OpenMI

The OpenMI Document Series

Part D - org.OpenMI.Backbone technical documentation

For OpenMI (Version 1.4)

Title	OpenMI Document Series: Part D - org.OpenMI.Backbone technical documentation for the OpenMI (version 1.4)
Editor	Peter Gijsbers, WL | Delft Hydraulics, Delft, The Netherlands
Authors	Stefan Westen, HR Wallingford Group, Wallingford, UK
Document production	Peter Gijsbers, WL | Delft Hydraulics, Delft, The Netherlands
Current version	V1.4
Date	21/05/2007
Status	Final © The OpenMI Association
Copyright	All methodologies, ideas and proposals in this document are the copyright of the OpenMI Association. These methodologies, ideas and proposals may not be used to change or improve the specification of any project to which this document relates, to modify an existing project or to initiate a new project, without first obtaining written approval from the OpenMI Association who own the particular methodologies, ideas and proposals involved.
Acknowledgement	This document has been produced as part of the OpenMI-Life project. The OpenMI-Life project is supported by the european Commission under the Life Programme and contributing to the implementation of the thematic component LIFE-Environment under the policy area "Sustainable management of ground water and surface water managment" Contract no : LIFE06 ENV/UK/000409. The first version of this document has been produced as part of the HarmonIT project; a research project supported by the European Commission under the Fifth Framework Programme and contributing to the implementation of the Key Action “Sustainable Management and Quality of Water” within the Energy, Environment and Sustainable Development. Contract no: EVK1-CT-2001-00090.

Preface

OpenMI stands for Open Modeling Interface and aims to deliver a standardized way of linking of environmental related models. This document describes the default implementation of the standardized OpenMI interfaces as being utilized in the OpenMI Software Development Kit. It is the fourth document in the OpenMI report series, which specifies the OpenMI interface standard, provides guidelines on its use and describes software facilities for migrating, setting up and running linked models.

Other titles in the series include:

A. Scope

B. Guidelines

C. org.OpenMI.Standard interface specification

D. org.OpenMI.Backbone technical documentation (this document)

E. org.OpenMI Development Support technical documentation

F. org.OpenMI.Utilities technical documentation

The interface specification is intended primarily for developers. For a more general overview of the OpenMI, see Part A (Scope).

The official reference to this document is:

OpenMI Association (2007) *The org.OpenMI.Backbone technical documentation*. Part D of the OpenMI Document Series

Disclaimer

The information in this document is made available on the condition that the user accepts responsibility for checking that it is correct and that it is fit for the purpose to which it is applied.

The OpenMI Association will not accept any responsibility for damage arising from actions based upon the information in this document.

Further information

Further information on the OpenMI Association and the Open Modelling Interface can be found on http://www.OpenMI.org.

Table of contents

1 Introduction

1.1 Background

OpenMI stands for Open Modeling Interfaces. It aims to deliver a standardized way to link environmental related computational models that run simultaneously. In summary, OpenMI primarily focuses on providing a complete protocol to explicitly define, describe and transfer (numerical) data between components on a time basis, including associated component access. It thus enables process interaction being represented more accurately, compared to sequential linkages.

The establishment of OpenMI will support and assist the scientific, consultancy and water management community in the integrated assessment of water management systems and thus strategic planning and integrated catchment management required by the European Water Framework Directive.

This standardized way of linking models is achieved by an intelligent protocol to describe, define and transfer data. This protocol is translated into a strict set of rules, i.e. formal interfaces, to be implemented by software code. Any component that implements these interfaces is called an OpenMI compliant component.

Within OpenMI, a distinction is made between the standardized interfaces, incorporated in the org.OpenMI.Standard namespace, and an implementation in other namespaces which provide a so-called Software Development Kit (SDK). The org.OpenMI.Backbone namespace provides the default implementation of most interfaces in a set of classes (see Figure 1). This Backbone is part of the OpenMI Software Development Kit and it just provides a means to work with the OpenMI interfaces. Other namespaces in the OpenMI SDK work with the classes from this backbone implementation. Application of this software layer is not mandatory, but it may save costs and effort to join.

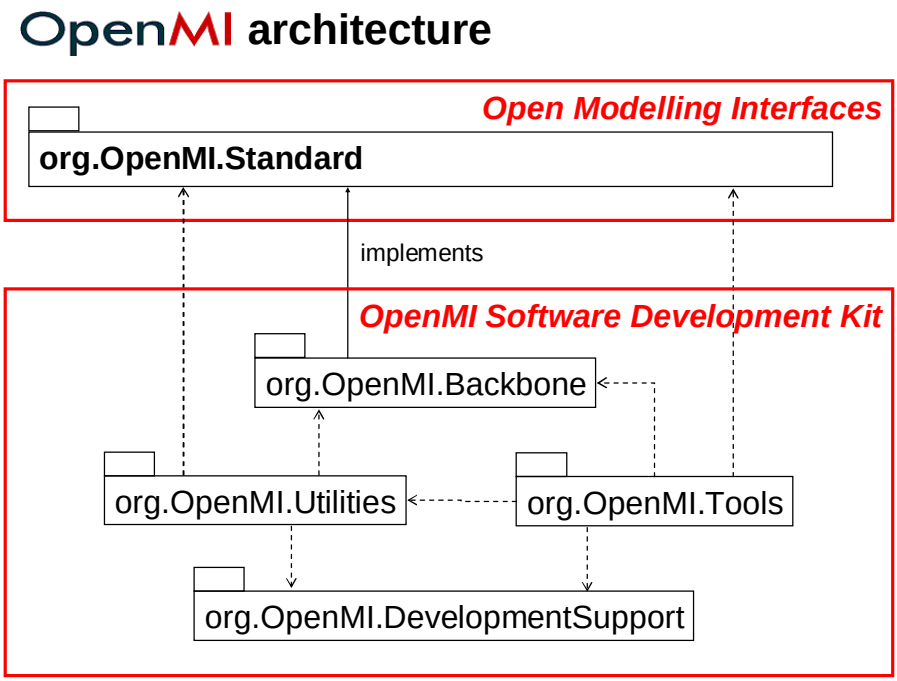

Figure 1 Namespaces in the OpenMI architecture

1.2 Requirements

The org.OpenMI.Backbone namespace provides the default implementation of the OpenMI SDK, without any business logic, of most interfaces from the org.OpenMI.Standard namespace.

