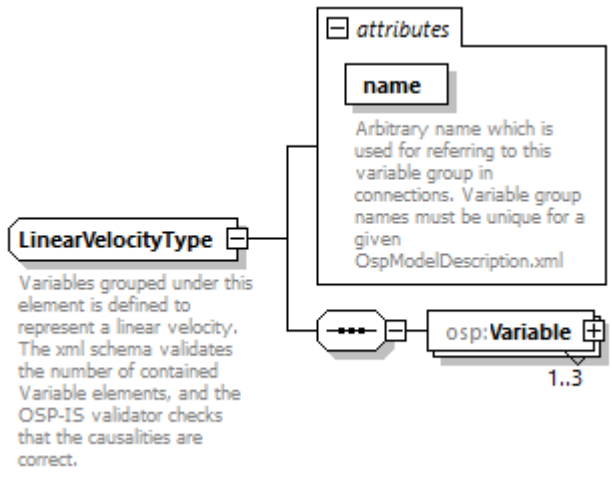
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Force osp:LinearVelocity**
used by	elements **GenericType/LinearMechanicalPort VariableGroupsType/LinearMechanicalPort**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable

	group names must be unique for a given OspModelDescription.xml
annotation	documentation This group must contain exactly 1 Force and exactly 1 LinearVelocity, where the causalities of the two groups must be opposite of each other. The xml schema validates that the element contains the correct child elements, and the OSP-IS validator verifies the causalities.

complexType **LinearMechanicalQuasiPortType**

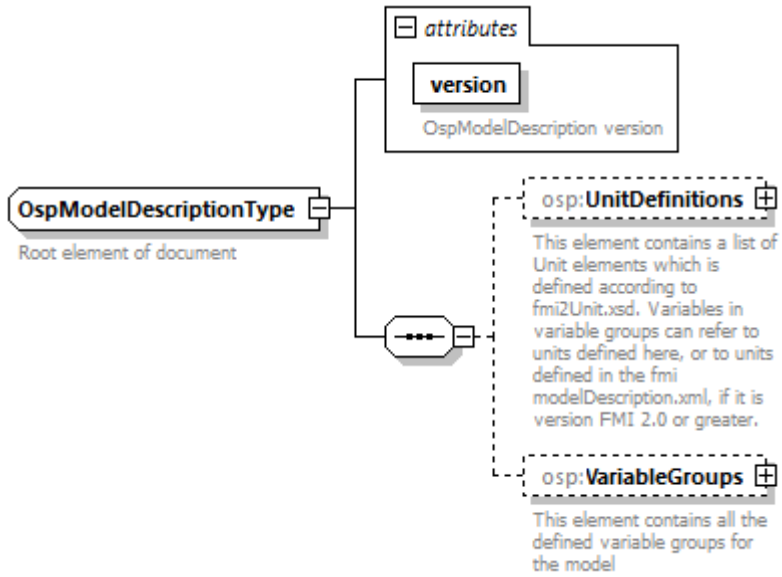
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Force osp:LinearDisplacement**
used by	elements **GenericType/LinearMechanicalQuasiPort** **VariableGroupsType/LinearMechanicalQuasiPort**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This group must contain exactly 1 Force and exactly 1 LinearDisplacement, where the causalities of the two groups must be opposite of each other. The xml schema validates that the element contains the correct child elements, and the OSP-IS validator verifies the causalities.

complexType **LinearVelocityType**

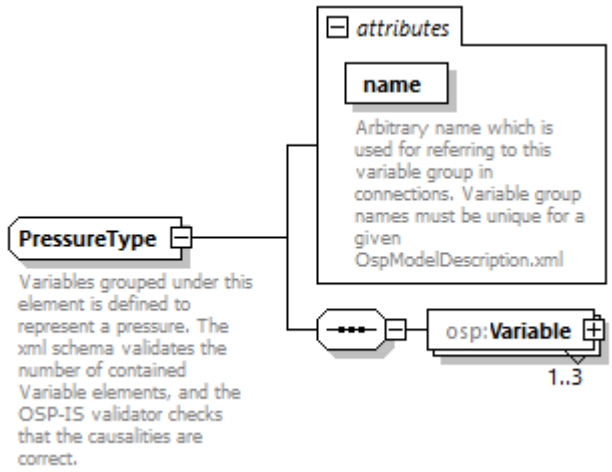
diagram	

namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **LinearMechanicalPortType/LinearVelocity GenericType/LinearVelocity VariableGroupsType/LinearVelocity**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: Variables grouped under this element is defined to represent a linear velocity. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

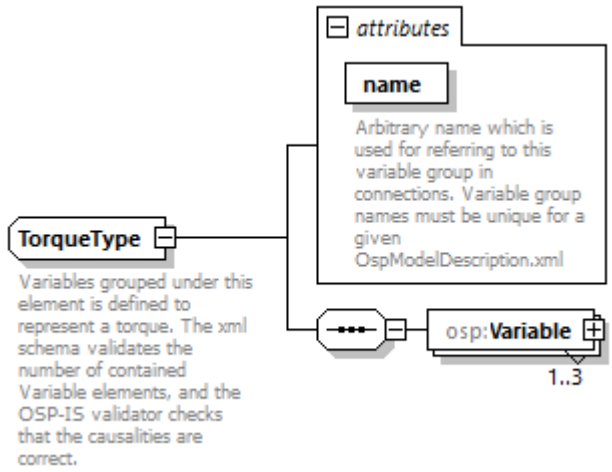
complexType **OspModelDescriptionType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:UnitDefinitions osp:VariableGroups**
used by	element **OspModelDescription**
attributes	Name: version; Type: **xs:string**; Use: required; Default: ; Fixed: 1.0; Annotation: documentation: OspModelDescription version
annotation	documentation: Root element of document

complexType **PressureType**

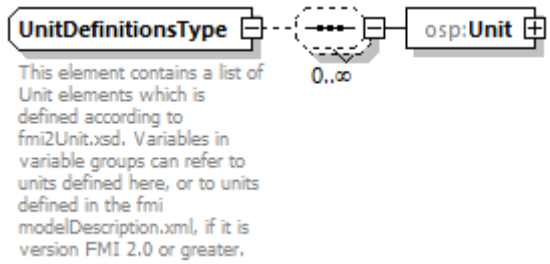
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **HydraulicPortType/Pressure HydraulicQuasiPortType/Pressure GenericType/Pressure VariableGroupsType/Pressure**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation Variables grouped under this element is defined to represent a pressure. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

complexType **TorqueType**

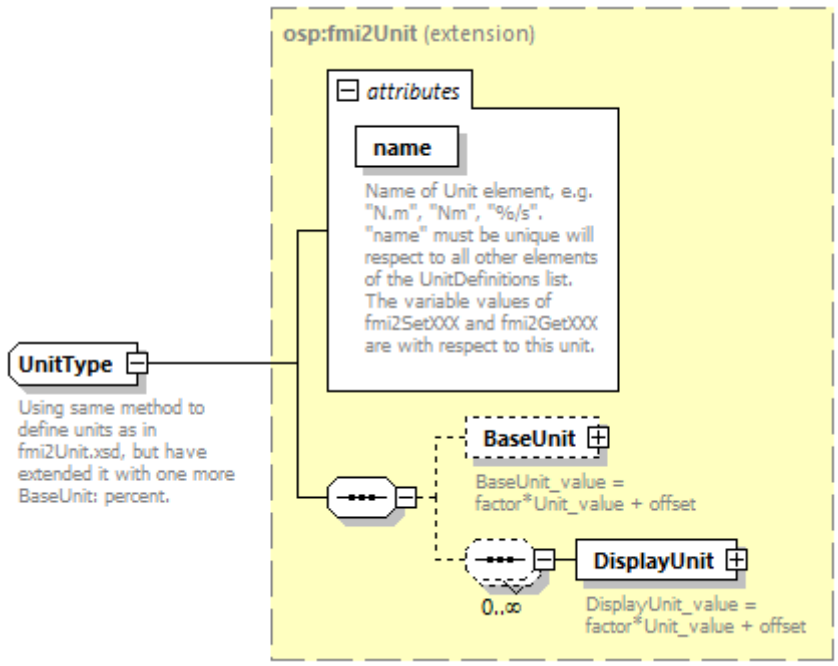
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **AngularMechanicalPortType/Torque AngularMechanicalQuasiPortType/Torque GenericType/Torque VariableGroupsType/Torque**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is

	used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation Variables grouped under this element is defined to represent a torque. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

complexType **UnitDefinitionsType**

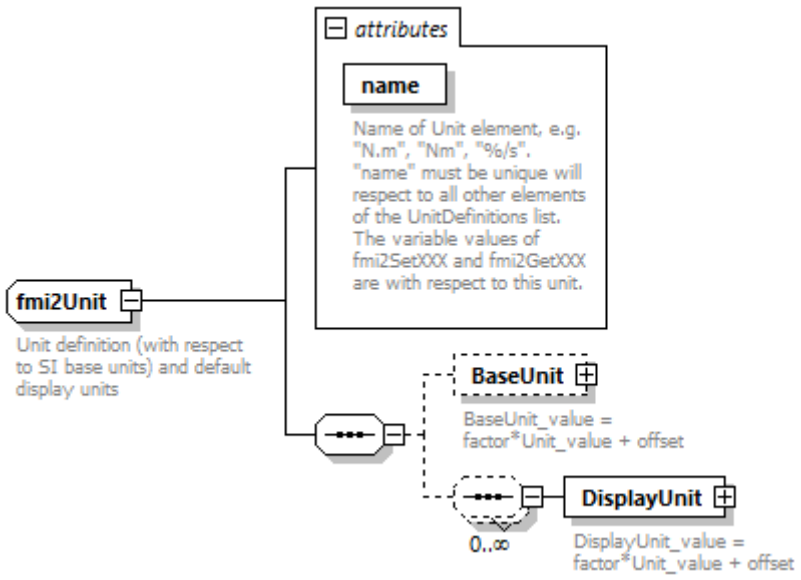
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Unit**
used by	element **OspModelDescriptionType/UnitDefinitions**
annotation	documentation This element contains a list of Unit elements which is defined according to fmi2Unit.xsd. Variables in variable groups can refer to units defined here, or to units defined in the fmi modelDescription.xml, if it is version FMI 2.0 or greater.

complexType **UnitType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
type	extension of **fmi2Unit**
properties	base osp:fmi2Unit

children	**BaseUnit DisplayUnit**
used by	element **UnitDefinitionsType/Unit**
attributes	Name: name; Type: **xs:normalizedString**; Use: required; Default: ; Fixed: ; Annotation: documentation: Name of Unit element, e.g. "N.m", "Nm", "%/s". "name" must be unique will respect to all other elements of the UnitDefinitions list. The variable values of fmi2SetXXX and fmi2GetXXX are with respect to this unit.
annotation	documentation: Using same method to define units as in fmi2Unit.xsd, but have extended it with one more BaseUnit: percent.

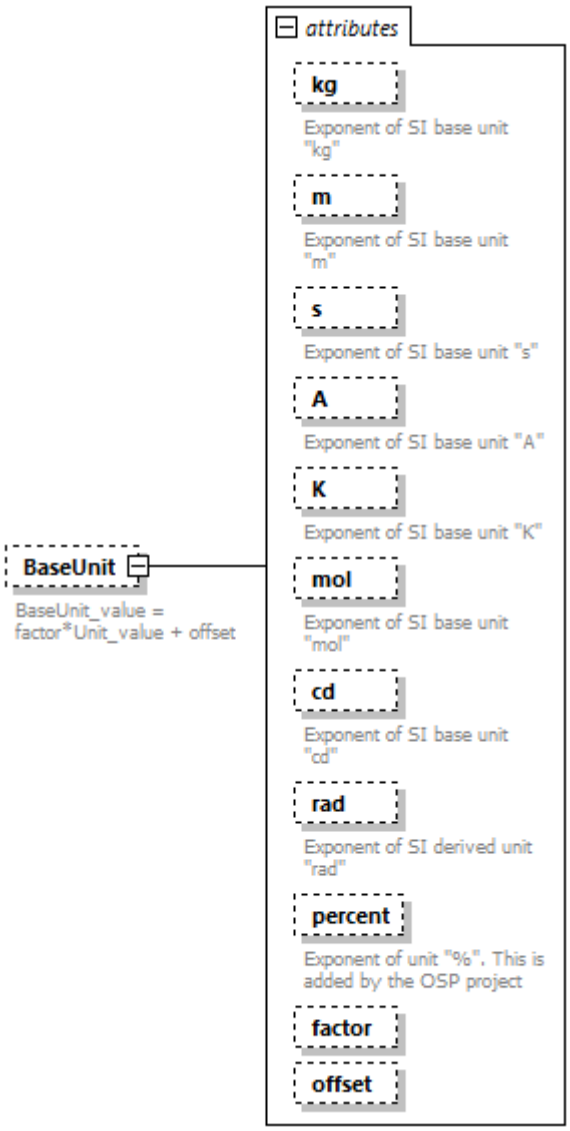
complexType **fmi2Unit**

diagram	
children	**BaseUnit DisplayUnit**
attributes	Name: name; Type: **xs:normalizedString**; Use: required; Default: ; Fixed: ; Annotation: documentation: Name of Unit element, e.g. "N.m", "Nm", "%/s". "name" must be unique will respect to all other elements of the UnitDefinitions list. The variable

		values of fmi2SetXXX and fmi2GetXXX are with respect to this unit.
annotation	documentation Unit definition (with respect to SI base units) and default display units	

element **fmi2Unit/BaseUnit**

diagram



properties

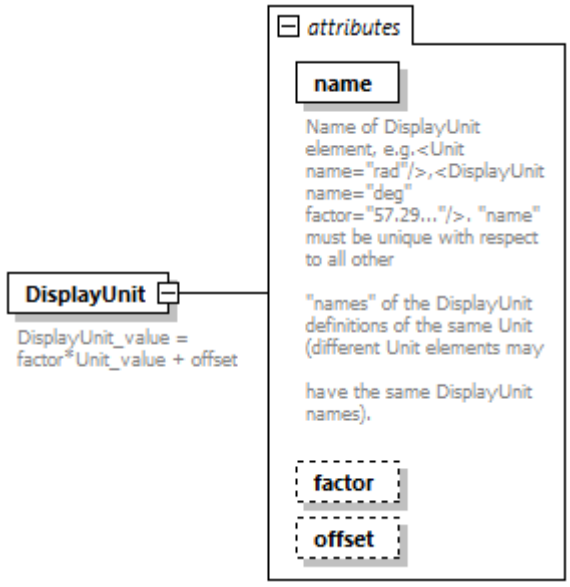
minOcc	0
maxOcc	1
content	complex

attributes

Name	Type	Use	Default	Fixed	Annotation
kg	**xs:int**		0		documentation Exponent of SI base unit "kg"
m	**xs:int**		0		documentation Exponent of SI base unit "m"

	s	**xs:int**	0	documentation Exponent of SI base unit "s"
	A	**xs:int**	0	documentation Exponent of SI base unit "A"
	K	**xs:int**	0	documentation Exponent of SI base unit "K"
	mol	**xs:int**	0	documentation Exponent of SI base unit "mol"
	cd	**xs:int**	0	documentation Exponent of SI base unit "cd"
	rad	**xs:int**	0	documentation Exponent of SI derived unit "rad"
	percent	**xs:int**	0	documentation Exponent of unit "%". This is added by the OSP project
	factor	**xs:double**	1	
	offset	**xs:double**	0	
annotation	documentation BaseUnit_value = factor*Unit_value + offset			

element **fmi2Unit/DisplayUnit**

diagram	
properties	content complex

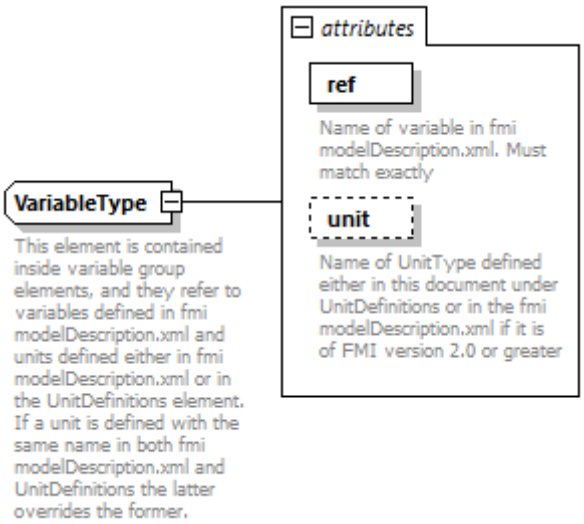
attributes	Name	Type	Use	Default	Fixed	Annotation
	name	**xs:normalizedString**	required			documentation Name of DisplayUnit element, e.g.<Unit name="rad"/>,<DisplayUnit name="deg" factor="57.29..."/>. "name" must be unique with respect to all other "names" of the DisplayUnit definitions of the same

