The OpenMI Document Series

OpenMI Standard 2 Reference

For the OpenMI (Version 2.0)

Grant agreement number LIFE06 ENV/UK/000409

Title	OpenMI Document Series: OpenMI Standard 2 Reference for the OpenMI (Version 2.0)
Editor	The OpenMI Association Technical Committee (OATC)
Authors	The OpenMI Association Technical Committee (OATC)
Current version	V 1.0
Date	30/11/2010
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.

Preface

The OpenMI Standard has been released by the OpenMI Association, which has a responsibility to support and maintain the OpenMI. This document provides a complete overview of version 2.0 of the OpenMI Standard and all its element details.

The Reference is part of the OpenMI report series, which specifies the OpenMI Standard 2, provides guidelines on its use and describes software facilities for migrating, setting up and running linked models.

Titles in the series include:

- Scope
- The OpenMI 'in a Nutshell'
- **OpenMI Standard 2 Reference** (this document)
- OpenMI Standard 2 Specification

The OpenMI is maintained by the OpenMI Association and this document, along with other more detailed documentation, can be obtained from www.openmi.org.

The official reference to this document is: The OpenMI Association (2010) *OpenMI Standard 2 Reference for the OpenMI (Version 2.0).* Part 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 www.openmi.org.

Contents

- Model Documentation

1. Model Detail

1 Element details

Standard

Type: **Package**

AdaptedOutputFactory - *(Logical diagram)*

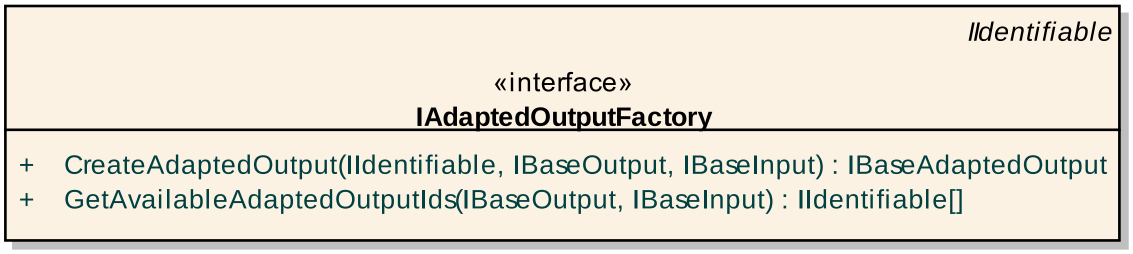

IIdentifiable
«interface»
IAdaptedOutputFactory
+ CreateAdaptedOutput(IIdentifiable, IBaseOutput, IBaseInput) : IBaseAdaptedOutput
+ GetAvailableAdaptedOutputIds(IBaseOutput, IBaseInput) : IIdentifiable[]

Figure: 1

Base Input Output - *(Logical diagram)*

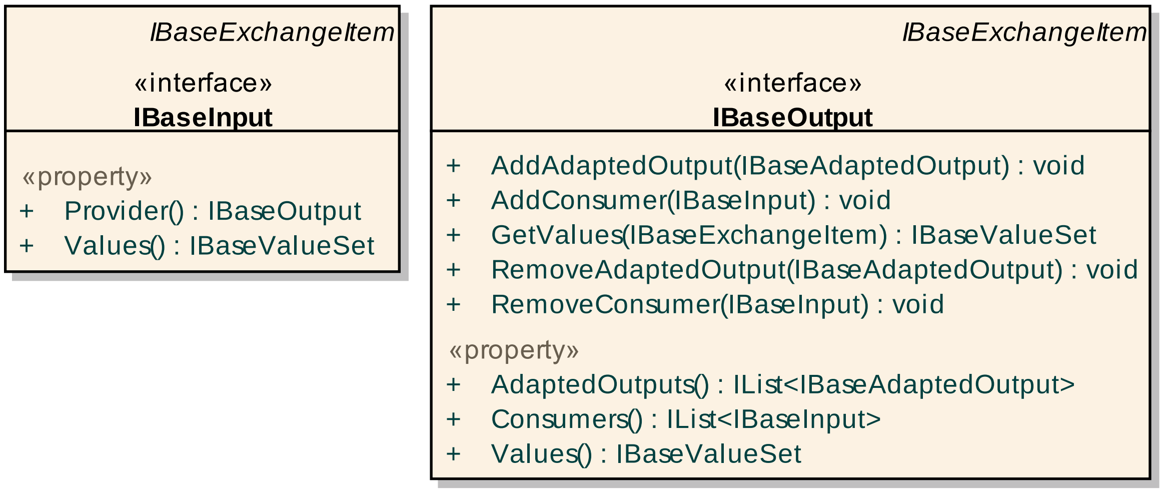

IBaseExchangeItem
«interface»
IBaseInput
«property»
+ Provider() : IBaseOutput
+ Values() : IBaseValueSet
IBaseExchangeItem
«interface»
IBaseOutput
+ AddAdaptedOutput(IBaseAdaptedOutput) : void
+ AddConsumer(IBaseInput) : void
+ GetValues(IBaseExchangeItem) : IBaseValueSet
+ RemoveAdaptedOutput(IBaseAdaptedOutput) : void
+ RemoveConsumer(IBaseInput) : void
«property»
+ AdaptedOutputs() : IList<IBaseAdaptedOutput>
+ Consumers() : IList<IBaseInput>
+ Values() : IBaseValueSet

Figure: 2

Basic - *(Logical diagram)*

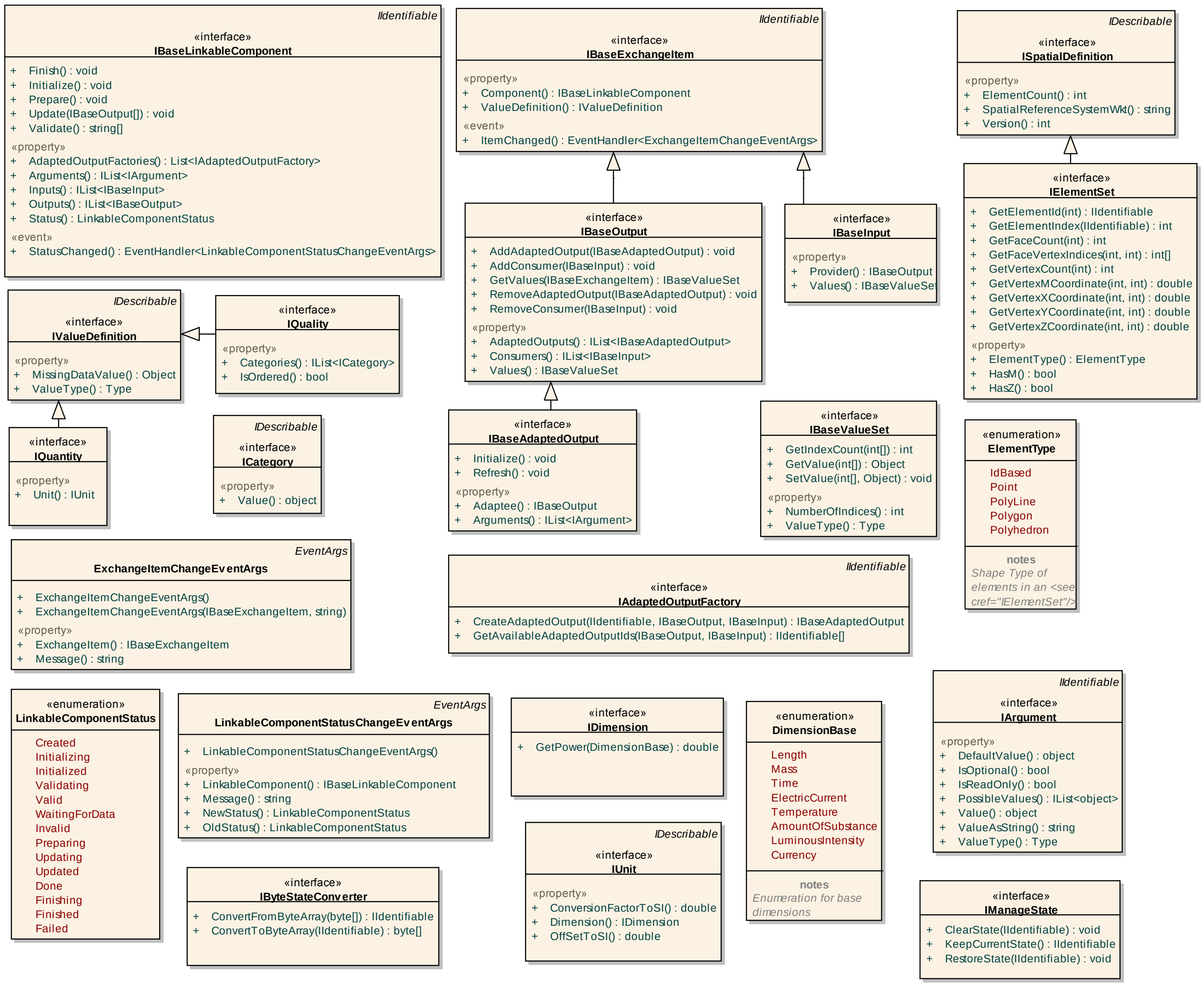

IIdentifiable
«interface»
IBaseLinkableComponent
+ Finish() : void
+ Initialize() : void
+ Prepare() : void
+ Update(IBaseOutput[]) : void
+ Validate() : string[]
«property»
+ AdaptedOutputFactories() : List<IAdaptedOutputFactory>
+ Arguments() : IList<IArgument>
+ Inputs() : IList<IBaseInput>
+ Outputs() : IList<IBaseOutput>
+ Status() : LinkableComponentStatus
«event»
+ StatusChanged() : EventHandler<LinkableComponentStatusChangeEventArgs>
IIdentifiable
«interface»
IBaseExchangeItem
«property»
+ Component() : IBaseLinkableComponent
+ ValueDefinition() : IValueDefinition
«event»
+ ItemChanged() : EventHandler<ExchangeItemChangeEventArgs>
IDescribable
«interface»
ISpatialDefinition
«property»
+ ElementCount() : int
+ SpatialReferenceSystemWkt() : string
+ Version() : int
«interface»
IElementSet
+ GetElementId(int) : IIdentifiable
+ GetElementIndex(IIdentifiable) : int
+ GetFaceCount(int) : int
+ GetFaceVertexIndices(int, int) : int[]
+ GetVertexCount(int) : int
+ GetVertexMCoordinate(int, int) : double
+ GetVertexXCoordinate(int, int) : double
+ GetVertexYCoordinate(int, int) : double
+ GetVertexZCoordinate(int, int) : double
«property»
+ ElementType() : ElementType
+ HasM() : bool
+ HasZ() : bool
«interface»
IBaseOutput
+ AddAdaptedOutput(IBaseAdaptedOutput) : void
+ AddConsumer(IBaseInput) : void
+ GetValues(IBaseExchangeItem) : IBaseValueSet
+ RemoveAdaptedOutput(IBaseAdaptedOutput) : void
+ RemoveConsumer(IBaseInput) : void
«property»
+ AdaptedOutputs() : IList<IBaseAdaptedOutput>
+ Consumers() : IList<IBaseInput>
+ Values() : IBaseValueSet
«interface»
IBaseInput
«property»
+ Provider() : IBaseOutput
+ Values() : IBaseValueSet
IDescribable
«interface»
IValueDefinition
«property»
+ MissingDataValue() : Object
+ ValueType() : Type
«interface»
IQuality
«property»
+ Categories() : IList<ICategory>
+ IsOrdered() : bool
«interface»
IQuantity
«property»
+ Unit() : IUnit
IDescribable
«interface»
ICategory
«property»
+ Value() : object
«interface»
IBaseAdaptedOutput
+ Initialize() : void
+ Refresh() : void
«property»
+ Adaptee() : IBaseOutput
+ Arguments() : IList<IArgument>
«interface»
IBaseValueSet
+ GetIndexCount(int[]) : int
+ GetValue(int[]) : Object
+ SetValue(int[], Object) : void
«property»
+ NumberOfIndices() : int
+ ValueType() : Type
«enumeration»
ElementType
IdBased
Point
PolyLine
Polygon
Polyhedron
notes
Shape Type of elements in an <see cref="IElementSet"/>
EventArgs
ExchangeItemChangeEventArgs
+ ExchangeItemChangeEventArgs()
+ ExchangeItemChangeEventArgs(IBaseExchangeItem, string)
«property»
+ ExchangeItem() : IBaseExchangeItem
+ Message() : string
IIdentifiable
«interface»
IAdaptedOutputFactory
+ CreateAdaptedOutput(IIdentifiable, IBaseOutput, IBaseInput) : IBaseAdaptedOutput
+ GetAvailableAdaptedOutputIds(IBaseOutput, IBaseInput) : IIdentifiable[]
«enumeration»
LinkableComponentStatus
Created
Initializing
Initialized
Validating
Valid
WaitingForData
Invalid
Preparing
Updating
Updated
Done
Finishing
Finished
Failed
EventArgs
LinkableComponentStatusChangeEventArgs
+ LinkableComponentStatusChangeEventArgs()
«property»
+ LinkableComponent() : IBaseLinkableComponent
+ Message() : string
+ NewStatus() : LinkableComponentStatus
+ OldStatus() : LinkableComponentStatus
«interface»
IDimension
+ GetPower(DimensionBase) : double
«enumeration»
DimensionBase
Length
Mass
Time
ElectricCurrent
Temperature
AmountOfSubstance
LuminousIntensity
Currency
notes
Enumeration for base dimensions
IIdentifiable
«interface»
IArgument
«property»
+ DefaultValue() : object
+ IsOptional() : bool
+ IsReadOnly() : bool
+ PossibleValues() : IList<object>
+ Value() : object
+ ValueAsString() : string
+ ValueType() : Type
«interface»
IByteStateConverter
+ ConvertFromByteArray(byte[]) : IIdentifiable
+ ConvertToByteArray(IIdentifiable) : byte[]
IDescribable
«interface»
IUnit
«property»
+ ConversionFactorToSI() : double
+ Dimension() : IDimension
+ OffSetToSI() : double
«interface»
IManageState
+ ClearState(IIdentifiable) : void
+ KeepCurrentState() : IIdentifiable
+ RestoreState(IIdentifiable) : void