The following requirements have been kept in mind.

- [Req-BB1] – provide argument less constructors to instantiate the standardized OpenMI classes based on the OpenMI interface specification
- [Req-BB2] – implement member functions of the OpenMI classes as defined by the interfaces
- [Req-BB3] – where appropriate, provide convenience classes that are formally not part of the interface specification, but are conceptually part of OpenMI.
- [Req-BB4] – where appropriate, implement convenience functions for the OpenMI classes (e.g. 'set' functions for properties)
- [Req-BB5] – provide implementations for the following interfaces:
  IDataOperation, IDimension, IElementSet, IEvent, ILink, ILinkableComponent, iQuantity, IScalarSet, ISpatialReference, ITimeSpan, ITimeStamp, IUnit, IVector, IVectorSet and IListener IPublisher

  The IManageState interface is implemented in the org.OpenMI.Utilities.Wrapper package.

1.3 Scope of this document

This report contains the technical documentation of the org.OpenMI.Backbone namespace. The technical documentation addresses the design as well implementation issues. After discussing the major concepts in chapter 2, the actual implemented classes are discussed in chapter 3.

1.4 Readership

This document is meant for code developers who have to implement, extend or maintain the source code of the OpenMI Software Development Kit. In order to understand this document, one needs to have basic understanding of model linking, object-orientation and UML-notation (particularly class diagrams and sequence diagrams). Within the text, the following style-convention is applied:

- OpenMI interface
- OpenMI method
- OpenMI property
- OpenMI argument

2 OpenMI Backbone: Concepts and scope

2.1 The OpenMI linking mechanism

OpenMI is a pull-based pipe and filter architecture, consisting of communicating components (providers and acceptors), which exchange data in a pre-defined way and in a pre-defined object format. Sometimes, this type of architecture is also referred to as a context based request-reply architecture, in which the context (i.e. the instantiated component) processes and replies to the requests in synchronized order.

At the highest level, OpenMI is defined as a set of interfaces via the org.OpenMI.Standard namespace. These interfaces contain all protocols to define, describe and transfer data. The actual implementation of these interfaces is allocated to other packages.

2.2 Task description of the OpenMI Backbone

The main task of the org.OpenMI.Backbone namespace is to provide a default implementation for the majority of the interfaces as defined in the org.OpenMI.Standard namespace. This default implementation provides two constructors to instantiate of OpenMI classes, either with or without arguments. In addition it provides some convenience functions which are not part of the Standard. E.g. the Standard only prescribes the ‘get’ part of a property, while the Backbone supports the ‘set’ part of such property as well.

The org.OpenMI.Backbone namespace does not contain any business logic to meet the data exchange mechanisms of OpenMI.

2.3 Outline of the org.OpenMI.Backbone namespace

2.3.1 Packages

The org.OpenMI.Backbone namespace consists of one package.

2.3.2 Relations to other namespaces

The org.OpenMI.Backbone package implements the majority of the interfaces as defined in org.OpenMI.Standard namespace. Most packages of the OpenMI SDK approach OpenMI classes through the org.OpenMI.Standard interfaces, while the classes as implemented by this Backbone implementation are used when instantiating and manipulating OpenMI-objects (see Figure 2).