					Unit (different Unit elements may have the same DisplayUnit names).
	factor	**xs:double**	1		
	offset	**xs:double**	0		
annotation	documentation DisplayUnit_value = factor*Unit_value + offset				

complexType **VariableGroupsType**

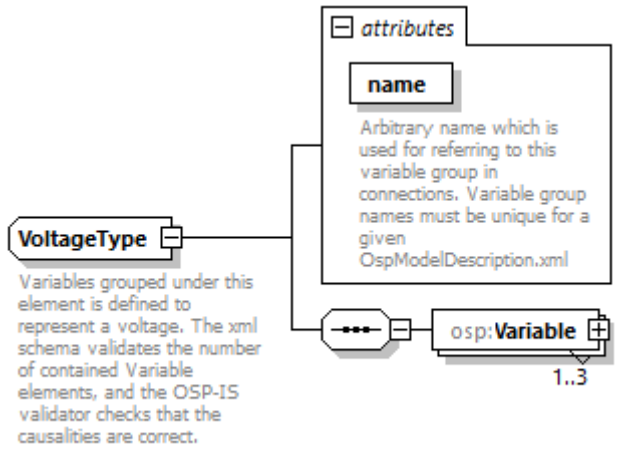
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**all defined variable groups**
used by	element **OspModelDescriptionType/VariableGroups**
annotation	documentation This element contains all the defined variable groups for the model

complexType **VariableType**

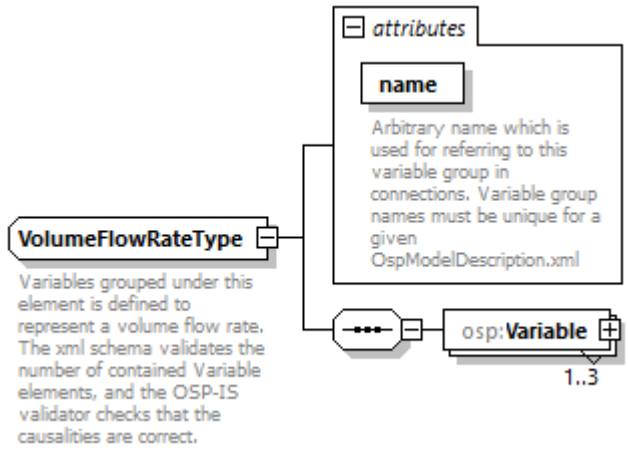
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
used by	elements **ForceType/Variable TorqueType/Variable VoltageType/Variable PressureType/Variable LinearVelocityType/Variable AngularVelocityType/Variable CurrentType/Variable VolumeFlowRateType/Variable LinearDisplacementType/Variable AngularDisplacementType/Variable ChargeType/Variable VolumeType/Variable LinearAccelerationType/Variable AngularAccelerationType/Variable ElectricPowerType/Variable FrequencyType/Variable NmeaTimeType/Variable NmeaStatusType/Variable NmeaGgaLatitudeLongitudeType/Variable NmeaGgaFixType/Variable NmeaGstRmsType/Variable NmeaGstEllipseType/Variable NmeaGstPositionErrorType/Variable NmeaWindDirectionType/Variable NmeaWindSpeedType/Variable NmeaTrueHeadingType/Variable BatteryFeedbackType/Variable ShaftSpeedType/Variable AzimuthAngleType/Variable BladePitchType/Variable GenericType/Variable**
attributes	Name / Type / Use / Default / Fixed / Annotation ref / **xs:string** / required / / / documentation: Name of variable in fmi modelDescription.xml. Must match exactly unit / / optional / / / documentation: Name of UnitType defined either in this

		document under UnitDefinitions or in the fmi modelDescription.xml if it is of FMI version 2.0 or greater
annotation	documentation This element is contained inside variable group elements, and they refer to variables defined in fmi modelDescription.xml and units defined either in fmi modelDescription.xml or in the UnitDefinitions element. If a unit is defined with the same name in both fmi modelDescription.xml and UnitDefinitions the latter overrides the former.	

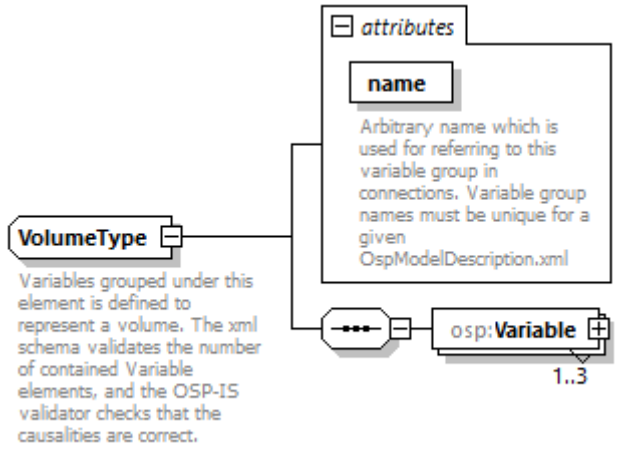
complexType **VoltageType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **ElectromagneticPortType/Voltage ElectromagneticQuasiPortType/Voltage GeneratorFeedbackType/Voltage BusFeedbackType/Voltage GenericType/Voltage VariableGroupsType/Voltage**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation Variables grouped under this element is defined to represent a voltage. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

complexType **VolumeFlowRateType**

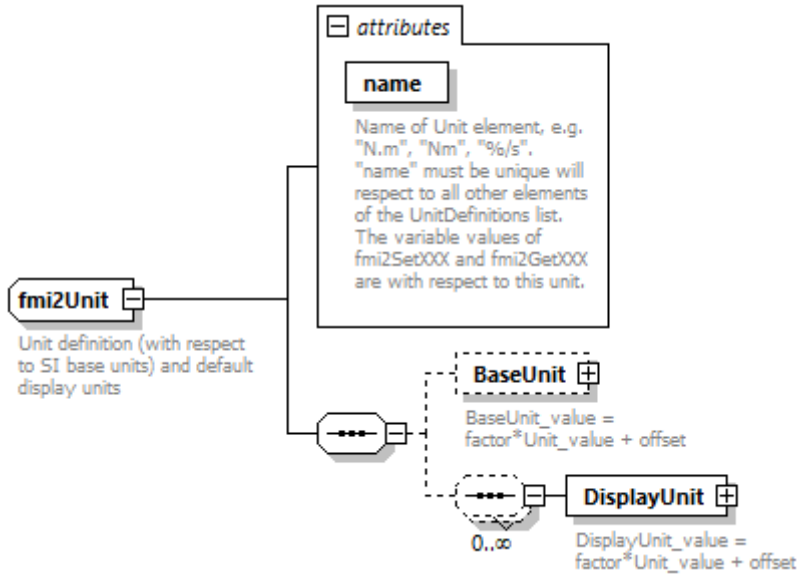
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **HydraulicPortType/VolumeFlowRate GenericType/VolumeFlowRate VariableGroupsType/VolumeFlowRate**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: Variables grouped under this element is defined to represent a volume flow rate. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

complexType **VolumeType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **HydraulicQuasiPortType/Volume GenericType/Volume VariableGroupsType/Volume**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in

	connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation Variables grouped under this element is defined to represent a volume. The xml schema validates the number of contained Variable elements, and the OSP-IS validator checks that the causalities are correct.

complexType **fmi2Unit**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**BaseUnit DisplayUnit**
used by	complexType **UnitType**
attributes	Name: name Type: **xs:normalizedString** Use: required Default: Fixed: Annotation: documentation Name of Unit element, e.g. "N.m", "Nm", "%/s". "name" must be unique will respect to all other elements of the UnitDefinitions list. The variable values of fmi2SetXXX and fmi2GetXXX are with respect to this unit.
annotation	documentation Unit definition (with respect to SI base units) and default display units

XML Schema documentation generated by **XMLSpy** Schema Editor **http://www.altova.com/xmlspy**

3.1.1 Example OspModelDescription.xml

Below is a simple example of what an OspModelDescription.xml file can look like. It defines four variable groups:

- A Generic variable group containing 3 variables
- A Force variable group containing 3 variables
- A LinearVelocity variable group containing 3 variables
- A LinearMechanicalPort variable group containing the Force and LinearVelocity variable groups

With a OspModelDescription.xml file like this, a simulation configuration tool can perform connections using 'actuator_limits', 'linear_mechanical_port', 'force' and 'linear_velocity' instead of all the variables individually.

```
<?xml version="1.0" encoding="utf-8" ?>
<OspModelDescription xmlns="https://open-simulation-platform.com/OspModelDescription/1.0.0"
version="1.0">
 <VariableGroups>
  <Generic name="actuator_limits">
   <Variable ref="Act_Limits[1]"/>
   <Variable ref="Act_Limits[2]"/>
   <Variable ref="Act_Limits[3]"/>
  </Generic>
  <LinearMechanicalPort name="linear_mechanical_port">
   <Force name="force">
    <Variable ref="p_Crane.e[1]"/>
    <Variable ref="p_Crane.e[2]"/>
    <Variable ref="p_Crane.e[3]"/>
   </Force>
   <LinearVelocity name="linear_velocity">
    <Variable ref="p_Crane.f[1]"/>
    <Variable ref="p_Crane.f[2]"/>
    <Variable ref="p_Crane.f[3]"/>
   </LinearVelocity>
  </LinearMechanicalPort>
 </VariableGroups>
</OspModelDescription>
```

4 ONTOLOGY

OSP-IS is implemented in the form of an ontology using the OWL 2 language. The ontology defines all the rules and semantics for any given variable group. This chapter describes how to implement a software tool which uses the OSP-IS ontology to validate OspModelDescription.xml files and simulation configurations. The OSP project has made a reference implementation of a validator, called osp-validator, which is presented in section 4.7.

Before we get into the details of the ontology, some quick OWL 2 definitions are needed [4]:

Class: A concept, group or set which individuals can belong to.

Individual: Represents a specific object or instance. Can belong to one or more classes and be related to other individuals through object properties.

Object property: Defines a relationship between two individuals.

The OSP-IS ontology contains all types defined in OSPModelDescription.xsd, and can be used to describe a single model, or a simulation configuration consisting of several simulators (models), where each sub-simulator contains variables and variable groups and are connected to other sub-simulators through variable and variable group connections. The ontology also defines a range of validation error classes which will describe - on a semantically higher level - why a simulation configuration is not valid according to OSP-IS. Validation using the OSP-IS ontology is done during classification, where the individuals describing the simulation configuration will be placed in one or several validation error classes if they are in violation of some of the rules defined in the ontology.

The OSP-IS ontology is implemented using OWL 2 and can be found in the osp.owl file documented and released here: https://data.dnvgl.com/osp/.

For the OSP-IS ontology to classify individuals correctly, it must be populated in the intended manner by the validation tool. The following sections list the classes, individuals and object properties which must be used during population, and specifically how individuals must be created.

4.1 CLASSES

Name	Description
Name	Class of individuals representing the names of simulators, variable groups and variables.
Simulator	Class of individuals representing simulators
Variable	Class of individuals representing variables
VariableGroup	Class of individuals representing variable groups
Unit	Class of individuals representing units
VariableConnection	Class of individuals representing variable connections
VariableGroupConnection	Class of individuals representing variable group connections

4.2 INDIVIDUALS

Name	Description
ind_causality_input	Individual representing input causality. Individuals of type Variable which has input

	causality must relate to this individual through the op_has_causality object property
ind_causality_output	Individual representing output causality. Individuals of type Variable which has output causality must relate to this individual through the op_has_causality object property
ind_datatype_boolean	Individual representing boolean datatype
ind_datatype_enum	Individual representing the enum datatype
ind_datatype_integer	Individual representing the integer datatype
ind_datatype_real	Individual representing the real datatype
ind-datatype_string	Individual representing the string datatype
ind_vgtype_generic	Individual representing the Generic variable group type
ind_vgtype_force	Individual representing the Force variable group type
ind_vgtype_torque	Individual representing the Torque variable group type
ind_vgtype_voltage	Individual representing the Voltage variable group type
ind_vgtype_pressure	Individual representing the Pressure variable group type
ind_vgtype_linear_velocity	Individual representing the LinearVelocity variable group type
ind_vgtype_angular_velocity	Individual representing the AngularVelocity variable group type
ind_vgtype_current	Individual representing the Current variable group type
ind_vgtype_volume_flow_rate	Individual representing the VolumeFlowRate variable group type
ind_vgtype_linear_displacement	Individual representing the LinearDisplacement variable group type
ind_vgtype_angular_displacement	Individual representing the AngularDisplacement variable group type
ind_vgtype_charge	Individual representing the Charge variable group type
ind_vgtype_volume	Individual representing the Volume variable group type
ind_vgtype_linear_mechanical_port	Individual representing the LinearMechanicalPort variable group type
ind_vgtype_angular_mechanical_port	Individual representing the AngularMechanicalPort variable group type
ind_vgtype_electromagnetic_port	Individual representing the ElectromagneticPort variable group type
ind_vgtype_hydraulic_port	Individual representing the HydraulicPort variable group type
ind_vgtype_linear_mechanical_quasi_port	Individual representing the LinearMechanicalQuasiPort variable group type
ind_vgtype_angular_mechanical_quasi_port	Individual representing the AngularMechanicalQuasiPort variable group type

ind_vgtype_electromagnetic_quasi_port	Individual representing the ElectroMagneticQuasiPort variable group type
ind_vgtype_hydraulic_quasi_port	Individual representing the HydraulicQuasiPort variable group type
ing_vgtype_linear_acceleration	Individual representing the LinearAcceleration variable group type
ing_vgtype_angular_acceleration	Individual representing the AngularAcceleration variable group type
ing_vgtype_nmea_gga_fix	Individual representing the NmeaGgaFix variable group type
ing_vgtype_nmea_gga_latitude_longitude	Individual representing the NmeaGgaLatitudeLongitude variable group type
ing_vgtype_nmea_gga	Individual representing the NmeaGga variable group type
ing_vgtype_nmea_gst_ellipse	Individual representing the NmeaGstEllipse variable group type
ing_vgtype_nmea_gst_position_error	Individual representing the NmeaGstPositionError variable group type
ing_vgtype_nmea_gst_rms	Individual representing the NmeaGstRms variable group type
ing_vgtype_nmea_gst	Individual representing the NmeaGst variable group type
ing_vgtype_nmea_mwv	Individual representing the NmeaMwv variable group type
ing_vgtype_nmea_status	Individual representing the NmeaStatus variable group type
ing_vgtype_nmea_sxn	Individual representing the NmeaSxn variable group type
ing_vgtype_nmea_ths	Individual representing the NmeaThs variable group type
ing_vgtype_nmea_time	Individual representing the NmeaTime variable group type
ing_vgtype_nmea_true_heading	Individual representing the NmeaTrueHeading variable group type
ing_vgtype_nmea_wind_direction	Individual representing the NmeaWindDirection variable group type
ing_vgtype_nmea_wind_speed	Individual representing the NmeaWindSpeed variable group type
ing_vgtype_azimuth_angle	Individual representing the AzimuthAngle variable group type
ing_vgtype_azimuth_thruster_feedback	Individual representing the AzimuthThrusterFeedback variable group type
ing_vgtype_azimuth_thruster_setpoint	Individual representing the AzimuthThrusterSetpoint variable group type
ing_vgtype_battery_feedback	Individual representing the BatteryFeedback variable group type
ing_vgtype_blade_pitch	Individual representing the BladePitch variable group type
ing_vgtype_bus_feedback	Individual representing the BusFeedback variable group type

ing_vgtype_electric_power	Individual representing the ElectricPower variable group type
ing_vgtype_fixed_thruster_setpoint	Individual representing the FixedThrusterSetpoint variable group type
ing_vgtype_fixed_thruster_feedback	Individual representing the FixedThrusterFeedback variable group type
ing_vgtype_frequency	Individual representing the Frequency variable group type
ing_vgtype _generator_feedback	Individual representing the GeneratorFeedback variable group type
ing_vgtype _shaft_speed	Individual representing the ShaftSpeed variable group type