Figure: 3

<u>Data Definition</u> - *(Logical diagram)*

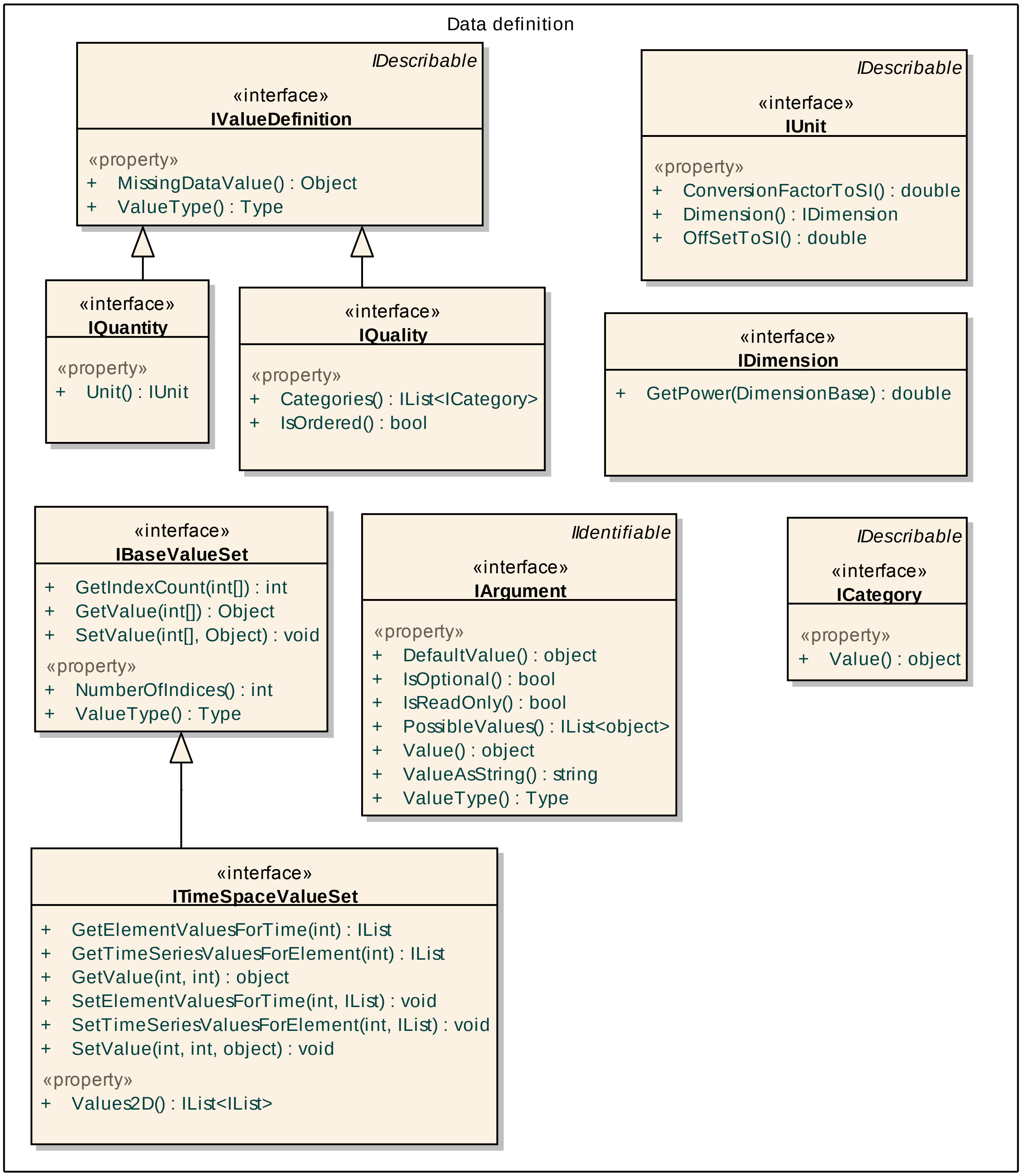

Data definition
IDescribable
«interface»
IValueDefinition
«property»
+ MissingDataValue() : Object
+ ValueType() : Type
IDescribable
«interface»
IUnit
«property»
+ ConversionFactorToSI() : double
+ Dimension() : IDimension
+ OffSetToSI() : double
«interface»
IQuantity
«property»
+ Unit() : IUnit
«interface»
IQuality
«property»
+ Categories() : IList<ICategory>
+ IsOrdered() : bool
«interface»
IDimension
+ GetPower(DimensionBase) : double
«interface»
IBaseValueSet
+ GetIndexCount(int[]) : int
+ GetValue(int[]) : Object
+ SetValue(int[], Object) : void
«property»
+ NumberOfIndices() : int
+ ValueType() : Type
IIdentifiable
«interface»
IArgument
«property»
+ DefaultValue() : object
+ IsOptional() : bool
+ IsReadOnly() : bool
+ PossibleValues() : IList<object>
+ Value() : object
+ ValueAsString() : string
+ ValueType() : Type
IDescribable
«interface»
ICategory
«property»
+ Value() : object
«interface»
ITimeSpaceValueSet
+ GetElementValuesForTime(int) : IList
+ GetTimeSeriesValuesForElement(int) : IList
+ GetValue(int, int) : object
+ SetElementValuesForTime(int, IList) : void
+ SetTimeSeriesValuesForElement(int, IList) : void
+ SetValue(int, int, object) : void
«property»
+ Values2D() : IList<IList>

Figure: 4

ExchangeItem - *(Logical diagram)*

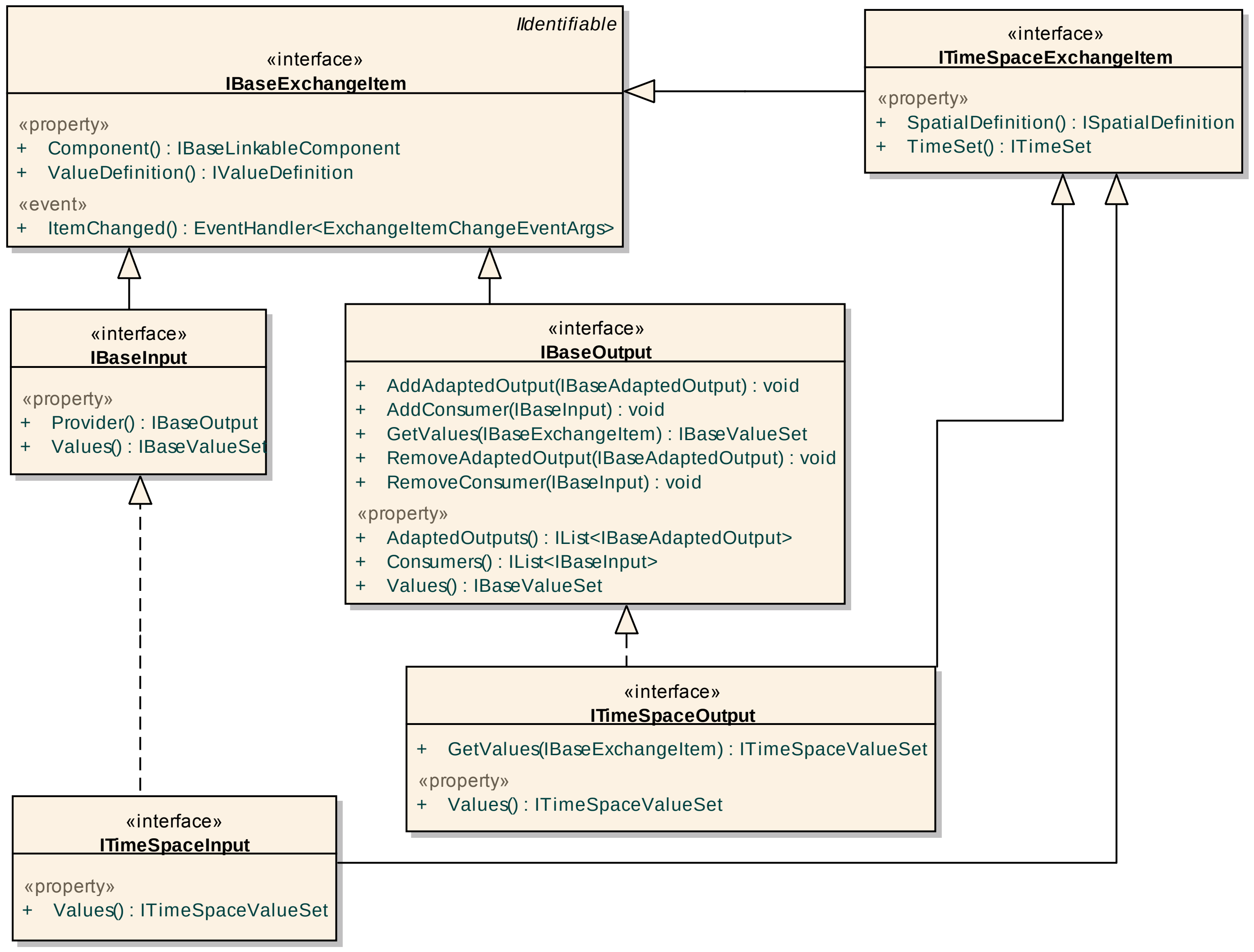

IIdentifiable
«interface»
IBaseExchangeItem
«property»
+ Component() : IBaseLinkableComponent
+ ValueDefinition() : IValueDefinition
«event»
+ ItemChanged() : EventHandler<ExchangeItemChangeEventArgs>
«interface»
ITimeSpaceExchangeItem
«property»
+ SpatialDefinition() : ISpatialDefinition
+ TimeSet() : ITimeSet
«interface»
IBaseInput
«property»
+ Provider() : IBaseOutput
+ Values() : IBaseValueSet
«interface»
IBaseOutput
+ AddAdaptedOutput(IBaseAdaptedOutput) : void
+ AddConsumer(IBaseInput) : void
+ GetValues(IBaseExchangeItem) : IBaseValueSet
+ RemoveAdaptedOutput(IBaseAdaptedOutput) : void
+ RemoveConsumer(IBaseInput) : void
«property»
+ AdaptedOutputs() : IList<IBaseAdaptedOutput>
+ Consumers() : IList<IBaseInput>
+ Values() : IBaseValueSet
«interface»
ITimeSpaceOutput
+ GetValues(IBaseExchangeItem) : ITimeSpaceValueSet
«property»
+ Values() : ITimeSpaceValueSet
«interface»
ITimeSpaceInput
«property»
+ Values() : ITimeSpaceValueSet