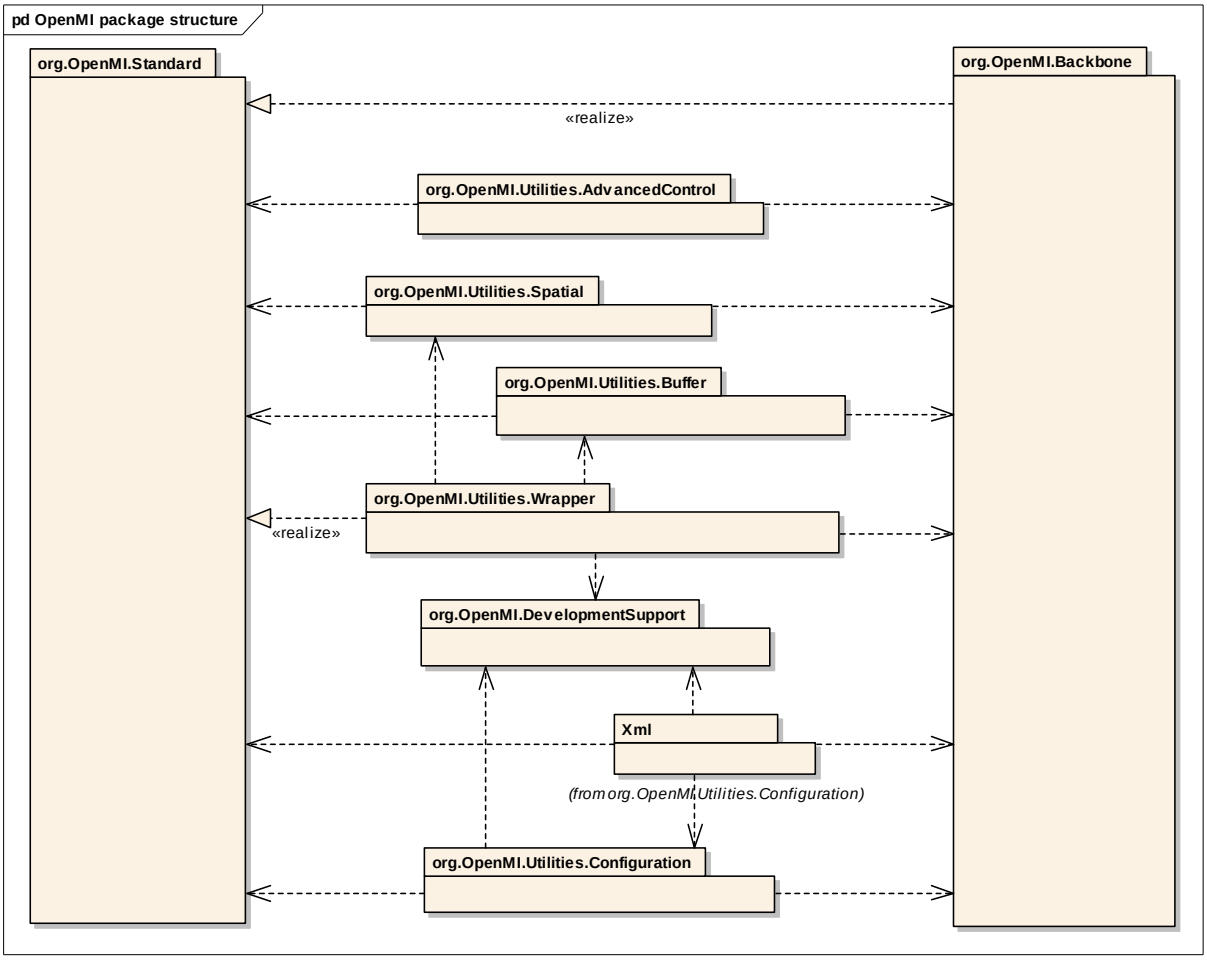

Figure 2 Relations between the org.OpenMI.Standard, the org.OpenMI.Backbone package and other packages within the OpenMI Software Development Kit

3 The org.OpenMI.Backbone package

3.1 General description

As indicated, the org.OpenMI.Backbone package provides the default implementation for the majority of the interfaces of org.OpenMI.Standard. This default implementation provides a variety of constructors for OpenMI objects, including an argument less one. For convenience purposes most classes contain a limited set of additional functions, e.g. to assign property values outside the constructor.

3.2 Static View

3.2.1 Value related classes

Figure 3 illustrates the implementation of the Value related classes. Both the ScalarSet and VectorSet class implement the Count property (get only) themselves (The Count property is inherited from IValueSet). All other properties are get/set.

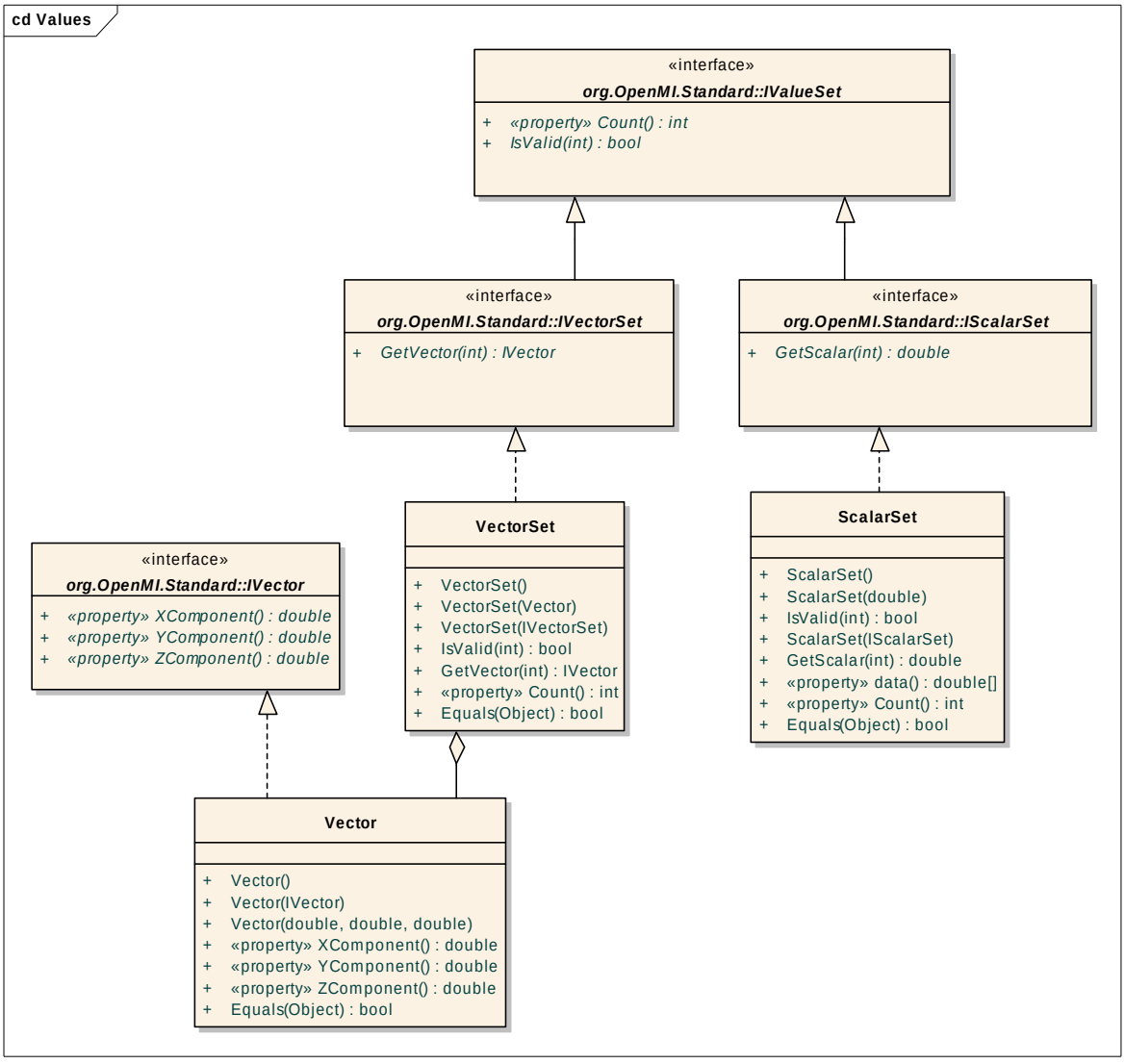

Figure 3 Implementation of the Value related classes

3.2.2 Time related classes

Figure 4 illustrates the implementation of the Time related classes. A number of convenience functions are added to compare two objects. All properties are get/set.