4.3 OBJECT PROPERTIES

Name	**Description**
op_has_causality	This object property relates an individual of class Variable to one of the defined causality individuals
op_has_connection	This object property relates individuals of class VariableGroupConnection to individuals of class VariableGroupConnection or VariableConnection
op_has_datatype	This object property relates individuals of class Variable to one of the defined datatype individuals
op_has_name	This object property relates individuals of class Simulator, VariableGroup and Variable to individuals of class Name
op_has_unit	This object property relates individuals of class Variable to individuals of class Unit
op_has_variable	This object property relates individuals of class Simulator and VariableGroup to individuals of class Variable
op_has_variable_group	This object property relates individuals of class Simulator and VariableGroup to individuals of class VariableGroup
op_has_variable_group_type	This object property relates individuals of type VariableGroup to one of the defined vgtype individuals
op_has_lhs	This object property relates individuals of class VariableConnection to individuals of class Variable; or it relates individuals of class VariableGroupConnection to individuals of class VariableGroupConnection or VariableConnection
op_has_rhs	This object property relates individuals of class VariableConnection to individuals of class Variable; or it relates individuals of class VariableGroupConnection to individuals of class

	VariableGroupConnection or VariableConnection

4.4 POPULATING THE OSP-IS ONTOLOGY

This section describes how to populate the ontology based on information described in a OspModelDescription.xml.

Individual to add	**Population steps**
Simulator name	- Must have IRI (Internationalized Resource Identifier) on the following format: prefix:<simulator_name> - Must be an instance of class Name - Must have a 'different individuals axiom' to all other individuals in the ontology
Variable name	- Must have IRI on the following format: prefix:<simulator_name>.<variable_name> - Must be an instance of class Name - Must have a 'different individuals axiom' to all other individuals in the ontology
Variable group name	- Must have IRI on the following format: prefix:<simulator_name>.<variable_group_name> - Must be an instance of class Name - Must have a 'different individuals axiom' to all other individuals in the ontology
Simulator	- Must have a unique IRI - Must be an instance of class Simulator - Must be related to an individual representing the simulators name, through the op_has_name object property - Must be related to all individuals of class Variable which belongs to this simulator, through the op_has_variable object property - Must be related to all individuals of class VariableGroup which belongs to this simulator, though the op_has_variable_group object property - Must have a 'different individuals axiom' to all other individuals in the ontology
Unit	- Must have IRI on the following format: prefix:<kg_exponent_value>.<m_exponent_value>.<s_exponent_value>.<A_exponent_value>.<K_exponent_value>.<mol_exponent_value>.<cd_exponent_value>.<rad_exponent_value>.<percent_exponent_value> - Must be an instance of class Unit - Must have a 'different individuals axiom' to all other individuals in the ontology
Variable	- Must have a unique IRI - Must be an instance of class Variable

	- Must be related to the individual representing the name of the variable through the op_has_name object property - Must be related to one of the causality individuals through the op_has_causality object property - Must be related to one of the datatype individuals through the op_has_datatype object property - If the variable has a unit it must be related to the respective unit individual through the op_has_unit object property - Must have a 'different individuals axiom' to all other individuals in the ontology
Variable group	- Must have a unique IRI - Must be an instance of class VariableGroup - Must be related to all variable individuals which is contained inside this variable group, through the op_har_variable object property - Must be related to all nested variable group individuals which is contained inside this variable group (only one level deep), through the op_has_variable_group object property - Must be related to one of the ind_vgtype_x individuals through the op_has_variable_group_type object property to indicate the type of the variable group - Must have a 'different individuals axiom' to all other individuals in the ontology
Variable connection	- Must have a unique IRI - Must be an instance of class VariableConnection - Must be related to one of the variable individuals involved in the connection, through the op_has_lhs object property - Must be related to the other variable individual involved in the connection, through the op_has_rhs object property - Must have a 'different individuals axiom' to all other individuals in the ontology
Variable group connection	A variable group connection represents that all recursively contained variables in both variable groups are connected in the order in which they appear. This means that for variable groups containing variables (like the Force), the ordering of those variables matters when making connections. The resulting variable connections from a variable group connection are made from top to bottom. - Must have a unique IRI - Must be an instance of class VariableGroupConnection - Must be related to one of the variable group individuals in the connection, through the op_has_lhs object property

	- Must be related to the other variable group individual in the connection, through the op_has_rhs object property - Must be related to nested VariableConnection individuals through the op_has_connection object property - Must be related to nested VariableGroupConnection individuals through the op_has_connection object property

4.5 Model Validation Using OSP-IS Ontology

When validating a single OspModelDescription.xml, some validations are be done by the xml schema. However, there are a few things that requires the OSP-IS ontology for validation. They are shown below.

4.5.1 Model Validations

- All <Variable> elements inside <Force> have the same causality, data type, and base unit
- All <Variable> elements inside <Torque> have the same causality, data type and base unit
- All <Variable> elements inside <Voltage> have the same causality, data type and base unit
- All <Variable> elements inside <Pressure> have the same causality, data type and base unit
- All <Variable> elements inside <LinearVelocity> have the same causality, data type and base unit
- All <Variable> elements inside <LinearAcceleration> have the same causality, data type and base unit
- All <Variable> elements inside <AngularVelocity> have the same causality, data type and base unit
- All <Variable> elements inside <AngularAcceleration> have the same causality, data type and base unit
- All <Variable> elements inside <Current> have the same causality, data type and base unit
- All <Variable> elements inside <VolumeFlowRate> have the same causality, data type and base unit
- All <Variable> elements inside <LinearDisplacement> have the same causality, data type and base unit
- All <Variable> elements inside <AngularDisplacement> have the same causality, data type and base unit
- All <Variable> elements inside <Charge> have the same causality, data type and base unit
- All <Variable> elements inside <Volume> have the same causality, data type and base unit
- All <Variable> elements inside a <Force> element, which is inside a <LinearMechanicalPort> element, must have opposite causalities of the <Variable> elements inside the <LinearVelocity> element, also inside the <LinearMehcanicalPort> element

- All <Variable> elements inside a <Torque> element, which is inside a <AngularMechanicalPort> element, must have opposite causalities of the <Variable> elements inside the <AngularVelocity> element, also inside the <AngularMehcanicalPort> element
- All <Variable> elements inside a <Voltage> element, which is inside a <ElectromagneticPort> element, must have opposite causalities of the <Variable> elements inside the <Current> element, also inside the <ElectromagneticPort> element
- All <Variable> elements inside a <Pressure> element, which is inside a <HydraulicPort> element, must have opposite causalities of the <Variable> elements inside the <VolumeFlowRate> element, also inside the < HydraulicPort > element
- All <Variable> elements inside a <Force> element, which is inside a <LinearMechanicalQuasiPort> element, must have opposite causalities of the <Variable> elements inside the <LinearDisplacement> element, also inside the <LinearMehcanicalQuasiPort> element
- All <Variable> elements inside a <Torque> element, which is inside a <AngularMechanicalQuasiPort> element, must have opposite causalities of the <Variable> elements inside the <AngularDisplacement> element, also inside the <AngularMehcanicalQuasPort> element
- All <Variable> elements inside a <Voltage> element, which is inside a <ElectromagneticQuasiPort> element, must have opposite causalities of the <Variable> elements inside the <Charge> element, also inside the <ElectromagneticQuasiPort> element
- All <Variable> elements inside a <Pressure> element, which is inside a <HydraulicQuasiPort> element, must have opposite causalities of the <Variable> elements inside the <Volume> element, also inside the < HydraulicQuasiPort > element
- All variable group elements must have unique names
- All <Variable> elements must have unique names
- The variable groups documented in section 5.4 do not have validations implemented in the ontology for this version of the OSP-IS. However, they are still included in VariableGroupConnection validations to ensure that they are only connected to variable groups of the same type and opposite causality.

Validation of a OspModelDescription.xml using the OSP-IS ontology is done by populating the OSP-IS ontology with the information in OspModelDescription.xml according to section 4.4, and then performing classification. Invalid individuals will be classified into validation error classes, which can be retrieved from the ontology and further processed by the validator implementation to construct error messages.

4.6 Configuration Validation Using OSP-IS Ontology

When the OSP-IS ontology has been correctly populated with individuals representing the system configuration (simulators and how they are connected), performing classification will put relevant individuals into their respective ValidationError classes if there are any violated rules. If the ontology becomes inconsistent, this is because of an error in the validator implementation. The classification process will look at the system configuration and put individuals into certain ValidationError classes which indicates the type of error they are associated with. The ValidationError types are as follows:

- Simulator does not have a unique name
- Variable does not have a unique name
- Variable group does not have a unique name
- Variable group connection connects variable groups of different types
- Variable connection is between variables of incompatible units
- Variable connection is between variables of incompatible causalities
- Variable connection is between variables of incompatible data types

4.6.1 Configuration Validations

- All Simulators must have unique names
- A VariableConnection can only connect two variables of the same data type, opposite causality, and same base unit
- A VariableGroupConnection can only connect two variable groups of the same type whose internal variables have opposite causalities.

4.7 OSP-Validator: A Reference Implementation

As part of the OSP project, a validator reference implementation has been made, called osp-validator. It is a java implementation found here: https://github.com/open-simulation-platform/osp-validator, which consists of a java library used in the configuration application kopl (https://open-simulation-platform.github.io/kopl), and a command line interface tool which is used by libcosim (https://open-simulation-platform.github.io/libcosim) to validate simulation configurations according to OSP-IS. The osp-validator reads OspModelDescription.xml and OspSystemStructure.xml; validates them according to their xml schemas; populates the OSP-IS ontology; performs classification to check for validation errors; and finally generates appropriate error messages. Visit: https://open-simulation-platform.github.io/osp-validator to read more about how to download and use the osp-validator.

5 PRINCIPLES FOR INTERFACING WITH CONTROL SYSTEMS

Connections between control systems and simulators are fundamentally different from connections between simulation models. Instead of physical couplings, they represent the exchange of information and, unlike with physical quantities, information is more difficult to standardize and categorize.

This exchange of information between the control system and several pieces of equipment (sensors and actuators) takes place via several interfaces. While there are a number of individual communication protocols and signal standards (hardwired signal standards such as 4-20mA, 1-5V, etc.) that can be used, there are typically no standards regulating which interfaces a given controller must have. Because of this, the same type of equipment may implement different communication protocols or signal standards when delivered by different vendors. And while there also is a range of typical interface signals found for each equipment type, any specific interface can include different combinations and subsets in arbitrary orders.

The lack of a (higher-level) interface standard on one hand and the use of numerous signal protocols on the other make it difficult to find a general method for interfacing simulators and control systems. To remedy this situation and to facilitate easy connections, we conceptually split the interface into two layers:

1. An **information layer** which takes care of information generation and interpretation. This is done via *sensor* and *actuator* FMUs which translate between physical quantities from the simulation (such as positions and velocities) and control-system-friendly signals (such as sensor readings). The sensor and actuator models are coupled to the remainder of the simulator through the variable groups defined in Chapter 5.4.
2. A **communication layer** which takes care of the communication between the simulator and the control system. This is done via *network* FMUs converting the signals into the correct protocols and formats used by the controllers. The network FMUs are coupled to the sensor and actuator FMUs through dedicated variable groups, which we will define shortly in Section 5.4., and to the control system via their preferred protocols.

In this way, we can standardize the interface between control systems and simulators by defining how these two layers are coupled for various types of equipment via variable groups. This is the focus of the present chapter.

Aside from this, we'll also discusses an example implementation centred around a DP controller and end with a brief overview of challenges and future work. But first, let us explain the conceptual split into information and communication layers in more detail.

5.1 THE INFORMATION LAYER: SENSOR AND ACTUATOR FMUS

Sensors and actuators form the gateway between the physical world and the control systems. Sensors are used to gather relevant information, while actuators enable a control system to manipulate systems and equipment. The OSP-IS suggest the use of sensor and actuator models to simulate the conversion between the physical signals in the simulation and the information and commands used by control systems.

Real-life sensors should be modelled (as FMUs) in a way where they generate data without any knowledge for what the data is used for. They take in physical signals from the simulation (e.g. position and velocity from a vessel model) and add sensor-specific characteristics (such as a noise

profile) and information specific to the sensor's data protocol. Their connection towards the control system is then simplified by defining dedicated variable groups matching the interfaces of real sensors.

Real-life actuators should be modelled (as FMUs) such that they exchange control system commands and equipment feedback with the simulation models. The connection towards the control system is again simplified by defining dedicated variable groups matching the interfaces of real actuators.

5.2 The Communication Layer: Network FMUs

While the sensor and actuator FMUs take care of the information generation and interpretation, they do not take care of the communication with the control system itself. That task is carried out by dedicated units: network FMUs. Network FMUs are responsible for formatting information correctly for the control system. They have an FMI variable interface towards the simulator side, and a network interface towards the controller on the other.

With some exceptions, they should be created by the control system supplier or developer. This way, they can choose the precise communication protocol they want to use. A control system supplier will thus have to take care of how the sensor and actuator FMU signals are translated into the correct communication protocol used by their system (for example, NMEA, HTTP, or UDP).

5.3 Example System: DP Controller

As promised, let's have a look at how these concepts come together for a realistic case with a DP controller. While DP controllers don't have an official standard defining their interface, they mostly follow the NMEA protocol (https://www.nmea.org/) when communicating with external equipment. In this section we define the relevant NMEA messages as variable groups and demonstrate how to construct the interface between controller and simulation using OSP-IS.