Figure: 5

ExchangeItem Status - *(Logical diagram)*

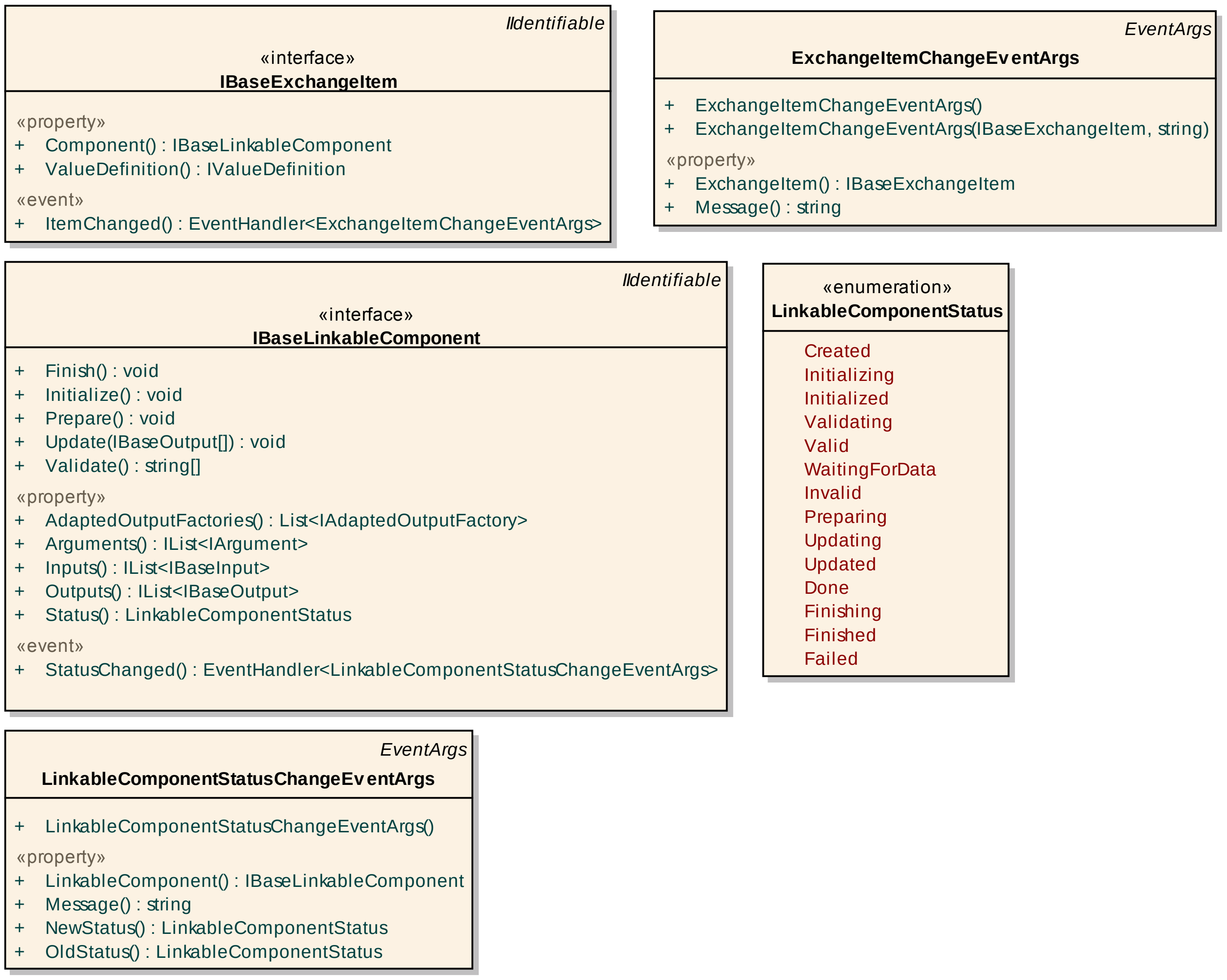

IIdentifiable
«interface»
IBaseExchangeItem
«property»
+ Component() : IBaseLinkableComponent
+ ValueDefinition() : IValueDefinition
«event»
+ ItemChanged() : EventHandler<ExchangeItemChangeEventArgs>
EventArgs
ExchangeItemChangeEventArgs
+ ExchangeItemChangeEventArgs()
+ ExchangeItemChangeEventArgs(IBaseExchangeItem, string)
«property»
+ ExchangeItem() : IBaseExchangeItem
+ Message() : string
IIdentifiable
«interface»
IBaseLinkableComponent
+ Finish() : void
+ Initialize() : void
+ Prepare() : void
+ Update(IBaseOutput[]) : void
+ Validate() : string[]
«property»
+ AdaptedOutputFactories() : List<IAdaptedOutputFactory>
+ Arguments() : IList<IArgument>
+ Inputs() : IList<IBaseInput>
+ Outputs() : IList<IBaseOutput>
+ Status() : LinkableComponentStatus
«event»
+ StatusChanged() : EventHandler<LinkableComponentStatusChangeEventArgs>
«enumeration»
LinkableComponentStatus
Created
Initializing
Initialized
Validating
Valid
WaitingForData
Invalid
Preparing
Updating
Updated
Done
Finishing
Finished
Failed
EventArgs
LinkableComponentStatusChangeEventArgs
+ LinkableComponentStatusChangeEventArgs()
«property»
+ LinkableComponent() : IBaseLinkableComponent
+ Message() : string
+ NewStatus() : LinkableComponentStatus
+ OldStatus() : LinkableComponentStatus

Figure: 6

How - *(Logical diagram)*

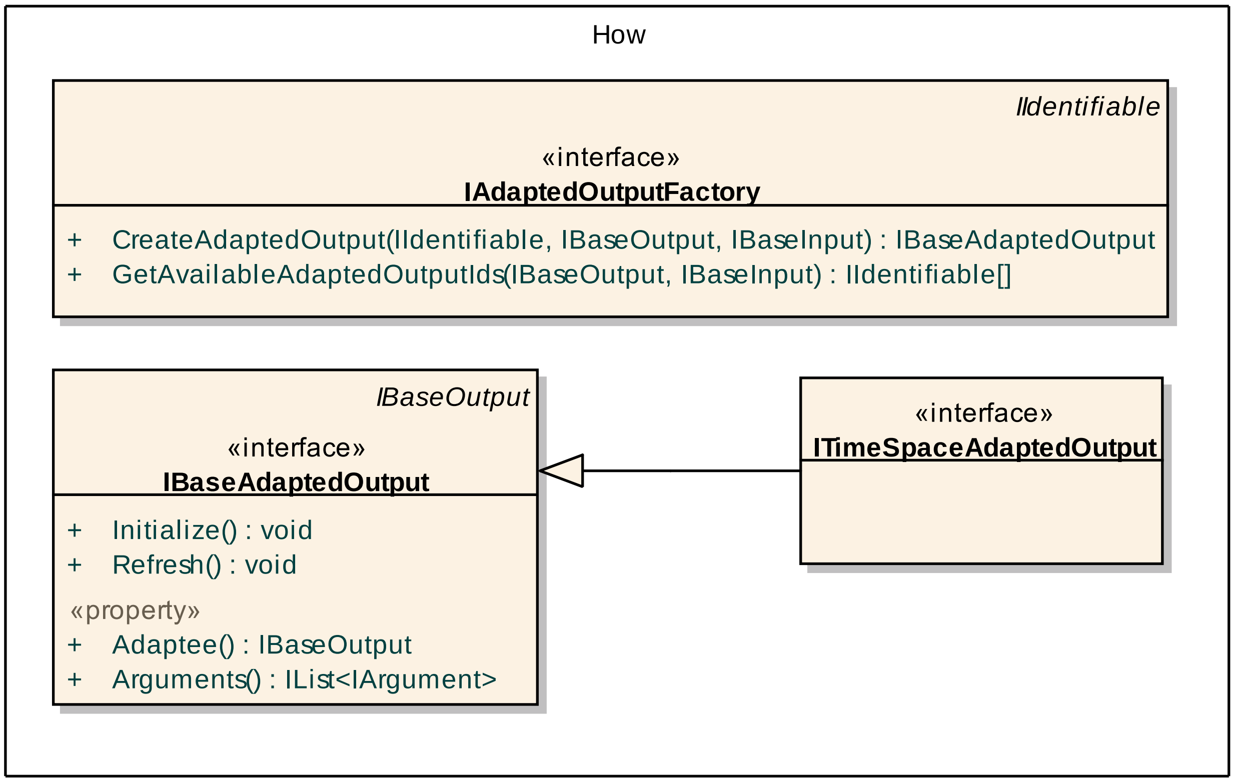

How
IIdentifiable
«interface»
IAdaptedOutputFactory
+ CreateAdaptedOutput(IIdentifiable, IBaseOutput, IBaseInput) : IBaseAdaptedOutput
+ GetAvailableAdaptedOutputIds(IBaseOutput, IBaseInput) : IIdentifiable[]
IBaseOutput
«interface»
IBaseAdaptedOutput
+ Initialize() : void
+ Refresh() : void
«property»
+ Adaptee() : IBaseOutput
+ Arguments() : IList<IArgument>
«interface»
ITimeSpaceAdaptedOutput