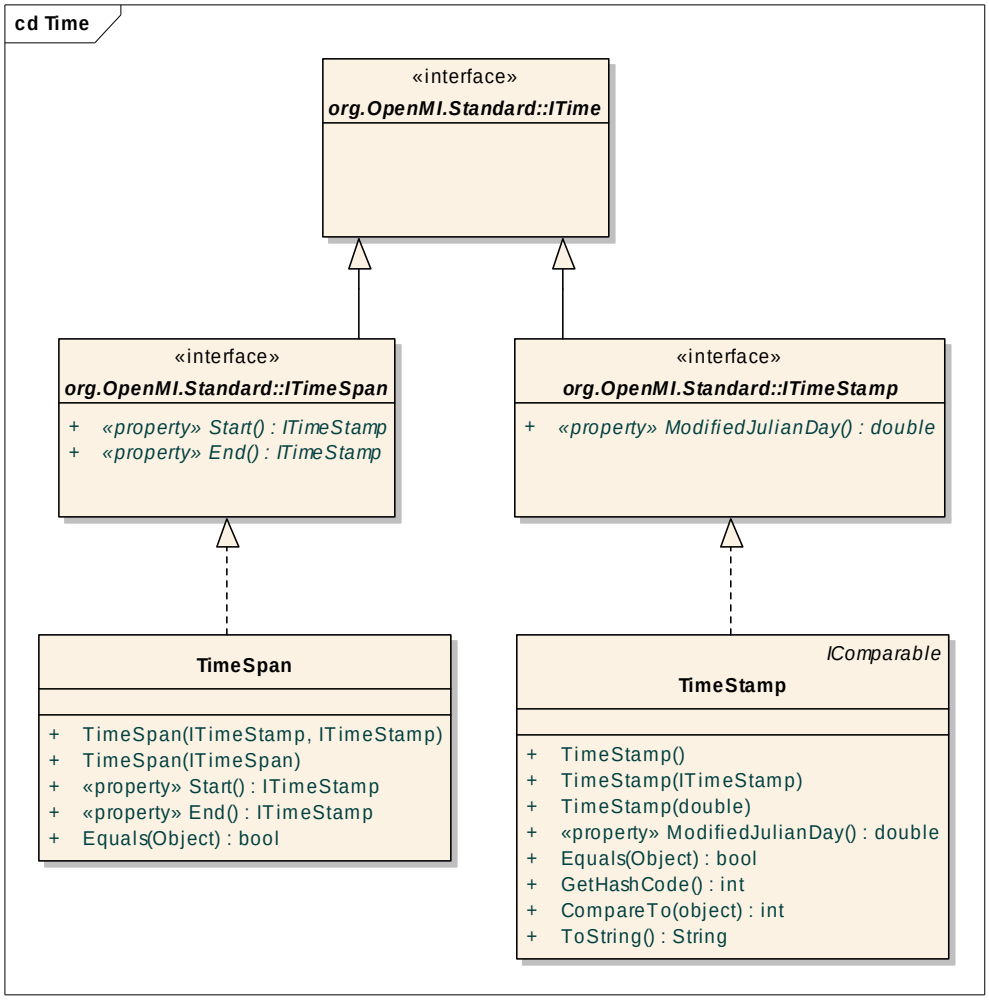

Figure 4 Implementation of the Time related classes

3.2.3 Quantity and related classes

As can be seen in Figure 5, the Quantity class and the Unit class provide a variety of constructors. All properties are get/set.