Figure 1 shows the example system and the clear conceptual separation between simulation (blue) and control system (green), as well as the separation into information layer (sensor FMUs in dark blue) and communication layer (network FMUs in dark green).

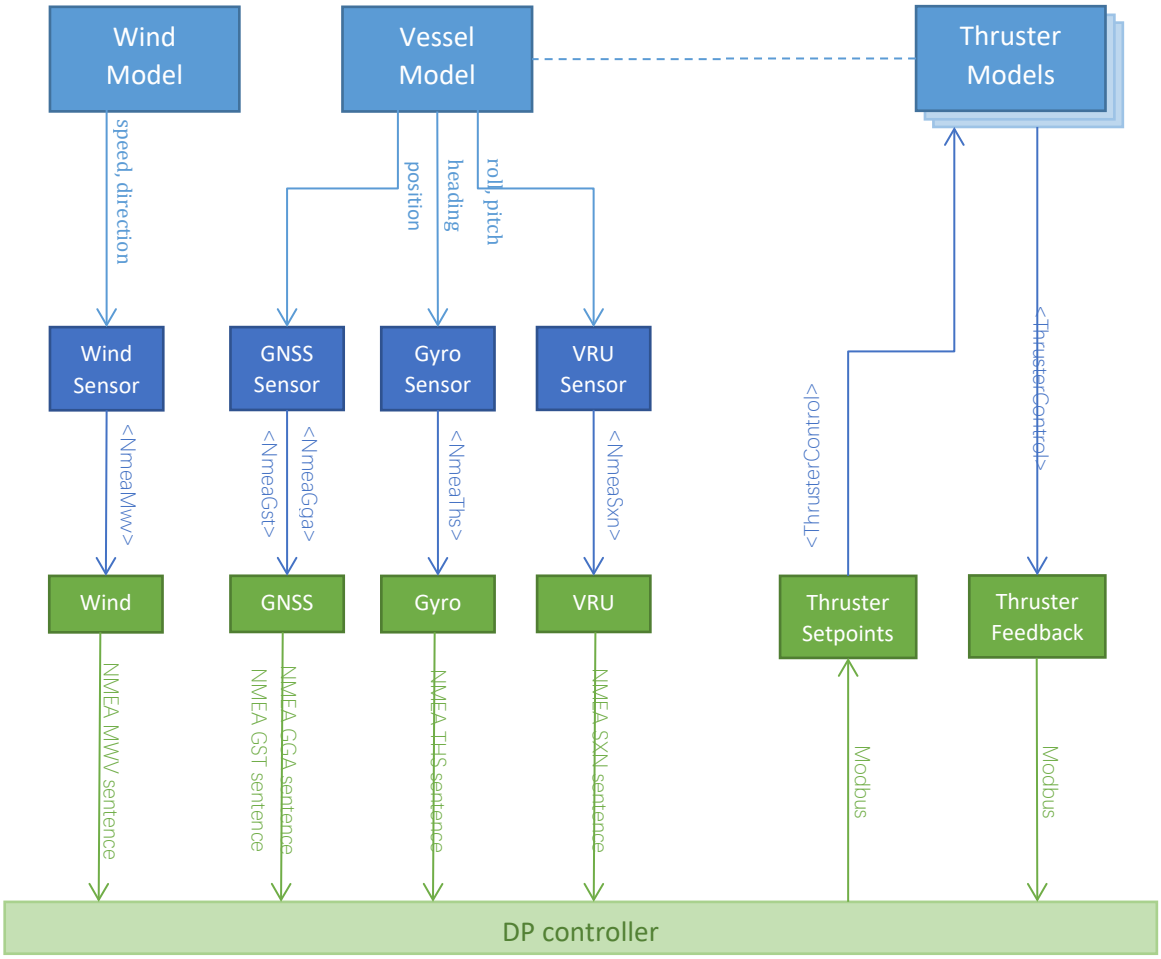


Figure 1: Example system with DP controller. The vessel and wind models (light blue) send physical signals to a group of sensor FMUs (dark blue) which add sensor noise and characteristics and pass the generated information on to a group of network FMUs (green). These provide the actual communication with the control system by using the correct protocols and formats. Actuators are modelled as FMUs as well to provide an interface between network FMUs (and thus the controller) and simulation models.

Let us walk through a few concrete examples for how the interfacing principles introduced above are used for this system:

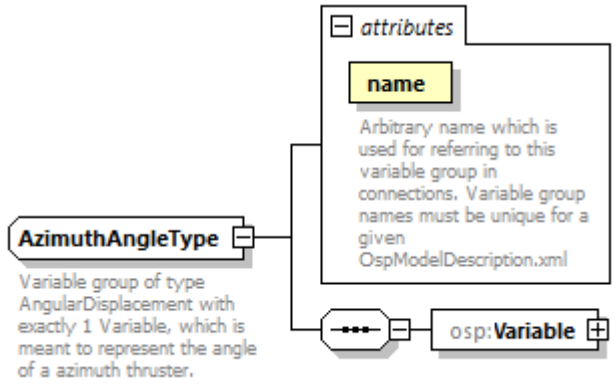
- The *Wind Model* (light blue) outputs wind speed and direction to the *Wind* sensor FMU (dark blue) using the variable groups defined in Chapter 3. The Wind sensor FMU simulates the characteristics of such a sensor and generates the information needed to construct NMEA MWV sentences. It then outputs these signals as variable groups as defined in.5.4. The *Wind* network FMU (dark green) receives them, constructs NMEA MWV communication packages based on them, and sends them off to the *DP control system*.
- The *Vessel* FMU (light blue) outputs its position to the *GNSS sensor* model (dark blue) using the variable groups defined in Chapter 3. The *GNSS sensor* model simulates position measurements based on its own placement on the ship, adds the relevant protocol signals for complete NMEA GGA and GST messages, and outputs the signals using the variable groups as defined in Section 5.4. The *GNSS network* FMU constructs the NMEA GGA and GST communication packages and sends them to the DP control system.

- The *DP controller* also communicates with the *Thruster Models* (light blue) by sending and receiving setpoints (e.g. desired RPM) and feedbacks (e.g. actual RPM), respectively. This is done via the *Thruster Setpoints* and *Thruster Feedback* network FMUs which take care of the communications with the DP controller. Typical protocols in use can be Modbus or CAN bus, while the packages can differ greatly between suppliers and thruster types.

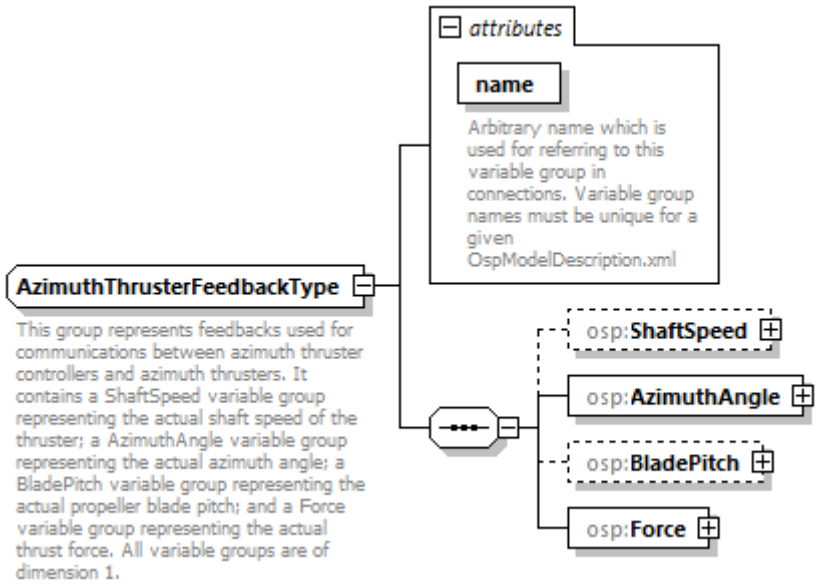
5.4 An Interface for DP Controllers

In this section, we finally define the variable groups used in the example above. These definitions are merely the beginning of the set of definitions needed to cover most control systems. For now, only a DP controller is supported. The ultimate goal is to provide interfaces for most of the control systems, sensors, and actuators used in the maritime domain.

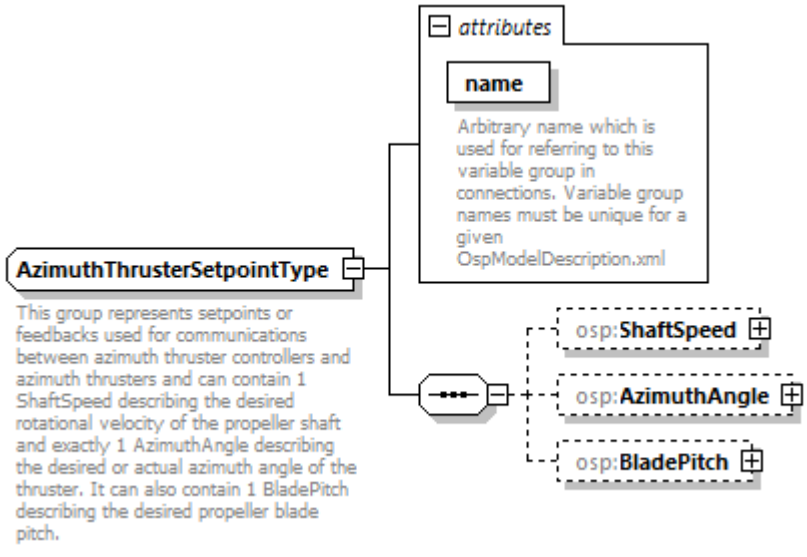
complexType **AzimuthAngleType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
type	restriction of **osp:AngularDisplacementType**
properties	base osp:AngularDisplacementType
children	**osp:Variable**
used by	elements **AzimuthThrusterSetpointType/AzimuthAngle AzimuthThrusterFeedbackType/AzimuthAngle GenericType/AzimuthAngle VariableGroupsType/AzimuthAngle**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: Variable group of type AngularDisplacement with exactly 1 Variable, which is meant to represent the angle of a azimuth thruster.

complexType **AzimuthThrusterFeedbackType**

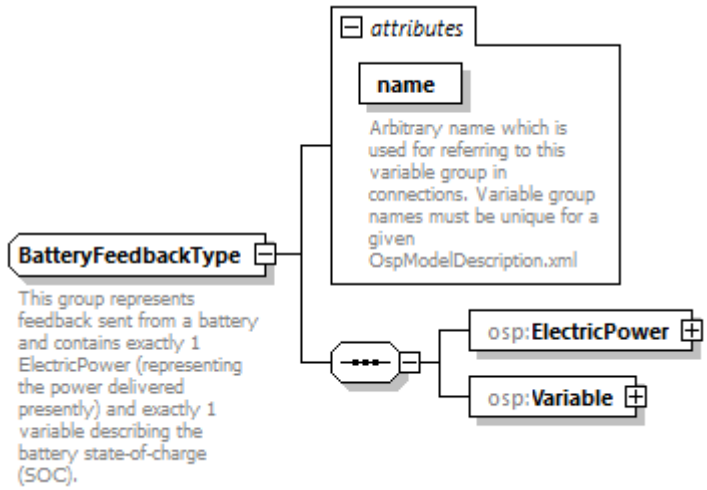
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:ShaftSpeed osp:AzimuthAngle osp:BladePitch osp:Force**
used by	elements **GenericType/AzimuthThrusterFeedback VariableGroupsType/AzimuthThrusterFeedback**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group represents feedbacks used for communications between azimuth thruster controllers and azimuth thrusters. It contains a ShaftSpeed variable group representing the actual shaft speed of the thruster; a AzimuthAngle variable group representing the actual azimuth angle; a BladePitch variable group representing the actual propeller blade pitch; and a Force variable group representing the actual thrust force. All variable groups are of dimension 1.

complexType **AzimuthThrusterSetpointType**

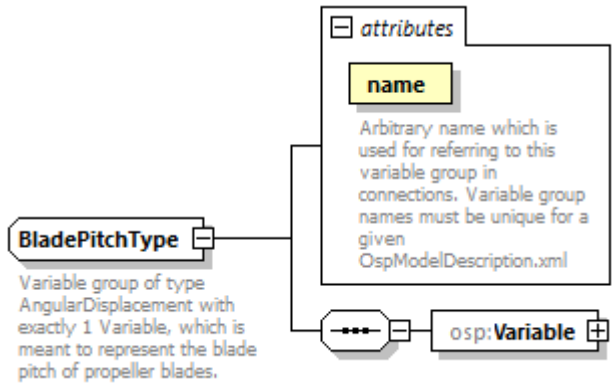
diagram	

namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:ShaftSpeed osp:AzimuthAngle osp:BladePitch**
used by	elements **GenericType/AzimuthThrusterSetpoint VariableGroupsType/AzimuthThrusterSetpoint**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group represents setpoints or feedbacks used for communications between azimuth thruster controllers and azimuth thrusters and can contain 1 ShaftSpeed describing the desired rotational velocity of the propeller shaft and exactly 1 AzimuthAngle describing the desired or actual azimuth angle of the thruster. It can also contain 1 BladePitch describing the desired propeller blade pitch.