Figure: 7

<u>Identifiable / Describable</u> - *(Logical diagram)*

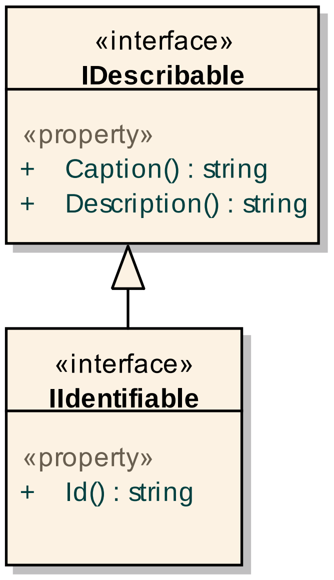
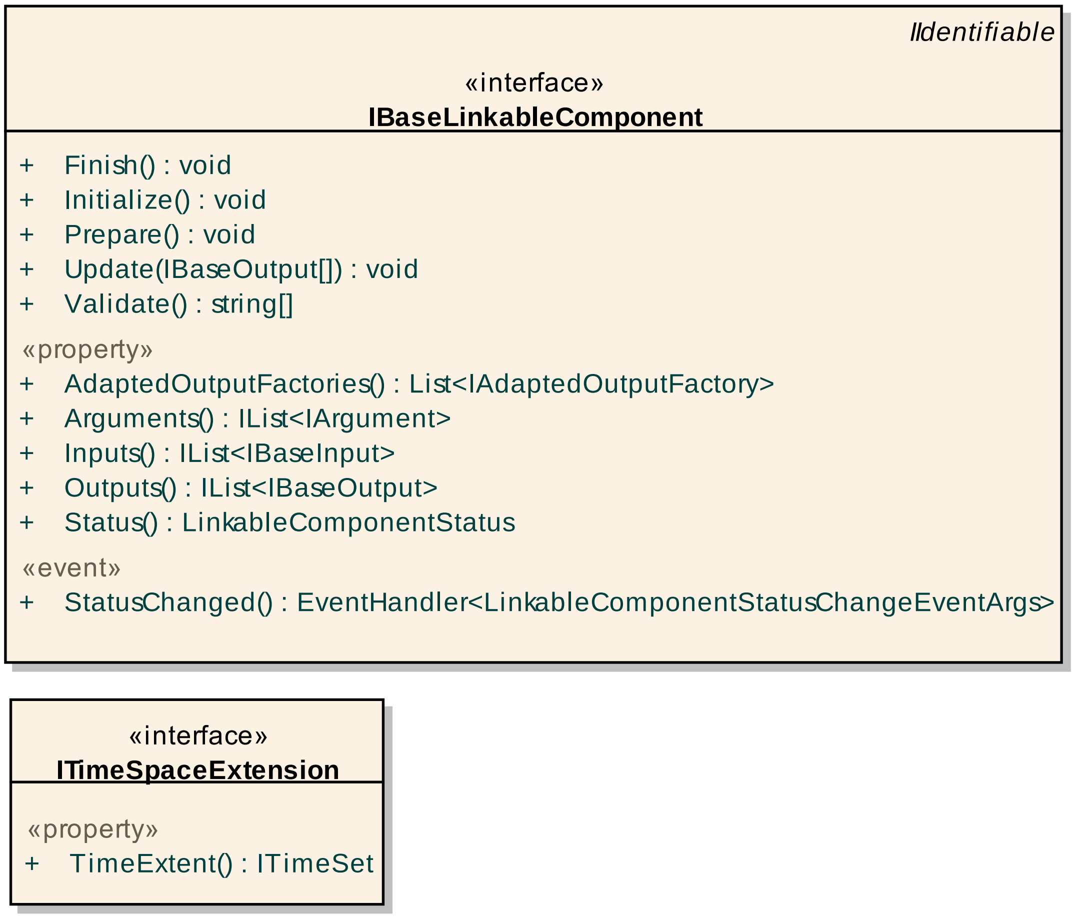


Figure: 8

<u>Linkable component</u> - *(Logical diagram)*



IIdentifiable
«interface»
IBaseLinkableComponent
+ Finish() : void
+ Initialize() : void
+ Prepare() : void
+ Update(IBaseOutput[]) : void
+ Validate() : string[]
«property»
+ AdaptedOutputFactories() : List<IAdaptedOutputFactory>
+ Arguments() : IList<IArgument>
+ Inputs() : IList<IBaseInput>
+ Outputs() : IList<IBaseOutput>
+ Status() : LinkableComponentStatus
«event»
+ StatusChanged() : EventHandler<LinkableComponentStatusChangeEventArgs>
«interface»
ITimeSpaceExtension
«property»
+ TimeExtent() : ITimeSet

Figure: 9

<u>Linkable component - short</u> - *(Logical diagram)*

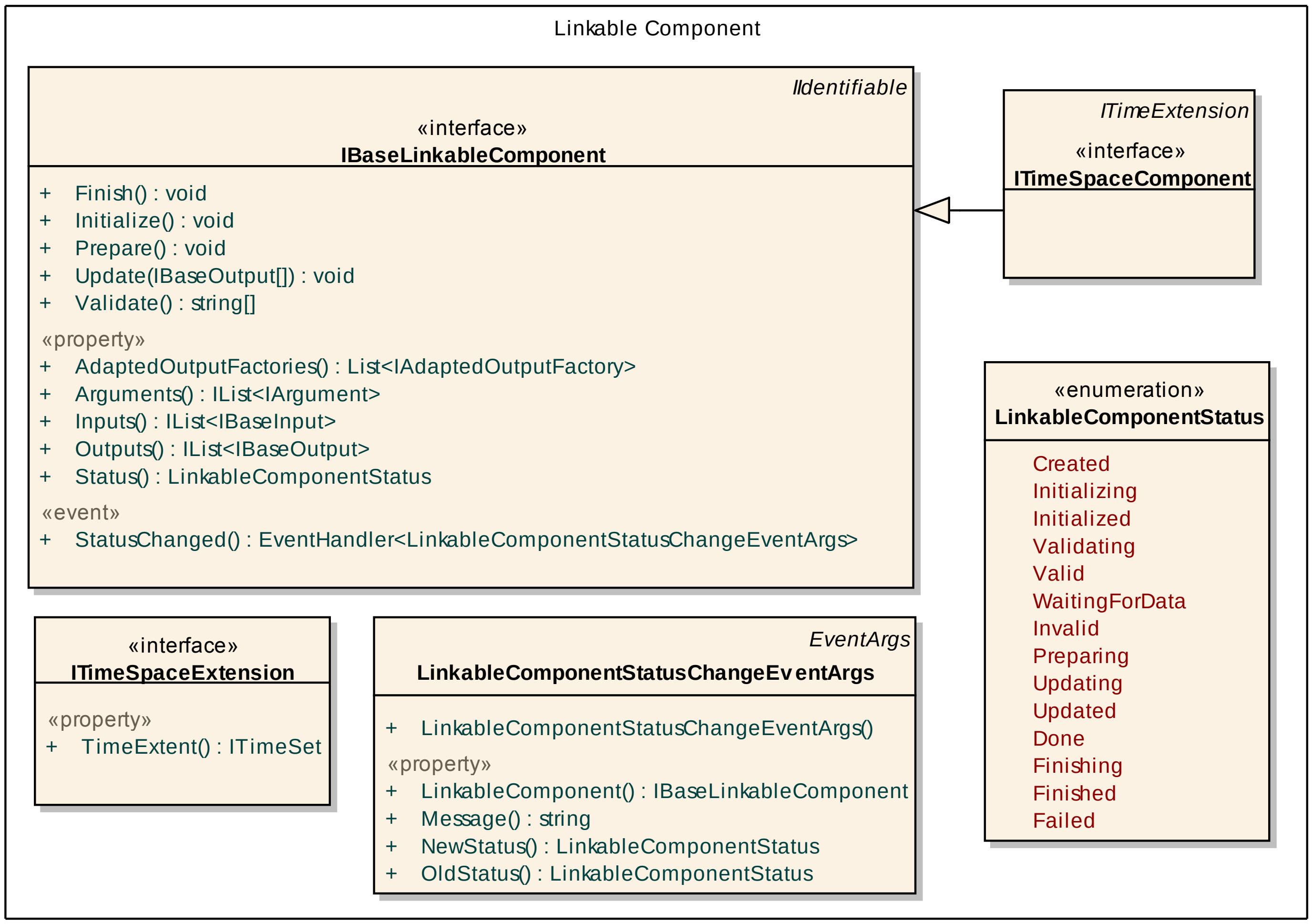

Linkable Component
IIdentifiable
«interface»
IBaseLinkableComponent
+ Finish() : void
+ Initialize() : void
+ Prepare() : void
+ Update(IBaseOutput[]) : void
+ Validate() : string[]
«property»
+ AdaptedOutputFactories() : List<IAdaptedOutputFactory>
+ Arguments() : IList<IArgument>
+ Inputs() : IList<IBaseInput>
+ Outputs() : IList<IBaseOutput>
+ Status() : LinkableComponentStatus
«event»
+ StatusChanged() : EventHandler<LinkableComponentStatusChangeEventArgs>
ITimeExtension
«interface»
ITimeSpaceComponent
«enumeration»
LinkableComponentStatus
Created
Initializing
Initialized
Validating
Valid
WaitingForData
Invalid
Preparing
Updating
Updated
Done
Finishing
Finished
Failed
«interface»
ITimeSpaceExtension
«property»
+ TimeExtent() : ITimeSet
EventArgs
LinkableComponentStatusChangeEventArgs
+ LinkableComponentStatusChangeEventArgs()
«property»
+ LinkableComponent() : IBaseLinkableComponent
+ Message() : string
+ NewStatus() : LinkableComponentStatus
+ OldStatus() : LinkableComponentStatus

Figure: 10

Output - *(Logical diagram)*

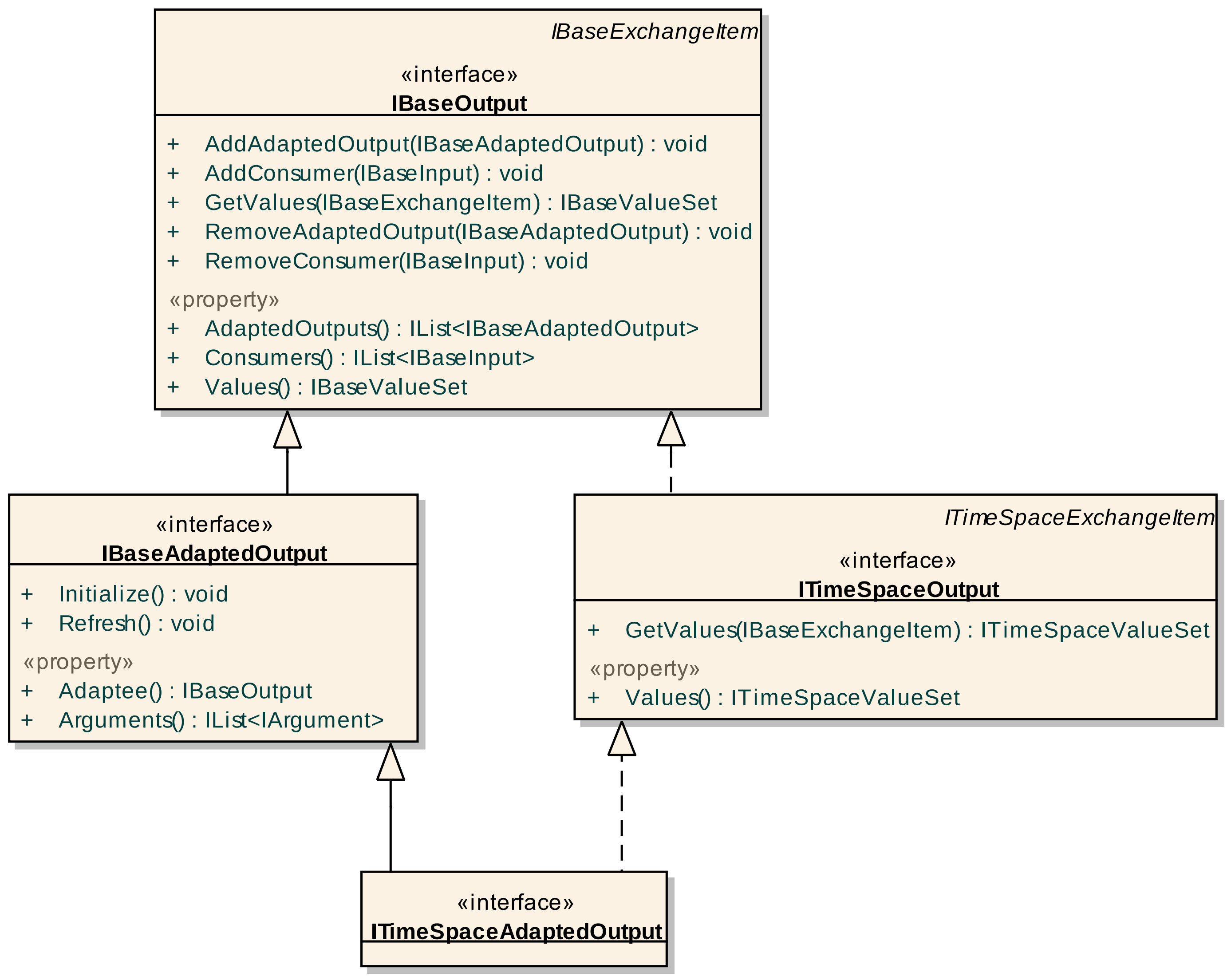

IBaseExchangeItem
«interface»
IBaseOutput
+ AddAdaptedOutput(IBaseAdaptedOutput) : void
+ AddConsumer(IBaseInput) : void
+ GetValues(IBaseExchangeItem) : IBaseValueSet
+ RemoveAdaptedOutput(IBaseAdaptedOutput) : void
+ RemoveConsumer(IBaseInput) : void
«property»
+ AdaptedOutputs() : IList<IBaseAdaptedOutput>
+ Consumers() : IList<IBaseInput>
+ Values() : IBaseValueSet
«interface»
IBaseAdaptedOutput
+ Initialize() : void
+ Refresh() : void
«property»
+ Adaptee() : IBaseOutput
+ Arguments() : IList<IArgument>
ITimeSpaceExchangeItem
«interface»
ITimeSpaceOutput
+ GetValues(IBaseExchangeItem) : ITimeSpaceValueSet
«property»
+ Values() : ITimeSpaceValueSet
«interface»
ITimeSpaceAdaptedOutput