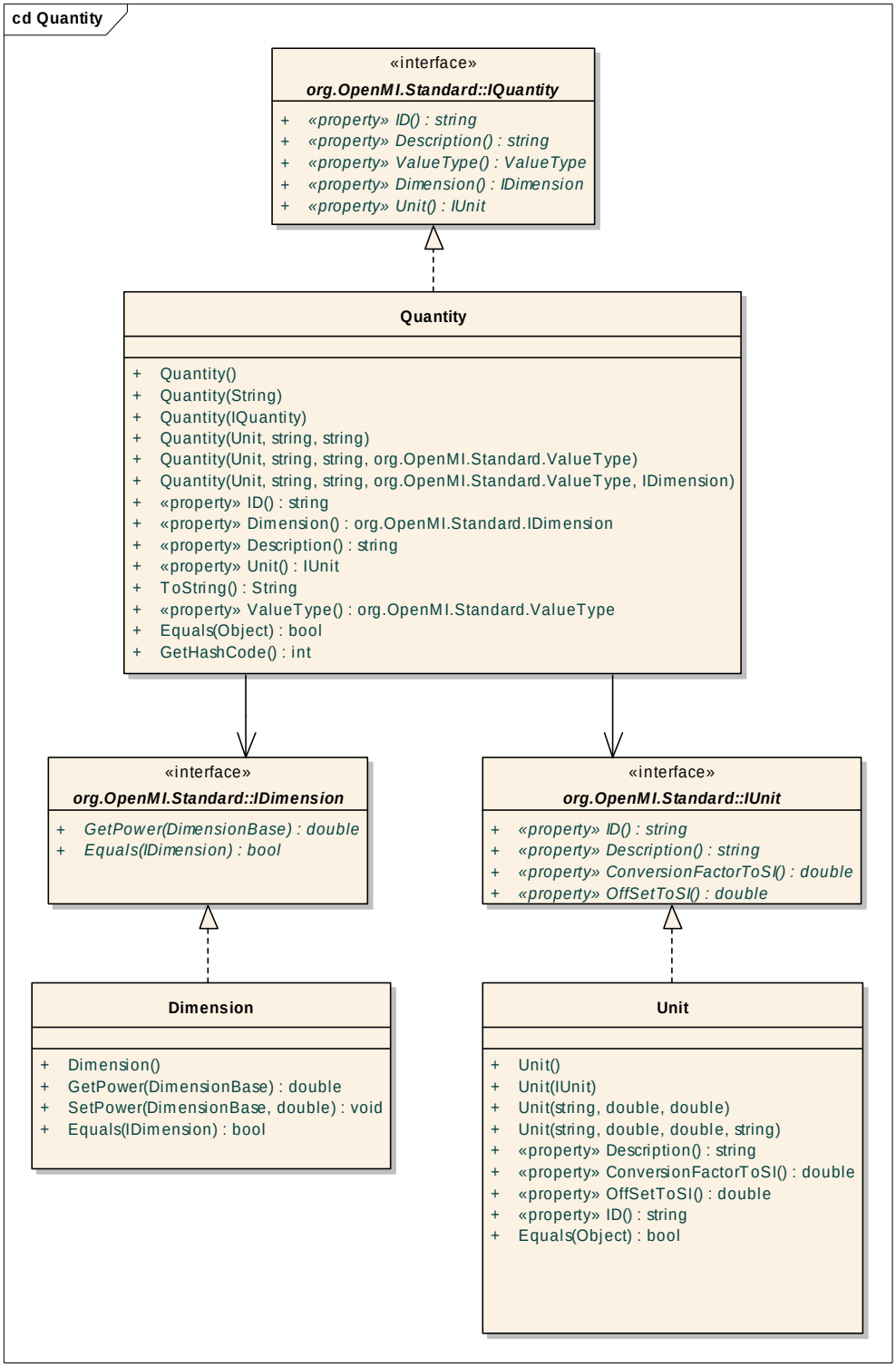

Figure 5 Implementation of the Quantity and related classes

3.2.4 DataOperation related classes

The org.OpenMI.Backbone package provides a variety of constructors for the DataOperation interface and the IArgument interface. The Count property is available as get only, the other properties are available as get/set (see Figure 6).

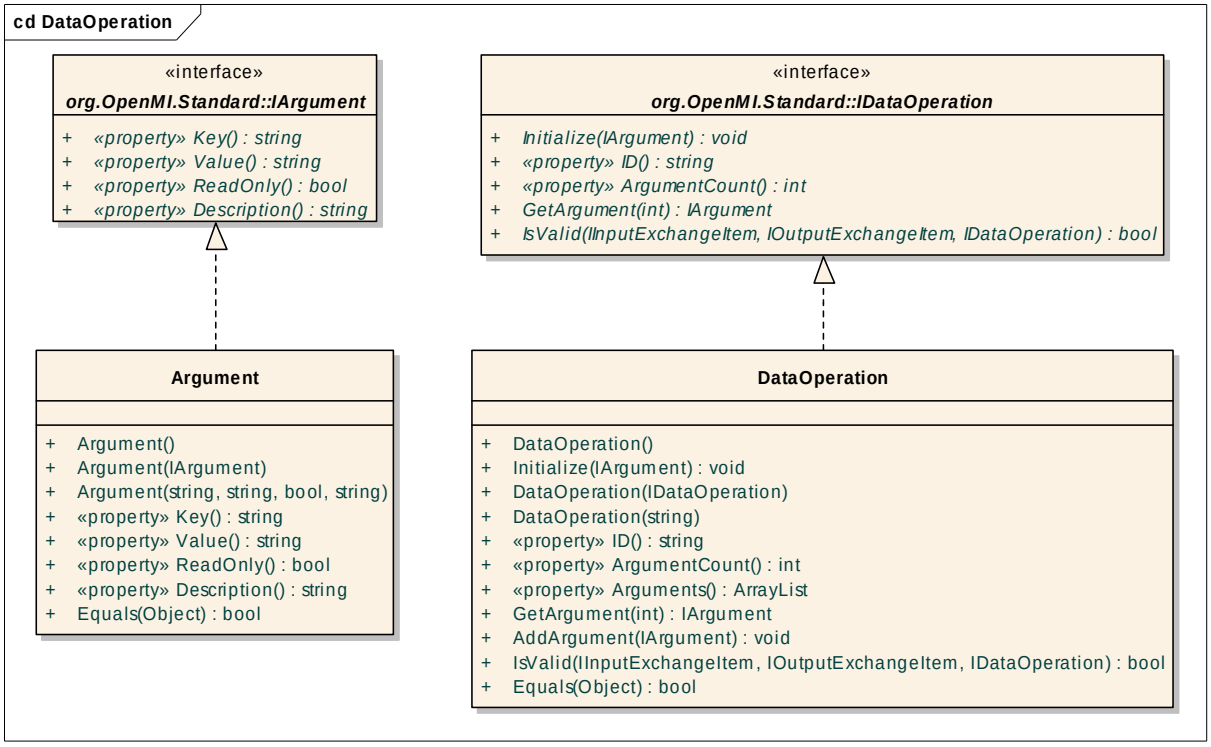

Figure 6 Implementation of the DataOperation and related classes

3.2.5 ElementSet related classes

Conceptually, the element set consists of an array of elements. Elements that are geo-referenced, are described by a list of vertices. While the org.OpenMI.Standard namespace only define two interfaces, namely IElementSet and ISpatialReference, the org.OpenMI.Backbone package provides classes for the Element and Vertex as well to enable internal handling of this (private) information. As can be seen in Figure 7, the ElementSet class contains an array of elements, while the Element class contains an array of vertices.

All properties are get/set, except for ElementCount in the ElementSet-class and VertexCount in the Element-class. The latter only offer a get.