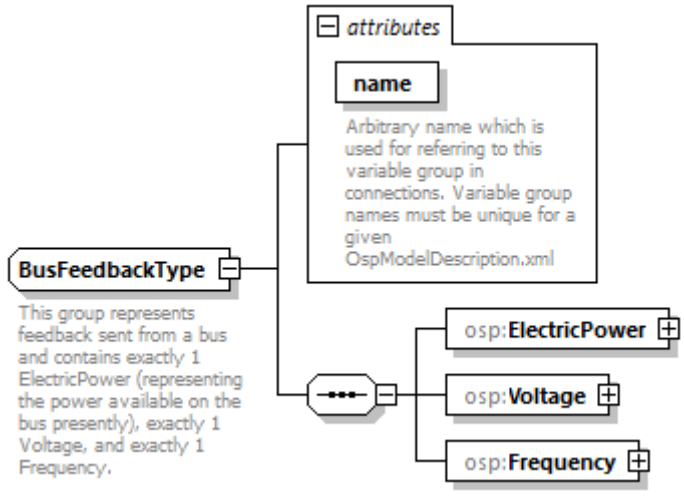
complexType **BatteryFeedbackType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:ElectricPower osp:Variable**
used by	elements **GenericType/BatteryFeedback VariableGroupsType/BatteryFeedback**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group represents feedback sent from a battery and contains exactly 1 ElectricPower (representing the power delivered presently) and exactly 1 variable describing the battery state-of-charge (SOC).

complexType **BladePitchType**

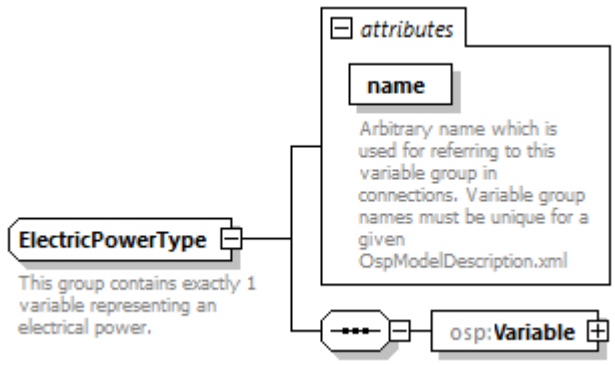
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
type	restriction of **osp:AngularDisplacementType**
properties	base osp:AngularDisplacementType
children	**osp:Variable**
used by	elements **FixedThrusterSetpointType/BladePitch FixedThrusterFeedbackType/BladePitch AzimuthThrusterSetpointType/BladePitch AzimuthThrusterFeedbackType/BladePitch GenericType/BladePitch VariableGroupsType/BladePitch**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: Variable group of type AngularDisplacement with exactly 1 Variable, which is meant to represent the blade pitch of propeller blades.

complexType **BusFeedbackType**

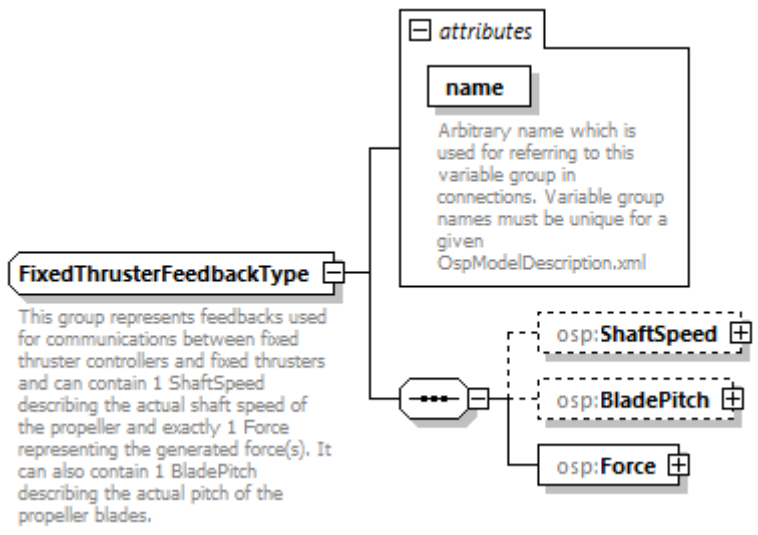
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:ElectricPower osp:Voltage osp:Frequency**
used by	elements **GenericType/BusFeedback VariableGroupsType/BusFeedback**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is

		used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This group represents feedback sent from a bus and contains exactly 1 ElectricPower (representing the power available on the bus presently), exactly 1 Voltage, and exactly 1 Frequency.	

complexType **ElectricPowerType**

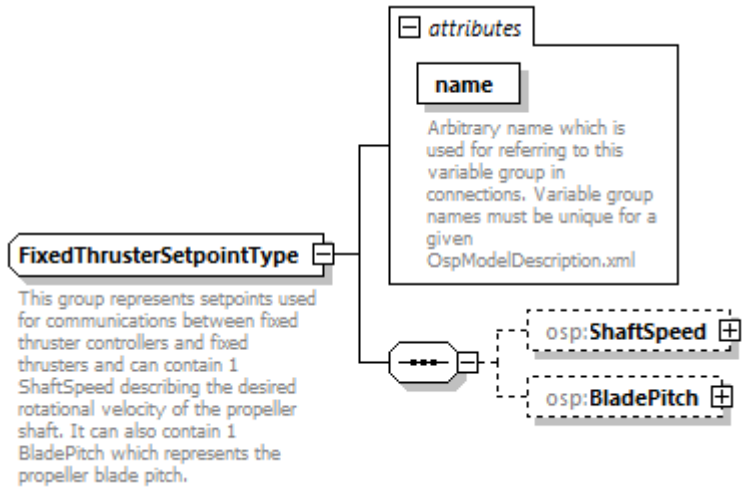
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **BatteryFeedbackType/ElectricPower GeneratorFeedbackType/ElectricPower BusFeedbackType/ElectricPower GenericType/ElectricPower VariableGroupsType/ElectricPower**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This group contains exactly 1 variable representing an electrical power.

complexType **FixedThrusterFeedbackType**

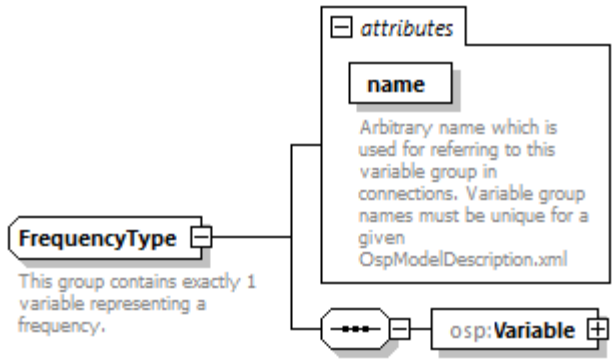
diagram	

namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:ShaftSpeed osp:BladePitch osp:Force**
used by	elements **GenericType/FixedThrusterFeedback VariableGroupsType/FixedThrusterFeedback**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group represents feedbacks used for communications between fixed thruster controllers and fixed thrusters and can contain 1 ShaftSpeed describing the actual shaft speed of the propeller and exactly 1 Force representing the generated force(s). It can also contain 1 BladePitch describing the actual pitch of the propeller blades.

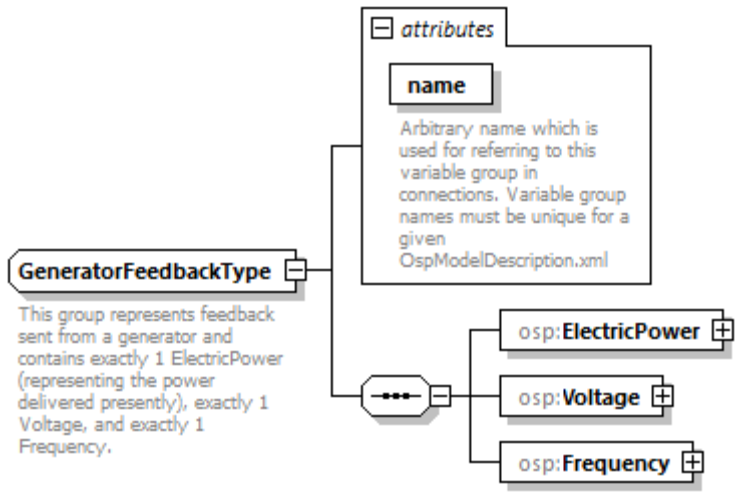
complexType **FixedThrusterSetpointType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:ShaftSpeed osp:BladePitch**
used by	elements **GenericType/FixedThrusterSetpoint VariableGroupsType/FixedThrusterSetpoint**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group represents setpoints used for communications between fixed thruster controllers and fixed thrusters and can contain 1 ShaftSpeed describing the desired rotational velocity of the propeller shaft. It can also contain 1 BladePitch which represents the propeller blade pitch.

complexType **FrequencyType**

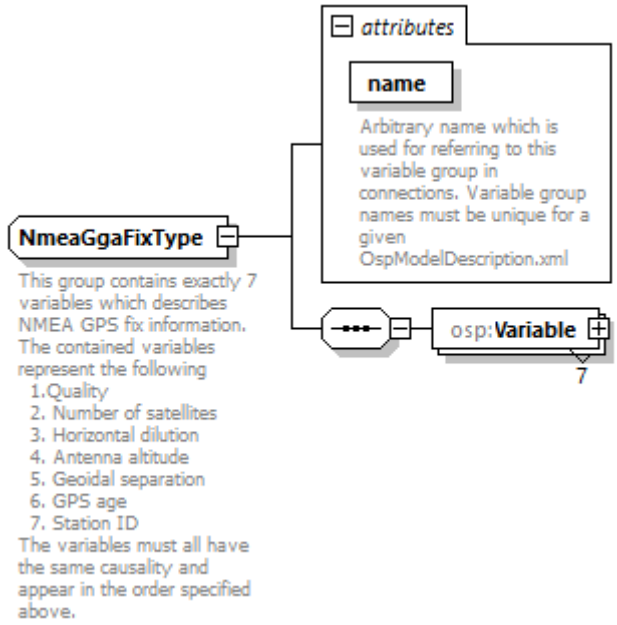
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **GeneratorFeedbackType/Frequency BusFeedbackType/Frequency GenericType/Frequency VariableGroupsType/Frequency**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group contains exactly 1 variable representing a frequency.

complexType **GeneratorFeedbackType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:ElectricPower osp:Voltage osp:Frequency**
used by	elements **GenericType/GeneratorFeedback VariableGroupsType/GeneratorFeedback**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml

annotation	documentation This group represents feedback sent from a generator and contains exactly 1 ElectricPower (representing the power delivered presently), exactly 1 Voltage, and exactly 1 Frequency.

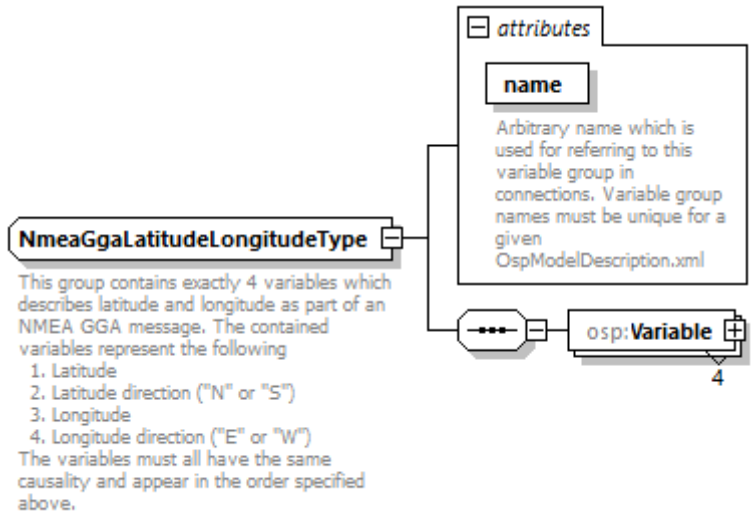
complexType **NmeaGggaFixType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **NmeaGgaType/NmeaGgaFix GenericType/NmeaGgaFix VariableGroupsType/NmeaGgaFix**

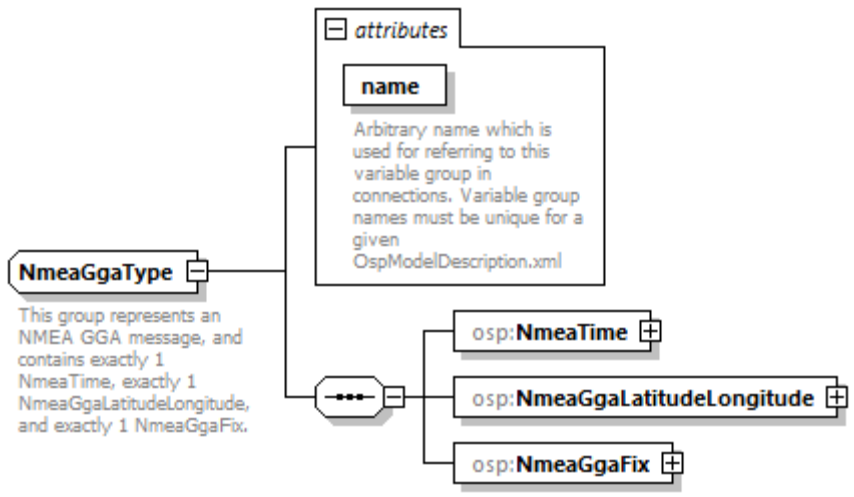
attributes	Name	Type	Use	Default	Fixed	Annotation
	name	**xs:string**	required			documentation Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml

annotation	documentation This group contains exactly 7 variables which describes NMEA GPS fix information. The contained variables represent the following 1.Quality 2. Number of satellites 3. Horizontal dilution 4. Antenna altitude 5. Geoidal separation 6. GPS age 7. Station ID The variables must all have the same causality and appear in the order specified above.

complexType **NmeaGgaLatitudeLongitudeType**

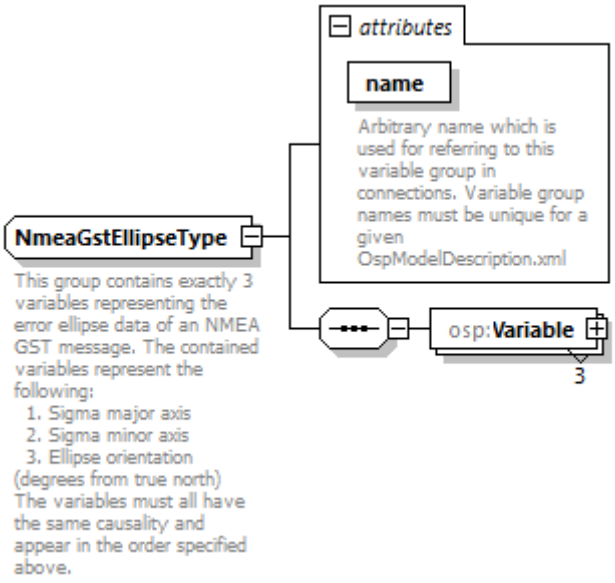
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **NmeaGgaType/NmeaGgaLatitudeLongitude GenericType/NmeaGgaLatitudeLongitude VariableGroupsType/NmeaGgaLatitudeLongitude**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This group contains exactly 4 variables which describes latitude and longitude as part of an NMEA GGA message. The contained variables represent the following 1. Latitude 2. Latitude direction ("N" or "S") 3. Longitude 4. Longitude direction ("E" or "W") The variables must all have the same causality and appear in the order specified above.

complexType **NmeaGgaType**

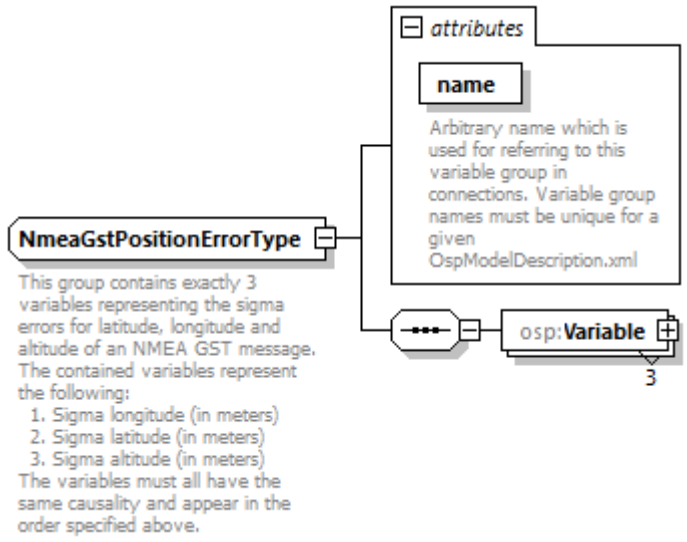
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0

children	**osp:NmeaTime osp:NmeaGgaLatitudeLongitude osp:NmeaGgaFix**
used by	elements **GenericType/NmeaGga VariableGroupsType/NmeaGga**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group represents an NMEA GGA message, and contains exactly 1 NmeaTime, exactly 1 NmeaGgaLatitudeLongitude, and exactly 1 NmeaGgaFix.