Figure: 11

Spatial - *(Logical diagram)*

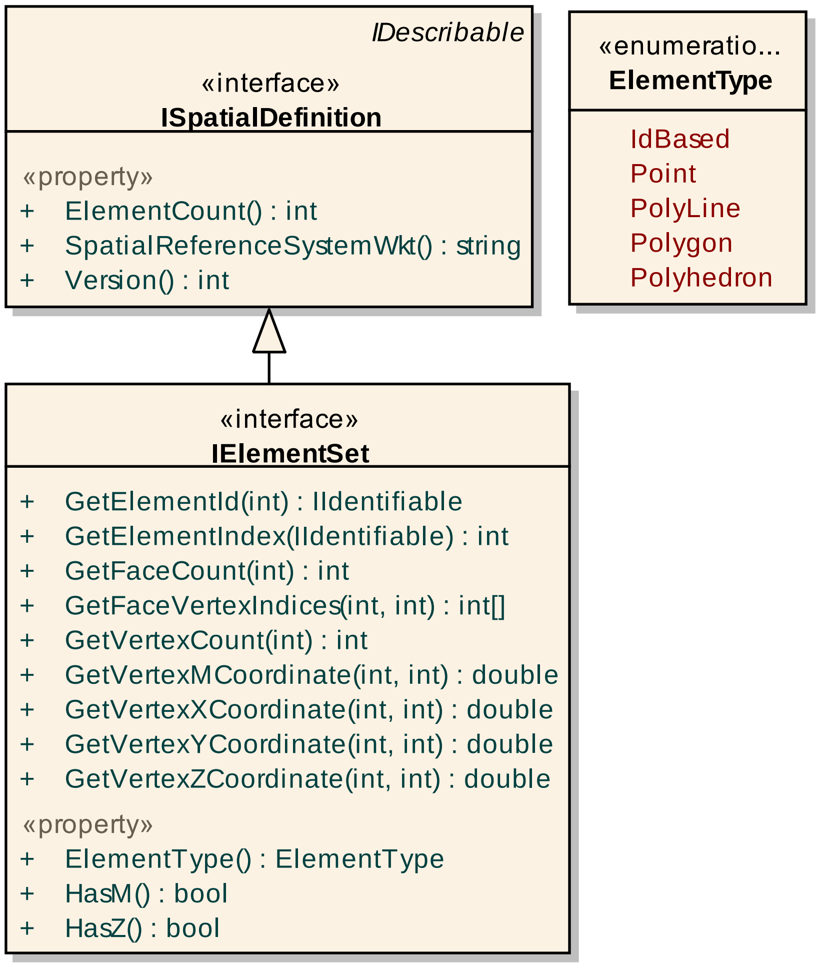

IDescribable
«interface»
ISpatialDefinition
«property»
+ ElementCount() : int
+ SpatialReferenceSystemWkt() : string
+ Version() : int
«enumeratio...
ElementType
IdBased
Point
PolyLine
Polygon
Polyhedron
«interface»
IElementSet
+ GetElementId(int) : IIdentifiable
+ GetElementIndex(IIdentifiable) : int
+ GetFaceCount(int) : int
+ GetFaceVertexIndices(int, int) : int[]
+ GetVertexCount(int) : int
+ GetVertexMCoordinate(int, int) : double
+ GetVertexXCoordinate(int, int) : double
+ GetVertexYCoordinate(int, int) : double
+ GetVertexZCoordinate(int, int) : double
«property»
+ ElementType() : ElementType
+ HasM() : bool
+ HasZ() : bool

Figure: 12

<u>**Standard - all**</u> - *(Logical diagram)*

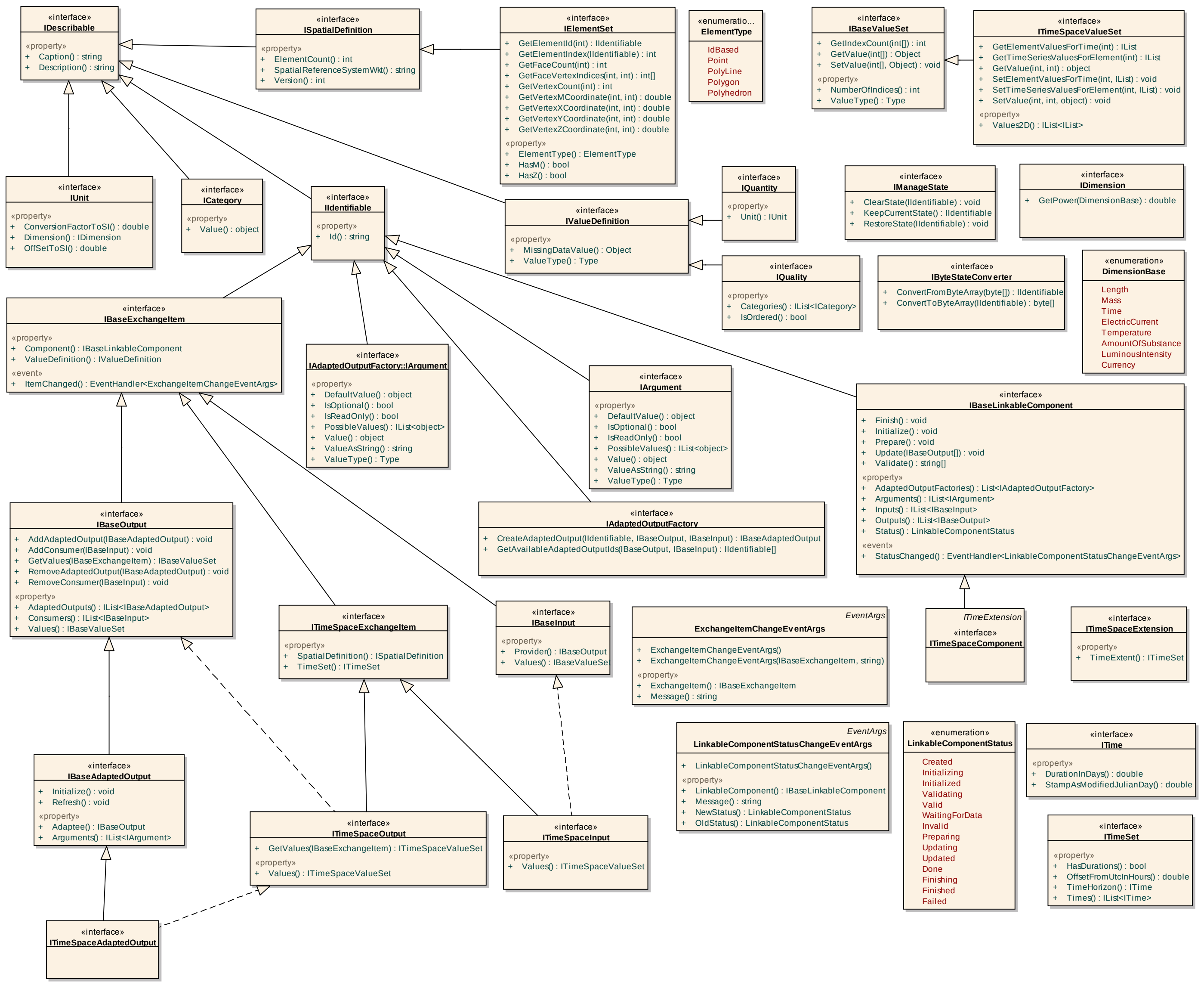

«interface» IDescribable
«property» + Caption() : string + Description() : string
«interface» ISpatialDefinition
«property» + ElementCount() : int + SpatialReferenceSystemWkt() : string + Version() : int
«interface» IElementSet
+ GetElementId(int) : IIdentifiable
+ GetElementIndex(IIdentifiable) : int
+ GetFaceCount(int) : int
+ GetFaceVertexIndices(int, int) : int[]
+ GetVertexCount(int) : int
+ GetVertexMCoordinate(int, int) : double
+ GetVertexXCoordinate(int, int) : double
+ GetVertexYCoordinate(int, int) : double
+ GetVertexZCoordinate(int, int) : double
«property» + ElementType() : ElementType + HasM() : bool + HasZ() : bool
«enumeratio... ElementType
IdBased Point PolyLine Polygon Polyhedron
«interface» IBaseValueSet
+ GetIndexCount(int[]) : int + GetValue(int[]) : Object + SetValue(int[], Object) : void
«property» + NumberOfIndices() : int + ValueType() : Type
«interface» ITimeSpaceValueSet
+ GetElementValuesForTime(int) : IList
+ GetTimeSeriesValuesForElement(int) : IList
+ GetValue(int, int) : object
+ SetElementValuesForTime(int, IList) : void
+ SetTimeSeriesValuesForElement(int, IList) : void
+ SetValue(int, int, object) : void
«property» + Values2D() : IList<IList>
«interface» IUnit
«property» + ConversionFactorToSI() : double + Dimension() : IDimension + OffSetToSI() : double
«interface» ICategory
«property» + Value() : object
«interface» IIdentifiable
«property» + Id() : string
«interface» IValueDefinition
«property» + MissingDataValue() : Object + ValueType() : Type
«interface» IQuantity
«property» + Unit() : IUnit
«interface» IManageState
+ ClearState(IIdentifiable) : void + KeepCurrentState() : IIdentifiable + RestoreState(IIdentifiable) : void
«interface» IDimension
+ GetPower(DimensionBase) : double
«interface» IQuality
«property» + Categories() : IList<ICategory> + IsOrdered() : bool
«interface» IByteStateConverter
+ ConvertFromByteArray(byte[]) : IIdentifiable + ConvertToByteArray(IIdentifiable) : byte[]
«enumeration» DimensionBase
Length Mass Time ElectricCurrent Temperature AmountOfSubstance LuminousIntensity Currency
«interface» IBaseExchangeItem
«property» + Component() : IBaseLinkableComponent + ValueDefinition() : IValueDefinition
«event» + ItemChanged() : EventHandler<ExchangeItemChangeEventArgs>
«interface» IAdaptedOutputFactory::IArgument
«property» + DefaultValue() : object + IsOptional() : bool + IsReadOnly() : bool + PossibleValues() : IList<object> + Value() : object + ValueAsString() : string + ValueType() : Type
«interface» IArgument
«property» + DefaultValue() : object + IsOptional() : bool + IsReadOnly() : bool + PossibleValues() : IList<object> + Value() : object + ValueAsString() : string + ValueType() : Type
«interface» IBaseLinkableComponent
+ Finish() : void + Initialize() : void + Prepare() : void + Update(IBaseOutput[]) : void + Validate() : string[]
«property» + AdaptedOutputFactories() : List<IAdaptedOutputFactory> + Arguments() : IList<IArgument> + Inputs() : IList<IBaseInput> + Outputs() : IList<IBaseOutput> + Status() : LinkableComponentStatus
«event» + StatusChanged() : EventHandler<LinkableComponentStatusChangeEventArgs>
«interface» IBaseOutput
+ AddAdaptedOutput(IBaseAdaptedOutput) : void
+ AddConsumer(IBaseInput) : void
+ GetValues(IBaseExchangeItem) : IBaseValueSet
+ RemoveAdaptedOutput(IBaseAdaptedOutput) : void
+ RemoveConsumer(IBaseInput) : void
«property» + AdaptedOutputs() : IList<IBaseAdaptedOutput> + Consumers() : IList<IBaseInput> + Values() : IBaseValueSet
«interface» IAdaptedOutputFactory
+ CreateAdaptedOutput(IIdentifiable, IBaseOutput, IBaseInput) : IBaseAdaptedOutput
+ GetAvailableAdaptedOutputIds(IBaseOutput, IBaseInput) : IIdentifiable[]
«interface» ITimeSpaceExchangeItem
«property» + SpatialDefinition() : ISpatialDefinition + TimeSet() : ITimeSet
«interface» IBaseInput
«property» + Provider() : IBaseOutput + Values() : IBaseValueSet
EventArgs ExchangeItemChangeEventArgs
+ ExchangeItemChangeEventArgs() + ExchangeItemChangeEventArgs(IBaseExchangeItem, string)
«property» + ExchangeItem() : IBaseExchangeItem + Message() : string
ITimeExtension «interface» ITimeSpaceComponent
«interface» ITimeSpaceExtension
«property» + TimeExtent() : ITimeSet
EventArgs LinkableComponentStatusChangeEventArgs
+ LinkableComponentStatusChangeEventArgs()
«property» + LinkableComponent() : IBaseLinkableComponent + Message() : string + NewStatus() : LinkableComponentStatus + OldStatus() : LinkableComponentStatus
«enumeration» LinkableComponentStatus
Created Initializing Initialized Validating Valid WaitingForData Invalid Preparing Updating Updated Done Finishing Finished Failed
«interface» ITime
«property» + DurationInDays() : double + StampAsModifiedJulianDay() : double
«interface» IBaseAdaptedOutput
+ Initialize() : void + Refresh() : void
«property» + Adaptee() : IBaseOutput + Arguments() : IList<IArgument>
«interface» ITimeSpaceOutput
+ GetValues(IBaseExchangeItem) : ITimeSpaceValueSet
«property» + Values() : ITimeSpaceValueSet
«interface» ITimeSpaceInput
«property» + Values() : ITimeSpaceValueSet
«interface» ITimeSet
«property» + HasDurations() : bool + OffsetFromUtcInHours() : double + TimeHorizon() : ITime + Times() : IList<ITime>
«interface» ITimeSpaceAdaptedOutput