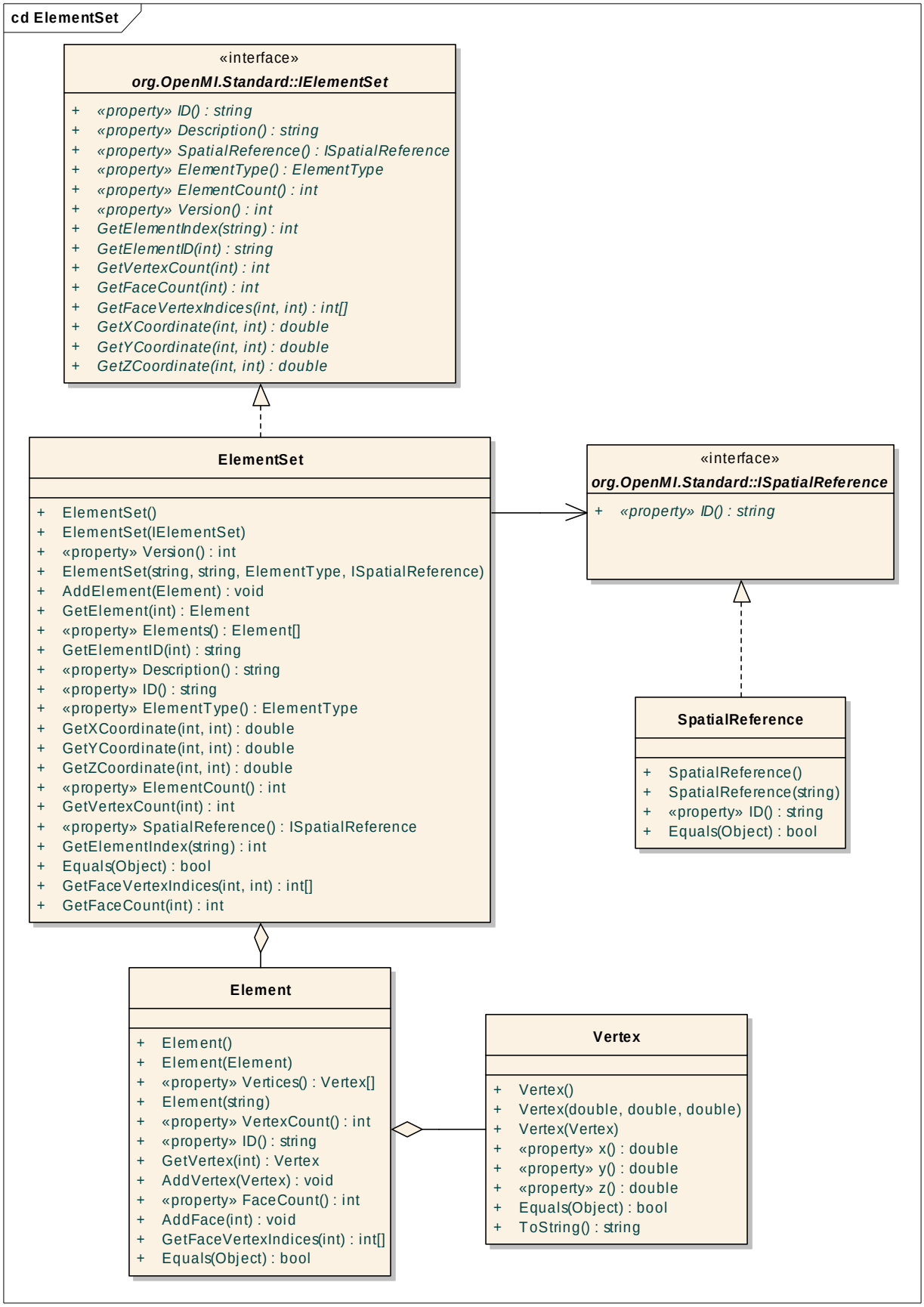

Figure 7 Implementation of the ElementSet and related classes

3.2.6 The Link class

Figure 8 illustrates the implementation of the Link. The arguments in the constructor list the Source information, than the Target information, then the ID and Description and finally an array list of DataOperations. All properties are get/set, except for DataOperationsCount property, which is offered via a get method only.