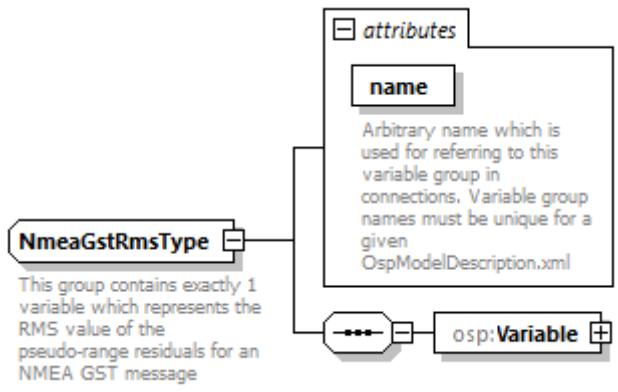
complexType **NmeaGstEllipseType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **NmeaGstType/NmeaGstEllipse GenericType/NmeaGstEllipse VariableGroupsType/NmeaGstEllipse**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group contains exactly 3 variables representing the error ellipse data of an NMEA GST message. The contained variables represent the following: 1. Sigma major axis 2. Sigma minor axis 3. Ellipse orientation (degrees from true north) The variables must all have the same causality and appear in the order specified above.

complexType **NmeaGstPositionErrorType**

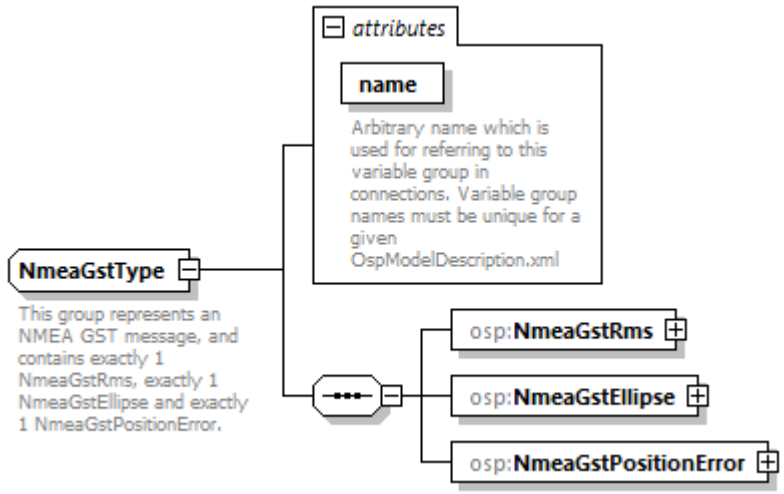
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **NmeaGstType/NmeaGstPositionError GenericType/NmeaGstPositionError VariableGroupsType/NmeaGstPositionError**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group contains exactly 3 variables representing the sigma errors for latitude, longitude and altitude of an NMEA GST message. The contained variables represent the following: 1. Sigma longitude (in meters) 2. Sigma latitude (in meters) 3. Sigma altitude (in meters) The variables must all have the same causality and appear in the order specified above.

complexType **NmeaGstRmsType**

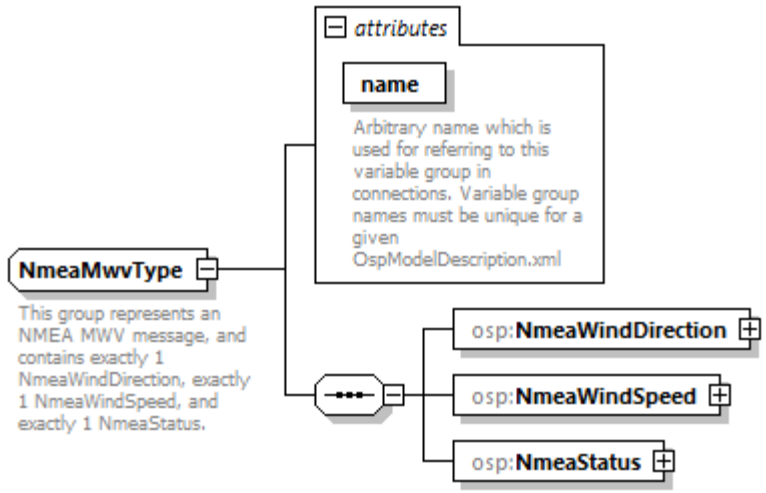
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **NmeaGstType/NmeaGstRms GenericType/NmeaGstRms VariableGroupsType/NmeaGstRms**
attributes	Name; Type; Use; Default; Fixed; Annotation

	name	**xs:string**	required	documentation Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This group contains exactly 1 variable which represents the RMS value of the pseudo-range residuals for an NMEA GST message			

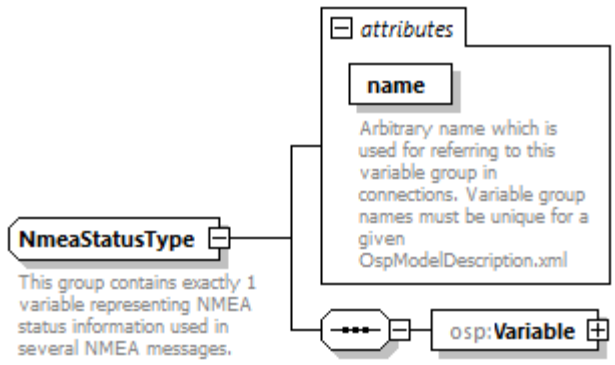
complexType **NmeaGstType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:NmeaGstRms osp:NmeaGstEllipse osp:NmeaGstPositionError**
used by	elements **GenericType/NmeaGst VariableGroupsType/NmeaGst**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This group represents an NMEA GST message, and contains exactly 1 NmeaGstRms, exactly 1 NmeaGstEllipse and exactly 1 NmeaGstPositionError.

complexType **NmeaMwvType**

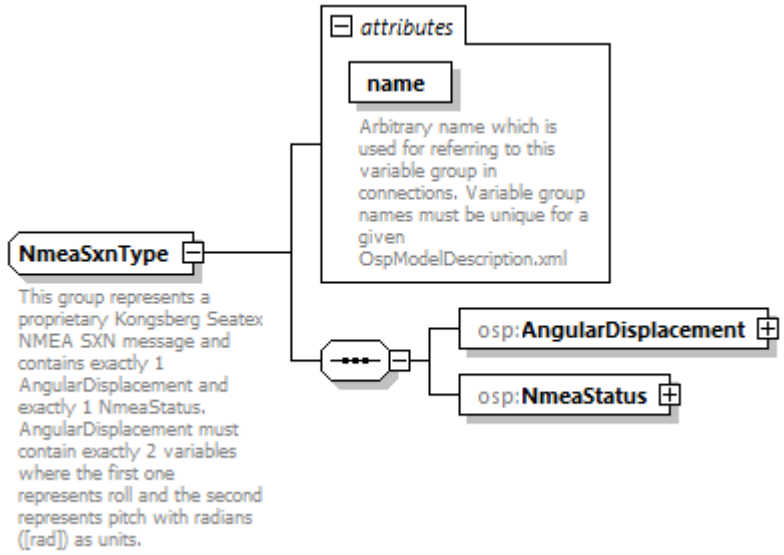
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:NmeaWindDirection osp:NmeaWindSpeed osp:NmeaStatus**
used by	elements **GenericType/NmeaMwv VariableGroupsType/NmeaMwv**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This group represents an NMEA MWV message, and contains exactly 1 NmeaWindDirection, exactly 1 NmeaWindSpeed, and exactly 1 NmeaStatus.

complexType **NmeaStatusType**

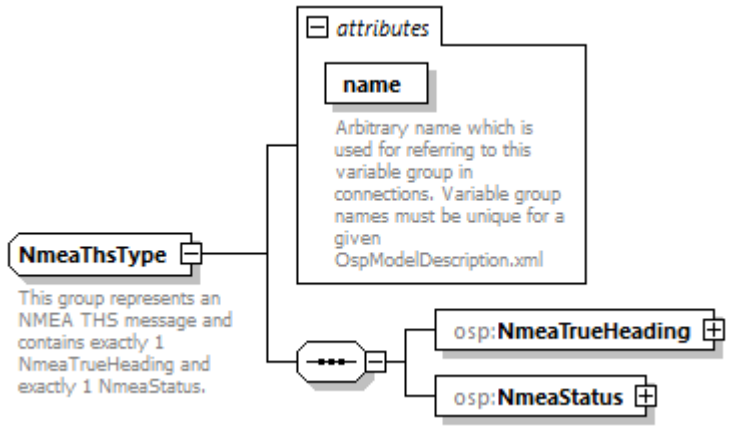
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **NmeaMwvType/NmeaStatus NmeaThsType/NmeaStatus NmeaSxnType/NmeaStatus GenericType/NmeaStatus VariableGroupsType/NmeaStatus**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given

	OspModelDescription.xml
annotation	documentation This group contains exactly 1 variable representing NMEA status information used in several NMEA messages.

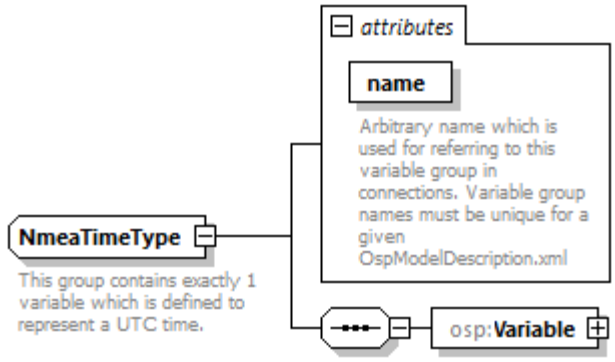
complexType **NmeaSxnType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:AngularDisplacement osp:NmeaStatus**
used by	elements **GenericType/NmeaSxn VariableGroupsType/NmeaSxn**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This group represents a proprietary Kongsberg Seatex NMEA SXN message and contains exactly 1 AngularDisplacement and exactly 1 NmeaStatus. AngularDisplacement must contain exactly 2 variables where the first one represents roll and the second represents pitch with radians ([rad]) as units.

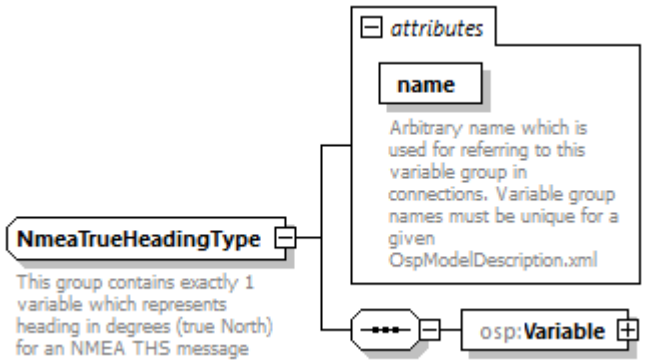
complexType **NmeaThsType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:NmeaTrueHeading osp:NmeaStatus**
used by	elements **GenericType/NmeaThs VariableGroupsType/NmeaThs**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This group represents an NMEA THS message and contains exactly 1 NmeaTrueHeading and exactly 1 NmeaStatus.

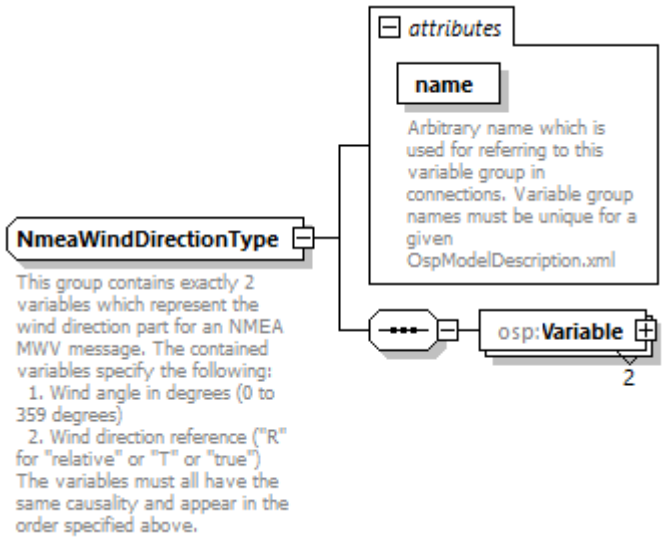
complexType **NmeaTimeType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **NmeaGgaType/NmeaTime GenericType/NmeaTime VariableGroupsType/NmeaTime**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This group contains exactly 1 variable which is defined to represent a UTC time.

complexType **NmeaTrueHeadingType**

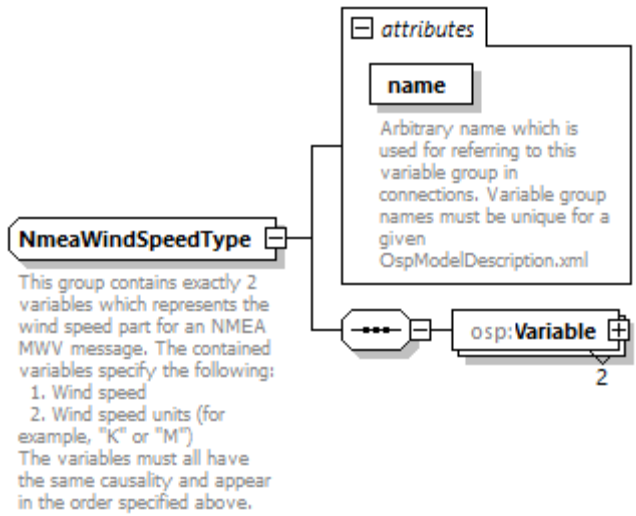
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **NmeaThsType/NmeaTrueHeading GenericType/NmeaTrueHeading VariableGroupsType/NmeaTrueHeading**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: This group contains exactly 1 variable which represents heading in degrees (true North) for an NMEA THS message

complexType **NmeaWindDirectionType**

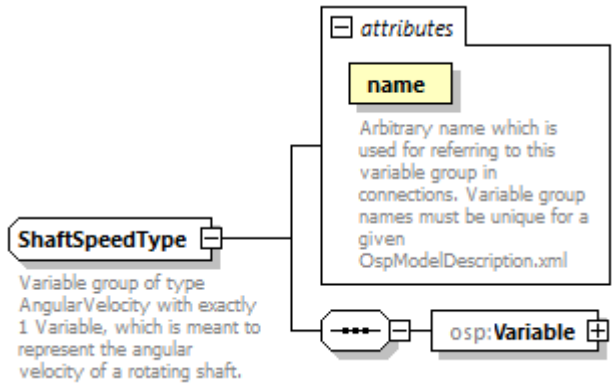
diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **NmeaMwvType/NmeaWindDirection GenericType/NmeaWindDirection VariableGroupsType/NmeaWindDirection**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is

		used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This group contains exactly 2 variables which represent the wind direction part for an NMEA MWV message. The contained variables specify the following: 1. Wind angle in degrees (0 to 359 degrees) 2. Wind direction reference ("R" for "relative" or "T" or "true") The variables must all have the same causality and appear in the order specified above.	

complexType **NmeaWindSpeedType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
children	**osp:Variable**
used by	elements **NmeaMwvType/NmeaWindSpeed GenericType/NmeaWindSpeed VariableGroupsType/NmeaWindSpeed**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation This group contains exactly 2 variables which represents the wind speed part for an NMEA MWV message. The contained variables specify the following: 1. Wind speed 2. Wind speed units (for example, "K" or "M") The variables must all have the same causality and appear in the order specified above.

complexType **ShaftSpeedType**

diagram	
namespace	https://open-simulation-platform.com/OspModelDescription/1.0.0
type	restriction of **osp:AngularVelocityType**
properties	base osp:AngularVelocityType
children	**osp:Variable**
used by	elements **FixedThrusterSetpointType/ShaftSpeed FixedThrusterFeedbackType/ShaftSpeed AzimuthThrusterSetpointType/ShaftSpeed AzimuthThrusterFeedbackType/ShaftSpeed GenericType/ShaftSpeed VariableGroupsType/ShaftSpeed**
attributes	Name: name; Type: **xs:string**; Use: required; Default: ; Fixed: ; Annotation: documentation: Arbitrary name which is used for referring to this variable group in connections. Variable group names must be unique for a given OspModelDescription.xml
annotation	documentation: Variable group of type AngularVelocity with exactly 1 Variable, which is meant to represent the angular velocity of a rotating shaft.