Figure: 13

<u>TimeExtension</u> - *(Logical diagram)*

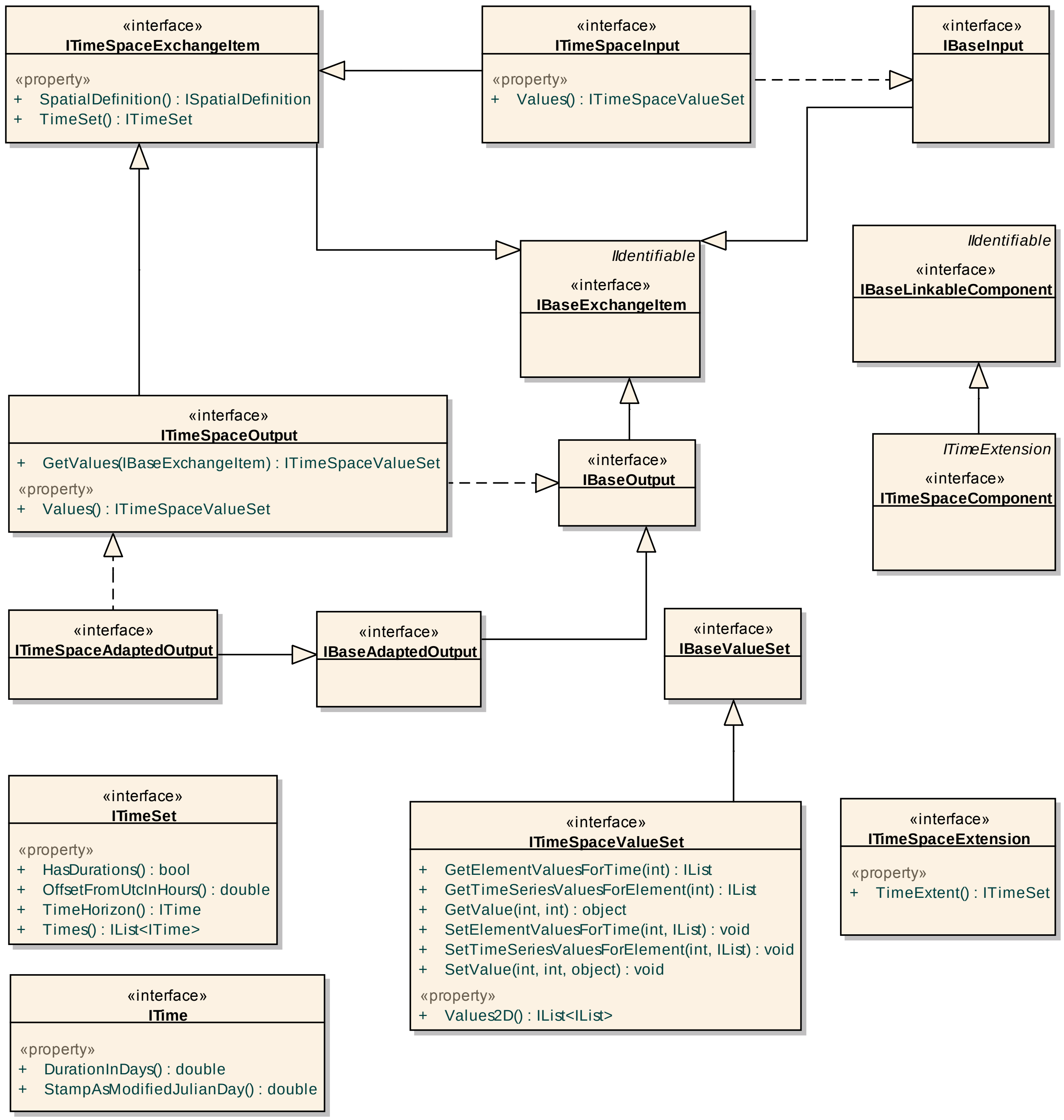


Figure: 14

Value definition - *(Logical diagram)*

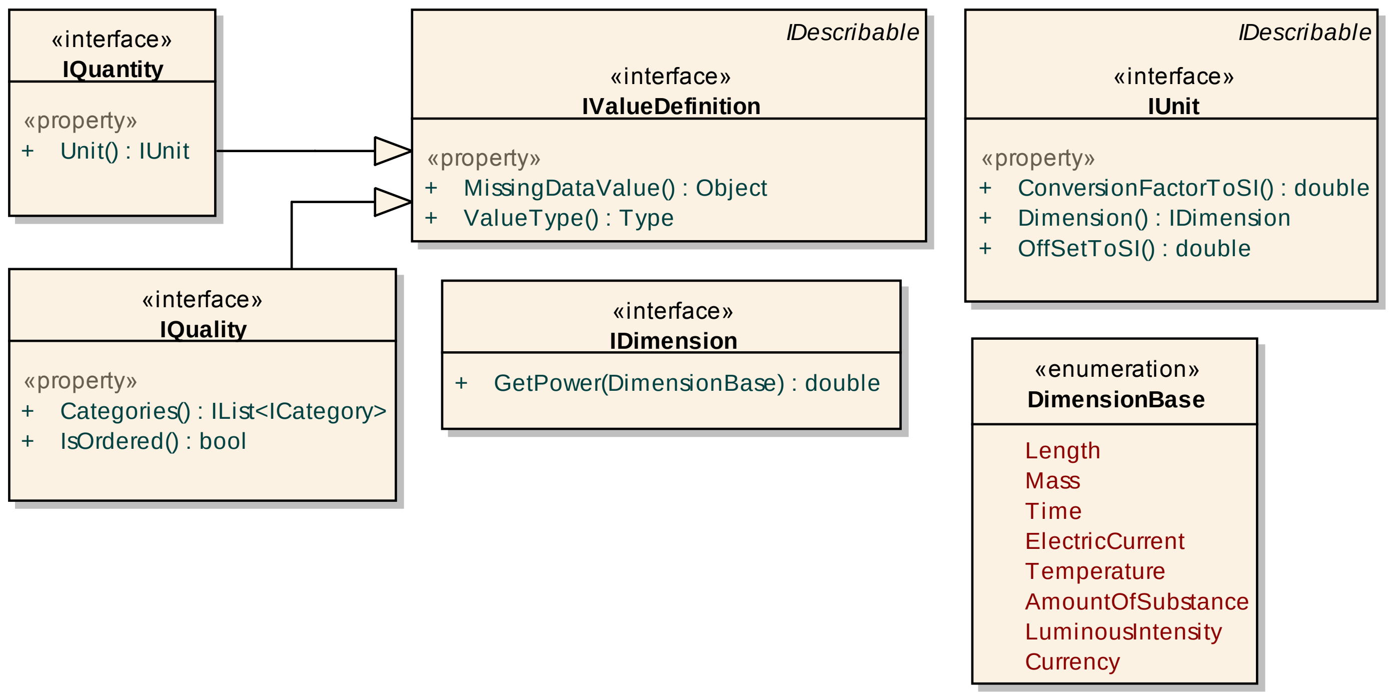


Figure: 15

ValueSet - *(Logical diagram)*

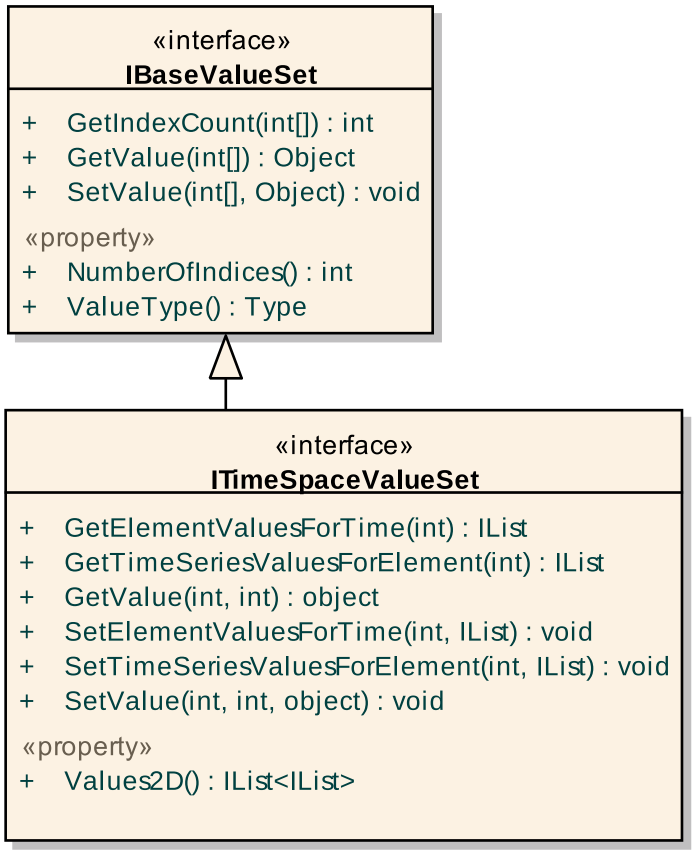

«interface»
IBaseValueSet
+ GetIndexCount(int[]) : int
+ GetValue(int[]) : Object
+ SetValue(int[], Object) : void
«property»
+ NumberOfIndices() : int
+ ValueType() : Type
«interface»
ITimeSpaceValueSet
+ GetElementValuesForTime(int) : IList
+ GetTimeSeriesValuesForElement(int) : IList
+ GetValue(int, int) : object
+ SetElementValuesForTime(int, IList) : void
+ SetTimeSeriesValuesForElement(int, IList) : void
+ SetValue(int, int, object) : void
«property»
+ Values2D() : IList<IList>