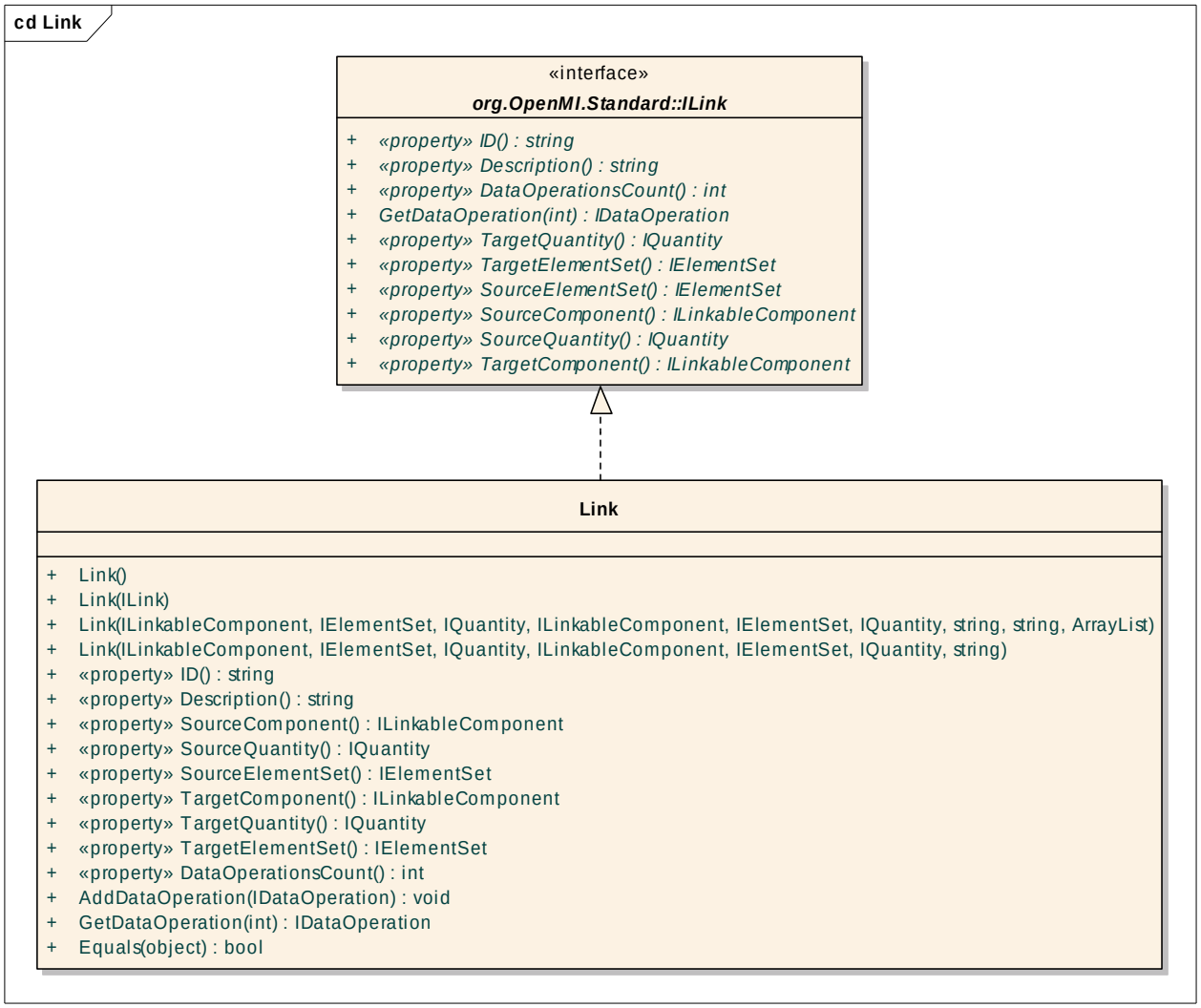

Figure 8 Implementation of the Link

3.2.7 The LinkableComponent class

Figure 9 illustrates the implementation of the LinkableComponent. Note that the IPublisher interface is not available separately. All properties are get/set, except for LinkCount property, which is offered via a get method only.

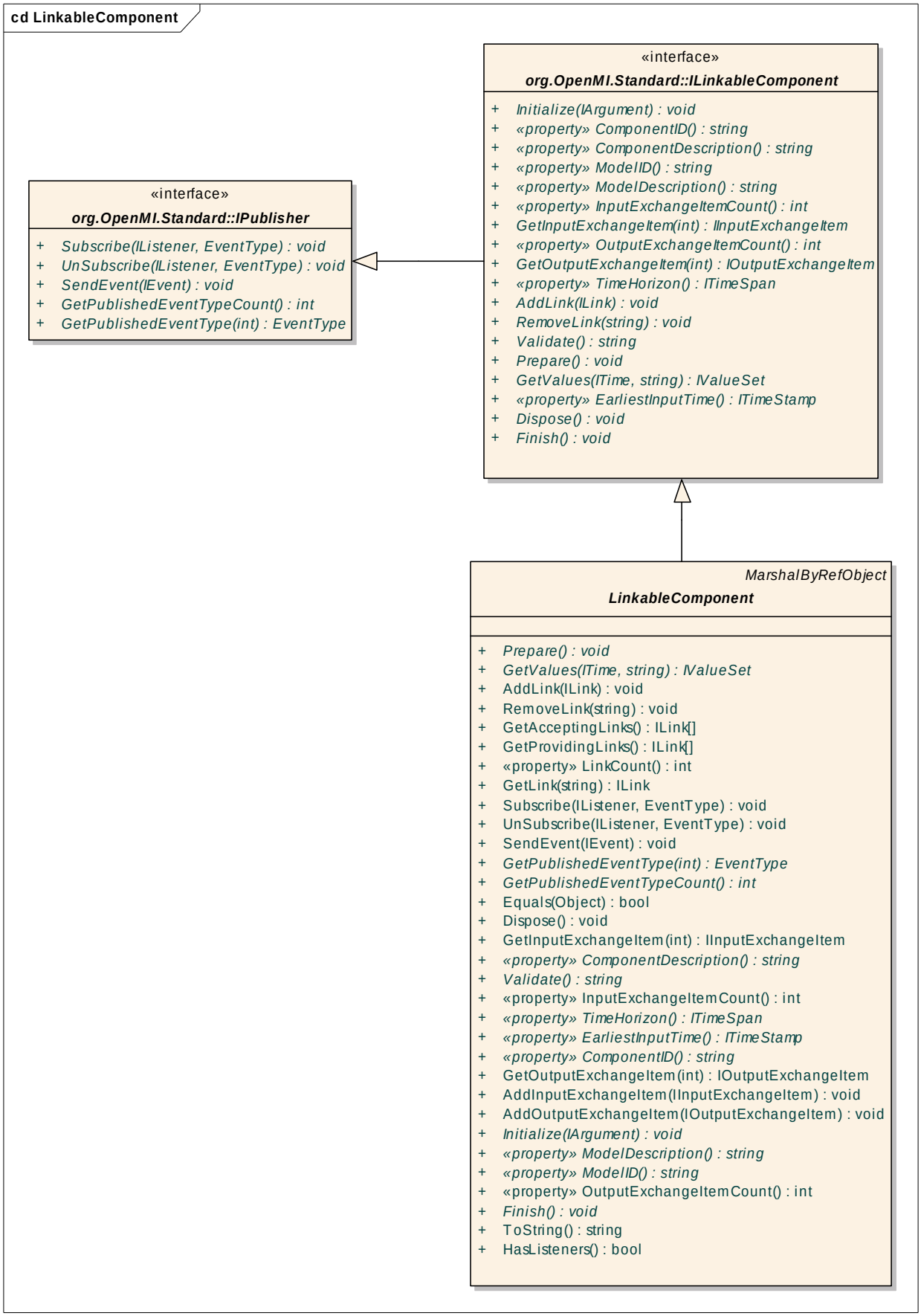

Figure 9 Implementation of the LinkableComponent

3.2.8 Events and exceptions

Figure 10 illustrates the current implementation of the Event class.