XML Schema documentation generated by **XMLSpy** Schema Editor **http://www.altova.com/xmlspy**

5.5 Status, Challenges, and Outlook

Interfacing with hardware and control systems is often cumbersome due to the large number of non-standard protocols and a wide range of different types of equipment in use. Our aim here is to create a "library" of precisely defined connections for fast and easy use and reuse. The variable groups introduced in Section 5.4 constitute a first best effort attempt for the case of DP controllers. They constitute unambiguous and machine-readable interfaces which make it easier to connect things and to save time and money.

Our hope is that this makes it easier to work with clients and partners on development and testing of control systems and hardware. FMI is too static for this purpose. For example, it currently lacks support for arrays[1] and only a fixed number of variables can be pre-defined for any given simulator interface.

The work presented in this section is, then, a first draft which needs to be extended and built upon in the future. It is challenging to find definitions which are general, consistent, and complete. Because of this, we need you.

It's important that you actively contribute to the further development of controller and hardware interfaces. Please fill in the form in Appendix B and contribute with suggestions for new variable

[1] While this is the situation today, it seems FMI 3.0 will include support for arrays.

group definitions that will make your work faster and easier, and send the form to contribute-ospis@opensimulationplatform.com.

5.5.1 A Note on Hardware-in-the-Loop Testing With Hardwired Signals

When it comes to Hardware-in-the-Loop testing where the control system has hardwired IOs, our efforts as presented in this section unfortunately do not seem to add a lot (yet). This is mainly because the simulator needs to be equipped with a remote IO unit. Then, each case requires starting almost from scratch and network FMUs would need to be re-built for each configuration. This is impractical and allows for little re-use.

What this calls for then is further work and research. And for that, we need you. If you believe you can help us make interfacing with simulators for Hardware-in-the-Loop testing in such cases easier, please get in touch with us using contribute-ospis@opensimulationplatform.com.

6 REFERENCES

[1] Modelica Association, "Official web site of the Function Mock-Up Interface," 2012. [Online]. Available: https://fmi-standard.org. [Accessed 31 1 2019].

[2] T. Preston-Werner, "Semantic Versioning 2.0.0," [Online]. Available: https://semver.org/.

[3] S. Sadjina, L. T. Kyllingstad, M. Rindarøy, S. Skjong, V. Æsøy and E. Pedersen, "Distributed Co-Simulation of Maritime Systems and Operations," *Journal of Offshore Mechanics and Arctic Engineering,* vol. 141, no. 1, p. 011302, 2019.

[4] B. S. M. OWL Working Group (Photographer). / OWL 2 Web Ontology Language Document Overview : W3C Recommendation 27 October 2009. [Website, "OWL 2 Web Ontology Language Document Overview: W3C Recommendation 27 October 2009," 2009.

[5] "Prostep ivip," [Online]. Available: https://www.prostep.org. [Accessed 12 2018].

[6] R. Kübler and W. Schiehlen, "Two methods of simulator coupling," *Mathematical and Computer Modelling of Dynamical Systems,* vol. 6, no. 2, p. 93–113, 2000.

[7] S. Sadjina, L. T. Kyllingstad, S. Skjong and E. Pedersen, "Energy conservation and power bonds in co-simulations: non-iterative adaptive step size control and error estimation," *Engineering with Computers,* pp. 607-620, 2017.

Appendix A TUTORIALS

In this Appendix we'll go over how to actually use the specifications introduced in this document. Together with you, we'll briefly review today's typical situation when it comes to having to connect different sub-simulator models. We'll use a very simple toy system for that purpose. Next, we will show you how OSP-IS leads to correct model connections almost magically. We will then explain how to implement the standard easily along with the exact steps necessary. We'll also provide an example of how OSP-IS makes digital collaboration much easier. And finally, we'll give you pointers with what to do next.

Are you a **simulation tool developer**? Then go right to A.4 and learn how OSP-IS helps you to get rid of having to deal with model interfacing issues once and for all and how it can empower you to create great simulation tools.

A.1 HOW TO MAGICALLY SOLVE INTERFACING ISSUES

Imagine you wanted to build and run a simulation out of some individual models and then study the results. We'll use a very simple toy system for this purpose. It consists of a mass, a damper, and a spring, see Figure 2. For simplicity, we'll only consider two spatial dimensions. We also choose somewhat obscure input and output variable names to help us make our point. Let's further imagine that you wanted to study and plot the mechanical power transmitted between the damper and the mass subsystems.

To best illustrate how OSP-IS can help you with model connections and the interpretations of simulation results, we'll first have a look at the situation as it will typically present itself to you today.

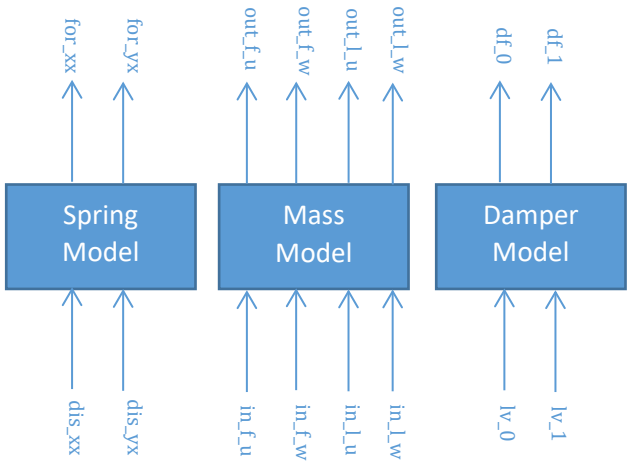


Figure 2: Toy system made up of three individual subsystems: mass, spring, and damper. Each comes with their own interface with somewhat obscure variable names.

A.1.1 THE AGONY OF CONNECTING MODELS

The unconnected system as it appears in Figure 2 brings about an entire host of issues:

- The model interfaces are very unintuitive with names that make it very hard to understand what the signals represent, see Table 1.

- Because of this, it is very difficult to know how to couple the signals to set up the system correctly.
- This is especially unfortunate if you don't enjoy this type of work and simply want to get started simulating and looking at the simulation results.
- In fact, it also makes the interpretation of results and visualizations much more difficult than it should be.

And yes, we absolutely made this simple toy example extra confusing for you. If our personal experiences with these types of situations are anything to go by, it can easily get much more confusing yet.[2]

Table 1: Overview of the toy system's model interfaces.

Model	Inputs	Outputs
Mass	in_f_u, in_f_w, in_l_u, in_l_w	out_f_u, out_f_w, out_l_u, out_l_w
Spring	dis_xx, dis_yx	for_xx, for_yx
Damper	lv_0, lv_1	df_0, df_1

So, what can you typically do if you find yourself in these kinds of situations?

1. You can try to consult the individual model documentations. Often, these can be hard to read and understand or may not even exist at all.
2. You can read the fmiModelDescription.xml file that comes with each model that uses FMI. This will typically give some indications but may not clear up all of your confusion and uncertainty.
3. You can also run some trial simulations and simply plot some or all of the variables to try to make sense of what's going on. Needless to say, this is far from simple either and really a last resort measure for most people.
4. Finally, you can bite the bullet and ask the model developers for help.

Let's assume you made it through all of these steps of varying annoyance. You now have a (hopefully) correctly connected simulator.

[2] We heard of some especially frustrating cases around Hardware-in-the-Loop testing with maritime control systems. These can quickly have 1000s of variable tags which, in lieu of any naming convention, can cause an entire group a full work week just to figure out and validate the signal mappings. Worse yet: Because each case is different, this needs to be repeated every time.

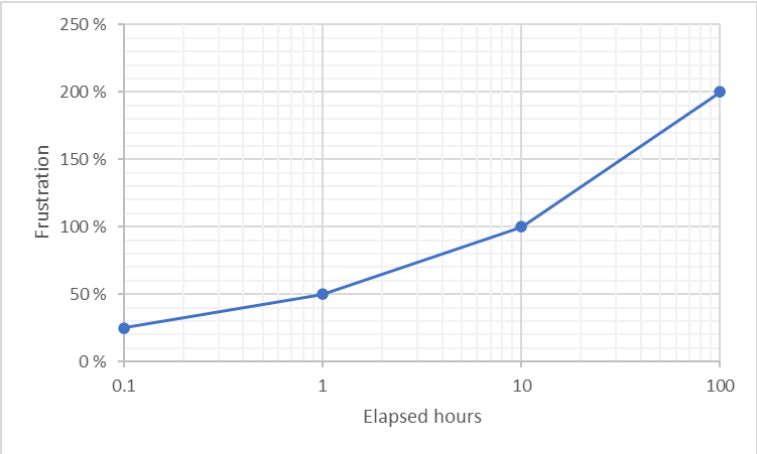


Figure 3: Your approximate levels of frustration while you try to figure out model interfaces.

As discussed, let us now try to plot the mechanical power transmitted between the damper and the mass subsystems. To do this, you need to calculate the dot product between (df_0, df_1) and (out_f_w, out_f_u).[3] Or was it the dot product between (df_0, df_1) and (out_f_u, out_f_w)? Or maybe the dot product between (df_0, df_1) and (out_l_w, out_l_u)?

Frustrating, isn't it? Let's go ahead and see how this can all become much easier with OSP-IS!

A.1.2 Saving Your Time and Your Nerves with OSP-IS

Let's move on from all these frustrations to a world where model connections become almost magically easy. Table 2 shows the available connections after OSP-IS has been implemented (and we'll show you how to do just that shortly). Can you guess how to correctly connect the spring and damper sub-models to the mass?

Table 2: Overview of the toy system's model interfaces using OSP-IS.

Model	Ports
Mass	<LinearMechanicalPort>, <LinearMechanicalQuasiPort>
Spring	<LinearMechanicalQuasiPort>
Damper	<LinearMechanicalPort>

[3] Note that this is not the only way to calculate a power in this case. For more information on powers and energies in co-simulations, the ambiguities surrounding this topic, and how it all relates to co-simulation errors, see [6].

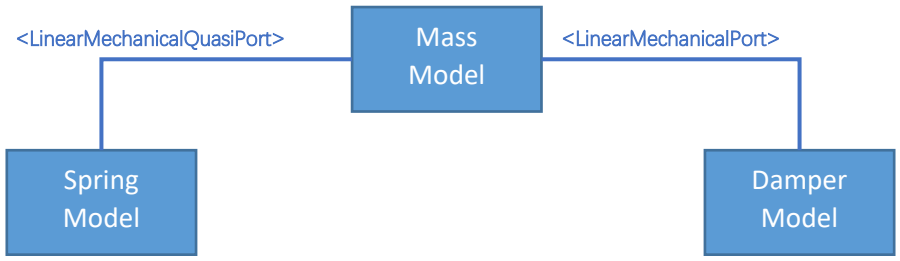


Figure 4: Toy system made up of three individual subsystems: mass, spring, and damper. Each comes with their own OSP-IS-compatible interface which makes connecting them much easier and less prone to errors.

Now, there is only exactly one single way to connect all three sub-models, see Figure 4! OSP-IS wraps all models into a different and much more intuitive interface.[4] Instead of dealing with individual variables with obscure names, you now get to work on *variable groups* with much higher interpretability.

The mass and the damper sub-models both have a <LinearMechanicalPort> variable group. This type of connector represents a force—velocity pair and coupling via mechanical power. The mass and the spring both have a <LinearMechanicalQuasiPort> variable group which represents a force—position pair.

So, what does your process look like now with the power of OSP-IS in place?

1. You connect all models easily and confidently.
2. You run your simulation knowing that the interface is correct.
3. You look at the results.
4. *Optional*: You can call it an early day and go have a beer or beverage of your liking to celebrate.

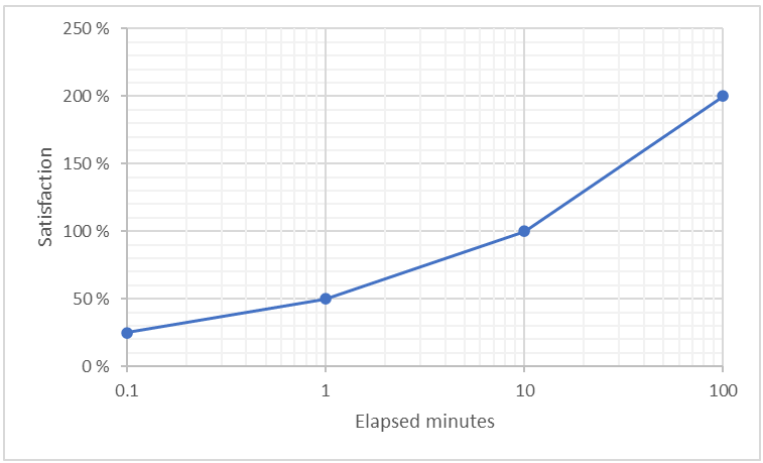


Figure 5: Your approximate levels of satisfaction while you enjoy the process of connecting models using OSP-IS.

Doesn't this seem much more enjoyable to you? It gets better! Let's turn to plotting the mechanical power transmitted between the damper and the mass subsystems: Simply calculate the dot product

[4] Of course, this won't keep people from using obscure variable names for their model interfaces. But now you can handle them much easier!

between the <Force> component of <LinearMechanicalPort> and the <LinearVelocity> component of <LinearMechanicalPort>. Easy, right?

Ok great! So how do you end your model connection headaches once and for all and get to the promised OSP-IS land where it all becomes magically easy? We'll now show you how!

A.1.3 Your Effortless Transition to OSP-IS

There are essentially two steps to implementing OSP-IS:

1. The model creator defines the model interface by constructing an OspModelDescription.xml file.
2. You set up an OspSystemStructure.xml file which specifies the connections between the models in your simulation.

In-depth information on both can be found in Section 3.1 and https://open-simulation-platform.github.io/libcosim/configuration respectively.

A.1.4 Defining the Model Interfaces

To define a model interface, the model creator declares the variable group structure of their model's FMU variables.