Figure: 16

<u>What and how can be exchanged</u> - *(Logical diagram)*

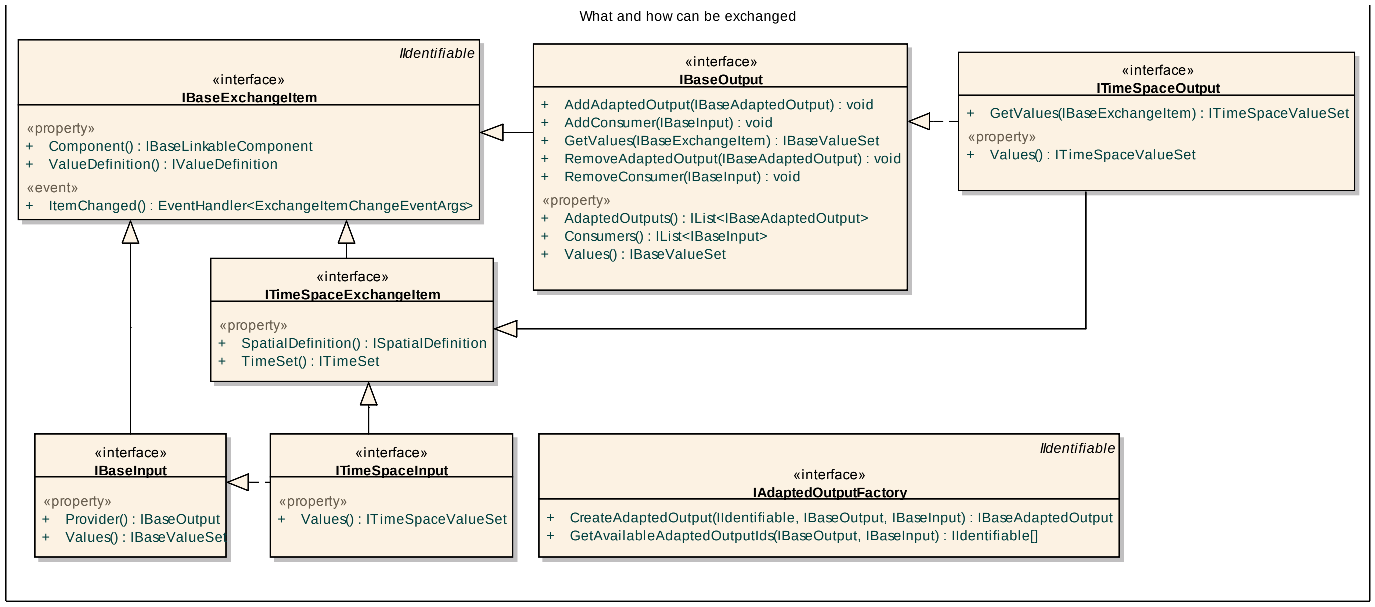

What and how can be exchanged
IIdentifiable
«interface»
IBaseExchangeItem
«property»
+ Component() : IBaseLinkableComponent
+ ValueDefinition() : IValueDefinition
«event»
+ ItemChanged() : EventHandler<ExchangeItemChangeEventArgs>
«interface»
IBaseOutput
+ AddAdaptedOutput(IBaseAdaptedOutput) : void
+ AddConsumer(IBaseInput) : void
+ GetValues(IBaseExchangeItem) : IBaseValueSet
+ RemoveAdaptedOutput(IBaseAdaptedOutput) : void
+ RemoveConsumer(IBaseInput) : void
«property»
+ AdaptedOutputs() : IList<IBaseAdaptedOutput>
+ Consumers() : IList<IBaseInput>
+ Values() : IBaseValueSet
«interface»
ITimeSpaceOutput
+ GetValues(IBaseExchangeItem) : ITimeSpaceValueSet
«property»
+ Values() : ITimeSpaceValueSet
«interface»
ITimeSpaceExchangeItem
«property»
+ SpatialDefinition() : ISpatialDefinition
+ TimeSet() : ITimeSet
«interface»
IBaseInput
«property»
+ Provider() : IBaseOutput
+ Values() : IBaseValueSet
«interface»
ITimeSpaceInput
«property»
+ Values() : ITimeSpaceValueSet
IIdentifiable
«interface»
IAdaptedOutputFactory
+ CreateAdaptedOutput(IIdentifiable, IBaseOutput, IBaseInput) : IBaseAdaptedOutput
+ GetAvailableAdaptedOutputIds(IBaseOutput, IBaseInput) : IIdentifiable[]

Figure: 17

When - *(Logical diagram)*

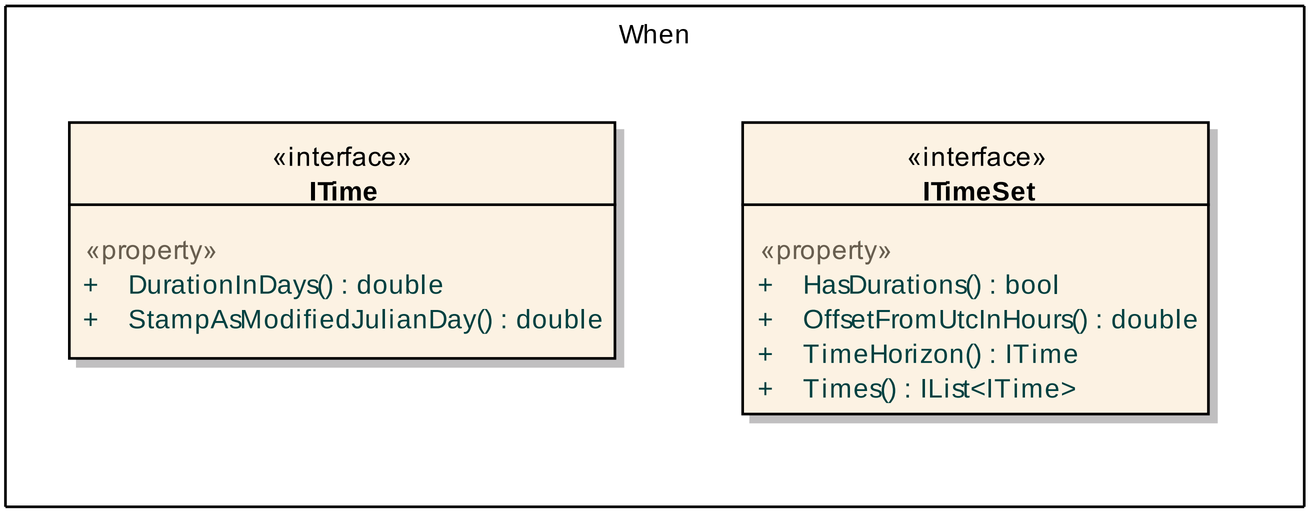


Figure: 18

Where - *(Logical diagram)*

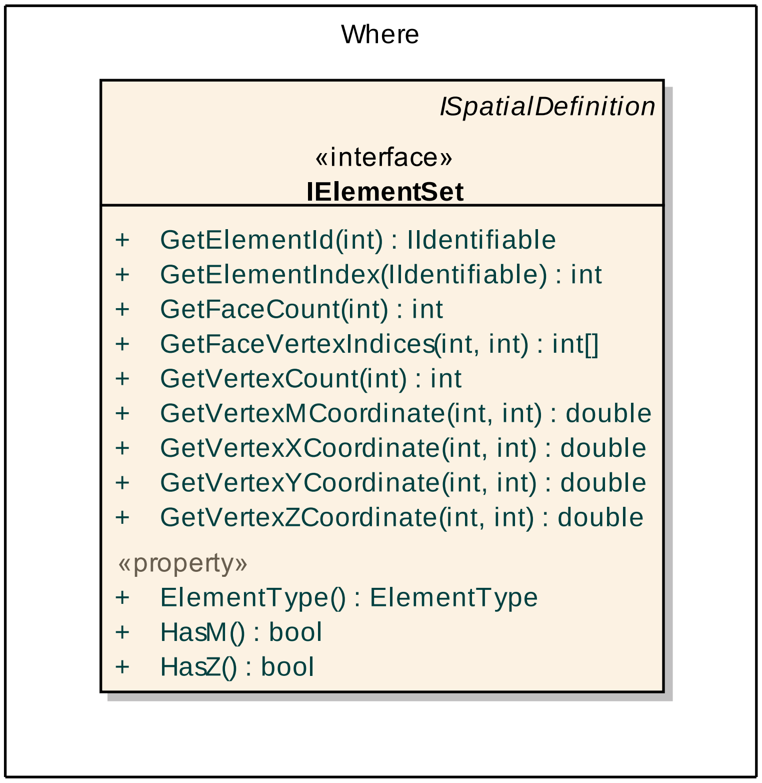

Where
ISpatialDefinition
«interface»
IElementSet
+ GetElementId(int) : IIdentifiable
+ GetElementIndex(IIdentifiable) : int
+ GetFaceCount(int) : int
+ GetFaceVertexIndices(int, int) : int[]
+ GetVertexCount(int) : int
+ GetVertexMCoordinate(int, int) : double
+ GetVertexXCoordinate(int, int) : double
+ GetVertexYCoordinate(int, int) : double
+ GetVertexZCoordinate(int, int) : double
«property»
+ ElementType() : ElementType
+ HasM() : bool
+ HasZ() : bool

Figure: 19

1.1 DimensionBase

Type: **Enumeration**

Enumeration for base dimensions

Custom Properties

- isActive = False

Attributes

Attribute	Notes	Constraints and tags
Length Public	Base dimension length.	*Default:*
Mass Public	Base dimension mass.	*Default:*
Time Public	Base dimension time.	*Default:*
ElectricCurrent Public	Base dimension electric current.	*Default:*

Attribute	Notes	Constraints and tags
Temperature Public	Base dimension temperature.	*Default:*
AmountOfSubstance Public	Base dimension amount of substance.	*Default:*
LuminousIntensity Public	Base dimension luminous intensity.	*Default:*
Currency Public	Base dimension currency.	*Default:*

1.2 ElementType

Type: **Enumeration**

Shape Type of elements in an "IElementSet"

Custom Properties

- isActive = False

Attributes

Attribute	Notes	Constraints and tags

Attribute	Notes	Constraints and tags
IdBased Public	Identifiable based	*Default:*
Point Public	Points	*Default:*
PolyLine Public	Lines / Polylines	*Default:*
Polygon Public	Polygons	*Default:*
Polyhedron Public	Polyhedrons	*Default:*

1.3 ExchangeItemChangeEventArgs

Type: **Class** **EventArgs**

The ExchangeItemChangeEventArgs contains the information that will be passed when the "IBaseExchangeItem"fires the ExchangeItemValueChanged event.
Sending exchange item events is optional, so it should not be used as a mechanism to build critical functionality upon.

Custom Properties

- isActive = False

Operations

Method	**Notes**	**Parameters**
ExchangeItem() IBaseExchangeItem Public	The exchange item of which the status has been changed.	
ExchangeItemChangeEventArgs() Public	Default constructor. Creates a new instance with an empty message and null as exchangeItem. Properties need to be set before actually using the instance.	
ExchangeItemChangeEventArgs() Public	Constructor that also initializes the "ExchangeItem"and the "Message"property	**IBaseExchangeItem** [in] exchangeItem **string** [in] message
Message() string Public	A message that describes the way in which the status of the exchange item has been changed.	