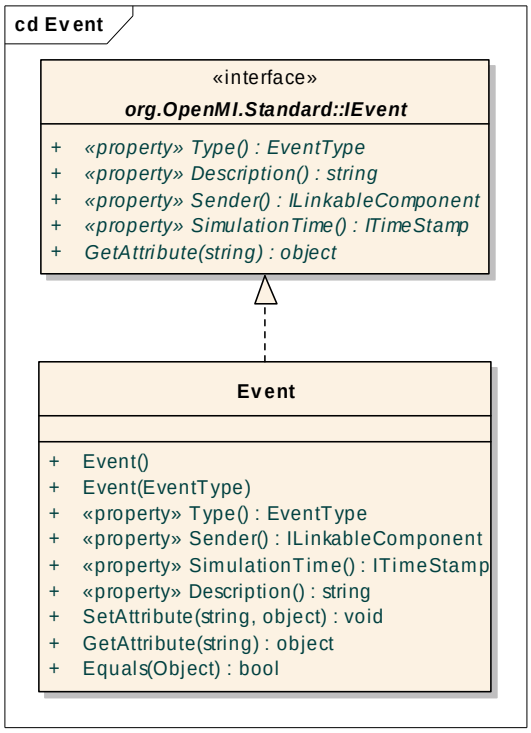

Figure 10 Implementation of the Event class

3.2.9 Exchange items

Figure 11 shows the implementation of the different types of exchange items. Exchange items are combinations of a quantity and an element set.

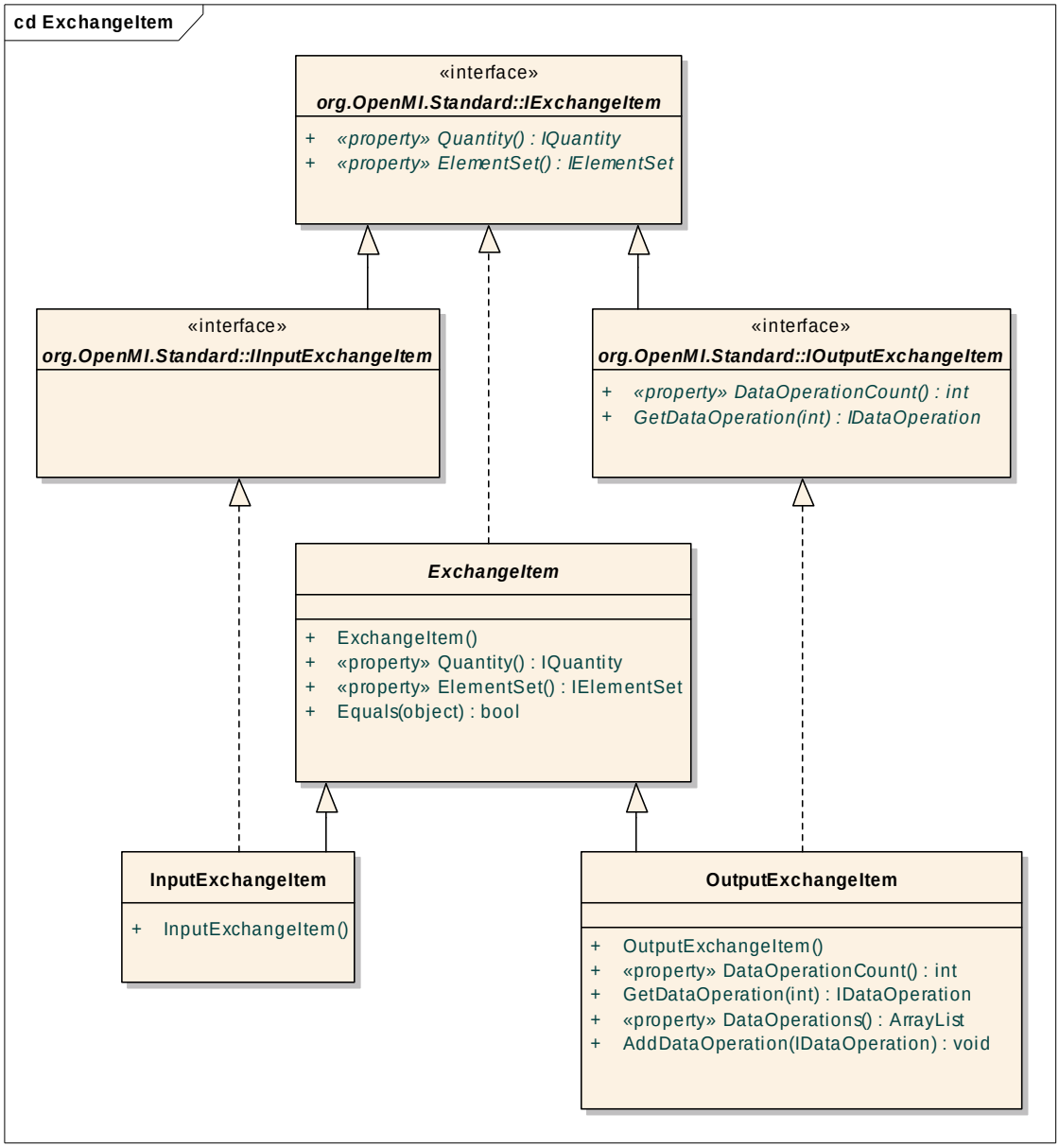

Figure 11 Implementation of the exchange items

3.2.10 Other interfaces

A number of interfaces will not be provided with a separate default implementation as they typically will be specific to the purpose of use. These include the IDiscreteTimes, IListener and IPublisher interfaces.

3.3 Dynamic view

The org.OpenMI.Backbone package contains no business logic by them selves. The classes are utilized in other packages. Hence, dynamic aspects are explained in the technical documentation of these packages.

Table 1 provides an overview of the exceptions generated by classes in the org.OpenMI.Backbone package.

Table 1 Overview of exceptions generated by org.OpenMI.Backbone

Class	Method	Exception
ElementSet	+GetElementIndex	Element with ID … not found.

3.4 Implementation remarks

3.4.1 C#-implementation

The C#-implementation is characterized by the usage of the following .NET features

- The LinkableComponent is implemented as a MarshallByRefObject.
- The TimeStamp class is implemented using the IComparable interface
- The Exception class is derived from System.Exception
-

The correct implementation of all methods has been tested using dedicated unit tests in combination with the NUnit framework for testing.

3.4.2 Java-implementation

The Java-implementation is still in development. No implementation details are available yet.