Consider the damper model first. All we have to do is wrap the FMU variables (with their obscure names) into a <LinearMechanicalPort> variable group like so:

```
<?xml version="1.0" encoding="utf-8" ?>
<OspModelDescription xmlns="https://open-simulation-platform.com/OspModelDescription/1.0.0"
version="1.0">

  <VariableGroups>

    <LinearMechanicalPort name="damper_linear_mechanical_port">

      <Force name="damper_force">
        <Variable ref="df_0"/>
        <Variable ref="df_1"/>
      </Force>

      <LinearVelocity name="damper_velocity">
        <Variable ref="lv_0"/>
        <Variable ref="lv_1"/>
      </LinearVelocity>

    </LinearMechanicalPort>

  </VariableGroups>

</OspModelDescription>
```

We then define a <LinearMechanicalQuasiPort> variable group for our spring model:

```
<?xml version="1.0" encoding="utf-8" ?>
<OspModelDescription xmlns="https://open-simulation-platform.com/OspModelDescription/1.0.0"
version="1.0">

  <VariableGroups>

    <LinearMechanicalPort name="spring_linear_mechanical_quasi_port">

      <Force name="spring_force">
        <Variable ref="for_xx"/>
        <Variable ref="for_yx"/>
      </Force>

      <LinearDisplacement name="spring_position">
        <Variable ref="dis_xx"/>
        <Variable ref="dis_yx"/>
      </LinearDisplacement>

    </LinearMechanicalPort>

  </VariableGroups>

</OspModelDescription>
```

Moving on to the mass model, we define the <LinearMechanicalPort> and <LinearMechanicalQuasiPort> variable groups based on the FMU variables:

```
<?xml version="1.0" encoding="utf-8" ?>
<OspModelDescription xmlns="https://open-simulation-platform.com/OspModelDescription/1.0.0"
version="1.0">

  <VariableGroups>

    <LinearMechanicalPort name="mass_linear_mechanical_port">

      <Force name="mass_force_1">
        <Variable ref="in_f_w"/>
        <Variable ref="in_f_u"/>
      </Force>

      <LinearVelocity name="mass_velocity">
        <Variable ref="out_f_w"/>
        <Variable ref="out_f_u"/>
      </LinearVelocity>

    </LinearMechanicalPort>

    <LinearMechanicalQuasiPort name="mass_linear_mechanical_quasi_port">

      <Force name="mass_force_2">
        <Variable ref="in_l_w"/>
        <Variable ref="in_l_u"/>
      </Force>

      <LinearDisplacement name="mass_position">
        <Variable ref="out_l_w"/>
```

```
        <Variable ref="out_l_u"/>
      </LinearDisplacement>

    </LinearMechanicalQuasiPort>

  </VariableGroups>

</OspModelDescription>
```

Once this is done, the osp-validator can (and should) be used to check if all <VariableGroups> in an OspModelDescription.xml file are defined correctly.

The validator can be downloaded at https://github.com/open-simulation-platform/osp-validator/releases.

A.1.5 Defining the Model Connections

With all model interfaces in place, we can move on to connect them for your simulation. Using interfaces adhering OSP-IS, this becomes a real breeze. Have a look:

```
<?xml version="1.0" encoding="utf-8" ?>
<OspSystemStructure xmlns="https://open-simulation-platform.com/OspModelDescription/1.0.0"
version="1.0">

   <!-- Set simulation parameters -->
   <StartTime>0.0</StartTime>
   <BaseStepSize>0.05</BaseStepSize>
   <Algorithm>fixedStep</Algorithm>

   <Simulators>

      <!-- Collect the model FMUs -->
      <Simulator name="mass"     source="mass.fmu"/>
      <Simulator name="spring"   source="spring.fmu"/>
      <Simulator name="damper"   source="damper.fmu"/>

   </Simulators>

   <Connections>

      <VariableGroupConnection>
    <!-- Set up the connection between mass and damper FMUs -->
         <VariableGroup simulator="mass"      name="mass_linear_mechanical_port"/>
         <VariableGroup simulator="damper"    name="damper_linear_mechanical_port"/>
      </VariableGroupConnection>

      <VariableGroupConnection>
    <!-- Set up the connection between mass and spring FMUs -->
         <VariableGroup simulator="mass"      name="mass_linear_mechanical_quasi_port"/>
         <VariableGroup simulator="spring"    name="spring_linear_mechanical_quasi_port"/>
      </VariableGroupConnection>

   </Connections>

</OspSystemStructure>
```

Again, the osp-validator can (and should) be used to check the validity of the connections in OspSystemStructure.xml. This gives you added certainty and confidence that things are truly moving in the right direction.[5]

To run your simulation, you will need a tool which supports the use of OspSystemStructure.xml. At the time of writing, only libcosim does so. It is fully open source, and we encourage you to check it out here https://open-simulation-platform.github.io/libcosim !

A.2 How OSP-IS Makes Digital Collaboration Easier

We have now taken you on a tour to demonstrate how OSP-IS can almost magically solve your model interfacing issues. The vision that drives us is to make digital collaboration easier within and across organizations and research groups. At the highest level of abstraction this calls for a common language. We hope that OSP-IS takes us all a big step further in the right direction by providing a common language for model interfacing[6].

To further illustrate this point, consider our toy system with its three simple models one last time. Imagine you are in possession of the spring and damper models and know their interfaces. But you still need that mass model to set up your simulation and study the system's behaviour. This means that you rely on *AwesomeMassModelSystems Ltd. Corp.* to give you their intricate and refined mass model. But because *AwesomeMassModelSystems Ltd. Corp.* supports OSP-IS, you can simply request a mass model with one <LinearMechanicalPort> and one <LinearMechanicalQuasiPort>.



Figure 6: How digital collaboration should feel like!

While you wait for the mass model, you can already set up an OspSystemStructure.xml file as shown in Section A.1.3. As soon as *AwesomeMassModelSystems Ltd. Corp.* delivers the requested mass model (along with an OspModelDescription.xml file similar to the one shown in Section A.1.3) you can immediately and easily connect it and get cracking with your simulation right away!

[5] The validator will not be able to catch all types of errors but check whether the standard is followed as intended.

[6] Another effort is documented in the "Towards a Maritime Domain Ontology" report, available at https://opensimulationplatform.com/

A.3 What's Next?

We took you all the way from the current state of model interfacing issues to how it all becomes much easier and smoother with OSP-IS. We also tried to draw a vivid picture of the future when we can all collaborate much more efficiently to create better models and better simulations.

But there is still much left to do to make the simulation world a better and smarter one and to raise digital collaboration to an entirely new level. And we can't do it ourselves. **We need you!**

If you want to dive deeper into the topics presented in this document, head on over to https://opensimulationplatform.com/ right away!

If you want to help us to make model interfacing easier and to empower everyone from model developers, system designers, commissioning engineers, to simulation tool developers, then get involved now by dropping us an e-mail at contribute-ospis@opensimulationplatform.com. Any ideas, comments, or criticism is warmly welcomed!

A.4 A Message to the Simulation Tool Developer

> OSP-IS helps you to get rid of having to deal with model interfacing issues once and for all and empower you to create more awesome simulation tools.

Imagine you wanted to create a workbench for that awesome new simulation tool that you and your team have been working on. One major pain you will likely have encountered by now is how to interpret the myriad of mysterious ways in which model developers create the interface variables and what names they use. The situation may look something like shown in Figure 7.

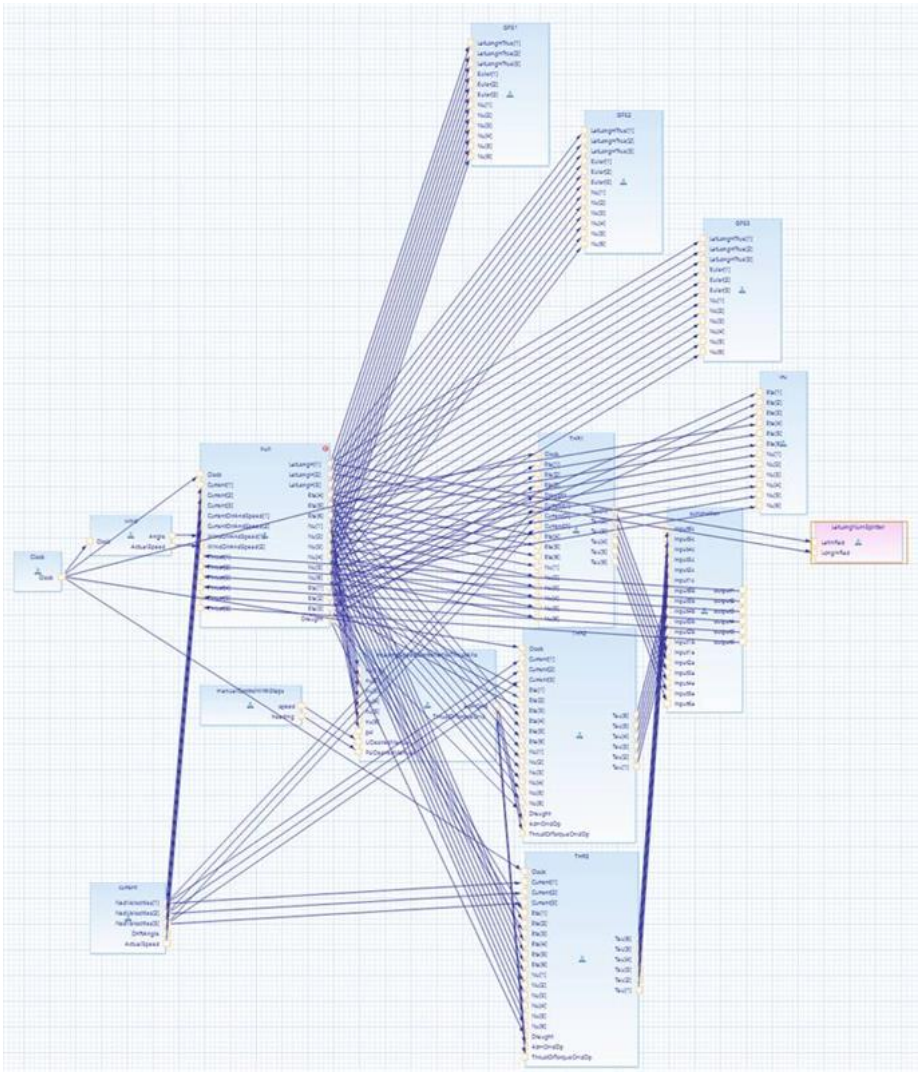


Figure 7: What the model connection problem may look like for you now.

Doesn't look like fun, does it? Imagine there was a way to make the visualization of model connections much easier and more systematic? Have a look at Figure 8 to see what that could look like when using OSP-IS as an interfacing standard.

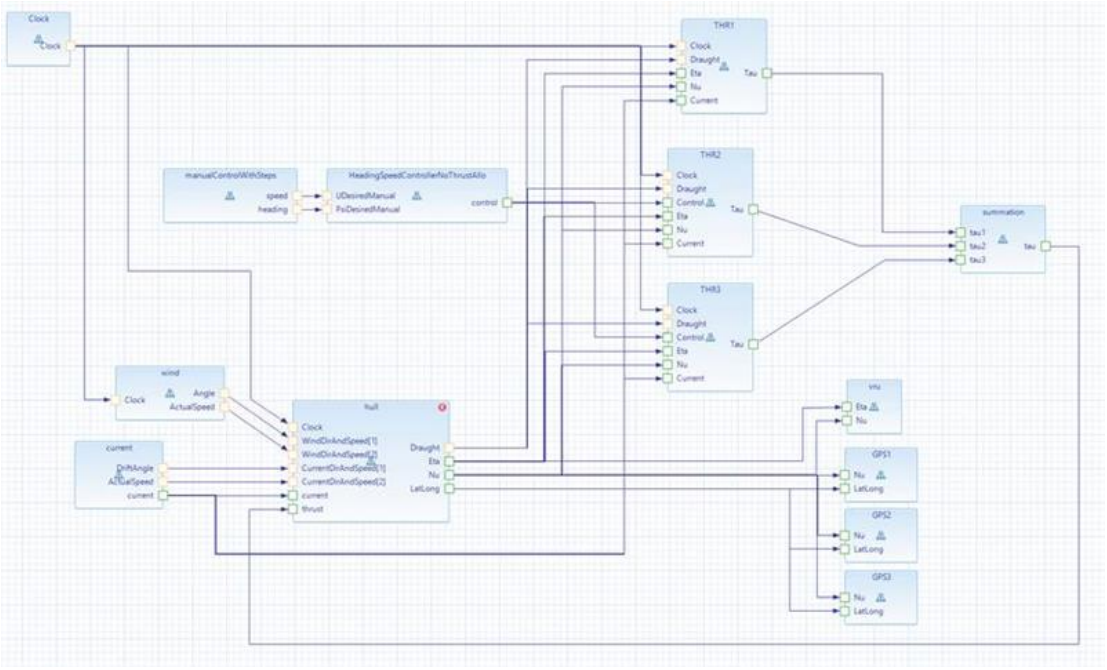


Figure 8: Illustration for what model connections look like using OSP-IS.

If we just piqued your curiosity and you want to know a bit more, we advise you to do the following:

1. Have a look at our Model Developer's Tutorial in Section A.1. It will also show you how OSP-IS shifts the responsibility for model interfacing over to model developers.
2. Head on over to https://open-simulation-platform.github.io/libcosim to check out libcosim. It is a simulation tool demo with OSP-IS implemented.
3. Check out the osp-validator to learn how to automatically validate model connections and interfaces, at https://open-simulation-platform.github.io/osp-validator
4. Go to https://opensimulationplatform.com/ and have a look at our use cases and how they use OSP-IS to make model interfacing magically simple.

If you want to help us to make model interfacing easier and to empower everyone from model developers, system designers, commissioning engineers, to simulation tool developers, then get involved now by dropping us an e-mail at contribute-ospis@opensimulationplatform.com. Any ideas, comments, or criticism is warmly welcomed!

Appendix B Form for New Variable Groups

Variable group name:

Description:

Contained variables

Datatype	Causality	Unit	Restrictions	Description

Restrictions on contained variables

Contained variable groups

Name	Restrictions

Restrictions on contained variable groups

Other restrictions

B.1 Examples

Variable group name: Force

Description: This variable group is meant to represent a force. It can contain 1 to 3 variables and all must be of the same causality.

Contained variables

Datatype	Causality	Unit	Restrictions	Description
Real	input/output			First force variable
Real	input/output			Second force variable
Real	input/output			Third force variable

Restrictions on contained variables

All variables must have Real data type
All variables must have the same causality, either input or output
The variable group must have at least 1 variable and at most 3 variables

Contained variable groups

Name	Restrictions

Restrictions on contained variable groups

Other restrictions

Variable group name: LinearVelocity

Description: This variable group is meant to represent a linear velocity. It can contain 1 to 3 variables and all must be of the same causality.

Contained variables

Datatype	Causality	Unit	Restrictions	Description
Real	input/output			First linear velocity variable
Real	input/output			Second linear velocity variable
Real	input/output			Third linear velocity variable

Restrictions on contained variables

All variable must have Real data type
All variable must have the same causality either input or output
The variable group must have at least 1 variable and at most 3 variables

Contained variable groups

Name	Restrictions

Restrictions on contained variable groups

Other restrictions

Variable group name: LinearMechanicalPort

Description: This variable group is meant to represent a Force/LinearVelocity pair called a LinearMechanicalPort, inspired by bond graph theory.

Contained variables

Datatype	Causality	Unit	Restrictions	Description

Restrictions on contained variables

This variable group can not contain any variables directly

Contained variable groups

Name	Restrictions
Force	This variable group must appear exactly once
LinearVelocity	This variable group must appear exactly once

Restrictions on contained variable groups

The Force- and LinearVelocity variable group must contain the same number of variables
The variables in the Force variable group must have the opposite causality of the variables in the LinearVelocity variable group

Other restrictions

Variable group name: NmeaWindDirection

Description: This group contains exactly 2 variables which represent the wind direction part for an NMEA MWV message.

Contained variables

Datatype	Causality	Unit	Restrictions	Description
Real	input/output	degrees		Wind angle in degrees (0 to 359 degrees)
String	input/output			Wind direction reference ("R" for "relative" or "T" for "true")

Restrictions on contained variables

The variables must appear in the order specified in the table above
Both variables must have the same causality in any given variable group; either input or output

Contained variable groups

Name	Restrictions

Restrictions on contained variable groups

Other restrictions