1.4 LinkableComponentStatus

Type: **Enumeration**

The LinkableComponentStatus enumerates the possible statuses that a linkable component can be in. The state diagram showing the possible statuses and the transitions from one status to another can be found in the documentation for OpenMI 2.0 on "http://www.openmi.org" .
They are also mentioned in the documentation of the various methods of the "IBaseLinkableComponent" .

Custom Properties

- isActive = False

Attributes

Attribute	**Notes**	**Constraints and tags**

Attribute	**Notes**	**Constraints and tags**
Created Public	The linkable component instance has just been Created . This status must and will be followed by "Initializing" .	*Default:*
Initializing Public	The linkable component is initializing itself. This status will end in a status change to "Initialized"or "Failed" .	*Default:*
Initialized Public	The linkable component has succesfully initialized itself. The connections between its inputs/outputs and those of other components can be established.	*Default:*
Validating Public	After links between the component's inputs/outputs and those of other components have been established, the component is validating whether its required input will be available when it updates itself, and whether indeed it will be able to provide the required output during this update. This Validating status will end in a status change to "Valid"or "Invalid" .	*Default:*
Valid Public	The component is in a valid state. When updating itself its required input will be available, and it it will be able to provide the required output.	*Default:*
WaitingForData Public	The component wants to update itself, but is not yet able to perform the actual computation, because it is still waiting for input data from other components.	*Default:*

Attribute	**Notes**	**Constraints and tags**
Invalid Public	The component is in an invalid state. When updating itself not all required input will be available, and/or it will not be able to provide the required output. After the user has modified the connections between the component's inputs/outputs and those of other components, the "Validating"state can be entered again.	*Default:*
Preparing Public	The component is preparing itself for the first GetValues() call. This Preparing state will end in a status change to "Updated"or "Failed" .	*Default:*
Updating Public	The component is updating itself. It has received all required input data from other components, and is now performing the actual computation. This Updating state will end in a status change to "Updated" , "Done"or "Failed" .	*Default:*
Updated Public	The component has succesfully updated itself.	*Default:*
Done Public	The last update process that the component performed was the final one. A next call to the Update method will leave the component's internal state unchanged.	*Default:*
Finishing Public	The ILinkableComponent was requested to perform the actions to be performed before it will either be disposed or re-intialized again. Typical actions would be writing the final result files, close all open files, free memory, etc. When all required actions have been performed, the status switches to "Created" when re-initialization is possible. The status switches to "Finished" when the component is to be disposed.	*Default:*

Attribute	Notes	Constraints and tags
Finished Public	The ILinkableComponent has successfully performed its finalization actions. Re-initialization of the component instance is not possible and should not be attempted. Instead the instance should be disposed, e.g. through the garbage collection mechanism.	*Default:*
Failed Public	@see ="Created" if the component supports being re-initialized. If it cannot be re-initialized, it can be released from memory. <summary> The linkable component has failed initialize itself, failed to prepare itself for computation, or failed to complete its update process.	*Default:*

1.5 LinkableComponentStatusChangeEventArgs

Type: **Class** **EventArgs**

The LinkableComponentStatusChangeEventArgs contains the information that will be passed when the "IBaseLinkableComponent"fires the StatusChanged event.

Custom Properties

- isActive = False

Operations

Method	Notes	Parameters
LinkableComponent() IBaseLinkableComponent Public	The linkable component that fired the status change event.	
LinkableComponentStatusChangeEventArgs() Public	Constructor.	
Message() string Public	A message providing additional information on the status change. If there is no message, an empty string is returned.	
NewStatus() LinkableComponentStatus Public	The linkable component's status after the status change.	
OldStatus() LinkableComponentStatus Public	The linkable component's status before the status change.	

1.6 IAdaptedOutputFactory

Type: **Interface** **IIdentifiable**

Used to create instances of "IBaseAdaptedOutput"items.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IAdaptedOutputFactory	Public IIdentifiable	

Operations

Method	Notes	Parameters
CreateAdaptedOutput() IBaseAdaptedOutput Public	Creates a "IBaseAdaptedOutput"that adapts the "adaptee"so that it fits the target. The adaptedOutputId used must be one of the IIdentifiable instances returned by the "GetAvailableAdaptedOutputIds"method. The returned "IBaseAdaptedOutput"will already be registered with the "adaptee" . @returns	**IIdentifiable** [in] adaptedOutputId The identifier of the adaptedOutput to create. **IBaseOutput** [in] adaptee "IBaseOutput"to adapt. **IBaseInput** [in] target "IBaseInput"to adapt the adaptee to, can be null .
GetAvailableAdaptedOutputIds() IIdentifiable Public	Get a list of identifiers of the available "IBaseAdaptedOutput" s that can make the "adaptee"match the "target" . If the "target" is null , the identifiers of all "IBaseAdaptedOutput" s that can adapt the "adaptee"are returned. @returns List of identifiers for the available "IBaseAdaptedOutput" s.	**IBaseOutput** [in] adaptee "IBaseOutput"to adapt. **IBaseInput** [in] target "IBaseInput"to adapt the adaptee to, can be null .

1.6.1 IArgument

Type: **Interface** **IIdentifiable**

The IArgument interface is used to set the arguments of a "IBaseLinkableComponent"and the arguments of an "IBaseAdaptedOutput"

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IArgument	Public IIdentifiable	

Connector	Source	Target	Notes

Operations

Method	Notes	Parameters
DefaultValue() object Public	The default value of the argument.	
IsOptional() bool Public	Specifies whether the argument is optional or not. If the Values property returns null and IsOptional == false , a value has to be set before the argument can be used.	
IsReadOnly() bool Public	Defines whether the Values property may be edited. This is used to let a "IBaseLinkableComponent"or an "IBaseAdaptedOutput"present the actual value of an argument that can not be changed by the user, but is needed to determine the values of other arguments or is informative in any other way.	
PossibleValues() IList<object> Public	List of possible allowed values for this argument. If for integral types or component specific types all possible values are allowed, null is returned. A list with length 0 indicates that there is indeed a limitation on the possible values, but that currently no values are possible. Effectively this means that the values will not and cannot be set.	
Value() object Public	The current value of the argument. If no value has been set yet, a default value is returned. If null is returned, this means that the default value is null .	
ValueAsString() string Public	The argument's value, represented as a string. If ValueType indicates that the argument's value is not of the type string, the ValueAsString function offers the possibility to treat it as a string, e.g. to let the GUI persist the value in the composition file.	
ValueType() Type Public	The type of the value of the argument, E.g. a integral type like string, integer or double, or a non integral type, such as a time series object.	

1.7 IArgument

Type: **Interface IIdentifiable**

The IArgument interface is used to set the arguments of a "IBaseLinkableComponent"and the arguments of an "IBaseAdaptedOutput"

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IArgument	Public IIdentifiable	

Operations

Method	Notes	Parameters
DefaultValue() object Public	The default value of the argument.	
IsOptional() bool Public	Specifies whether the argument is optional or not. If the Values property returns null and IsOptional == false , a value has to be set before the argument can be used.	
IsReadOnly() bool Public	Defines whether the Values property may be edited. This is used to let a "IBaseLinkableComponent"or an "IBaseAdaptedOutput"present the actual value of an argument that can not be changed by the user, but is needed to determine the values of other arguments or is informative in any other way.	
PossibleValues() IList<object> Public	List of possible allowed values for this argument. If for integral types or component specific types all possible values are allowed, null is returned. A list with length 0 indicates that there is indeed a limitation on the possible values, but that currently no values are possible. Effectively this means that the values will not and cannot be set.	
Value() object Public	The current value of the argument. If no value has been set yet, a default value is returned. If null is returned, this means that the default value is null .	
ValueAsString() string Public	The argument's value, represented as a string. If ValueType indicates that the argument's value is not of the type string, the ValueAsString function offers the possibility to treat it as a string, e.g. to let the GUI persist the value in the composition file.	
ValueType() Type Public	The type of the value of the argument, E.g. a integral type like string, integer or double, or a non integral type, such as a time series object.	

1.8 IBaseAdaptedOutput

Type: **Interface IBaseOutput**

An "IBaseAdaptedOutput"adds one or more data operations on top of an output item. It is in itself an "IBaseOutput" . The adaptedOutput extends an output item with functionality as spatial interpolation, temporal interpolation, unit conversion etc.

"IBaseAdaptedOutput"instances are created by means of an "IAdaptedOutputFactory" .

The "IBaseAdaptedOutput"is based on the adaptor design pattern. It adapts an "IBaseOutput"or another "IBaseAdaptedOutput"to make it suitable for new use or purpose. The object being adapted is typically called the "adaptee". The "IBaseAdaptedOutput"replaces the DataOperation that was used in OpenMI Standard version 1.x.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IBaseAdaptedOutput	Public IBaseOutput	
Generalization Source -> Destination	Public ITimeSpaceAdaptedOutput	Public IBaseAdaptedOutput	

Operations

Method	Notes	Parameters
Adaptee() IBaseOutput Public	Output item that this adaptedOutput extracts content from. In the adapter design pattern, it is the item being adapted.	
Arguments() IList<IArgument> Public	Arguments needed to let the adapted output do its work. An unmodifiable list of the (modifiable) arguments should be returned that can be used to get info on the arguments and to modify argument values. Validation of changes is done when they occur (e.g. using notifications). <returns>Unmodifiable list of IArgument for the adapted output</returns>	
Initialize() void Public	Let the adapted output initialize itself, based on the current values specified by the arguments. Only after initialize is called the refresh method might be called. A component must invoke the "Initialize()"method of all its adapted outputs at the end of the component's Prepare phase. In case of stacked adapted outputs, the adaptee must be initialized first.	
Refresh() void Public	Request the adapted output to refresh itself. This method will be called by the adaptee,	

Method	Notes	Parameters
	when it has been refreshed/updated. In the implementation of the refresh method the adapted output should update its contents according to the changes in the adaptee. After updating itself the adapted output must call refresh on all its adapted outputs, so the chain of outputs refreshes itself.	

1.9 IBaseExchangeItem

Type: **Interface** **IIdentifiable**

An item that can be exchanged, either as input or as output.

@remark This interface is not to be implemented directly, any class is to implement either the "IBaseInput"or "IBaseOutput" .

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IBaseExchangeItem	Public IIdentifiable	
Generalization Source -> Destination	Public IBaseInput	Public IBaseExchangeItem	
Generalization Source -> Destination	Public IBaseOutput	Public IBaseExchangeItem	
Generalization Source -> Destination	Public ITimeSpaceExchangeItem	Public IBaseExchangeItem	

Operations

Method	Notes	Parameters
Component() IBaseLinkableComponent Public	Gets the owner of the exchange item. For an output exchange item this is the component responsible for providing the content of the output item. It is possible for an exchange item to have no owner, in this case the method will return null.	
ItemChanged() EventHandler<ExchangeItemChangeEventArgs> Public	The ItemChanged event is fired when the content of an exchange item has changed. This might be because its ValueDefinition has changed, its TimeSet has changed, its ElementSet has changed, its Values have changed, or any permutation of these properties.	

Method	Notes	Parameters
ValueDefinition() IValueDefinition Public	Definition of the values in the exchange item. @remark The "IValueDefinition"should never be returned directly; all implementing classes should return either an "IQuality" , an "IQuantity" , or a custom derived vale definition interface.	

1.10 IBaseInput

Type: **Interface** **IBaseExchangeItem**

An input item that can accept values for an "IBaseLinkableComponent" .

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IBaseInput	Public IBaseExchangeItem	
Realisation Source -> Destination	Public ITimeSpaceInput	Public IBaseInput	

Operations

Method	Notes	Parameters
Provider() IBaseOutput Public	The provider this input should get its values from.	
Values() IBaseValueSet Public	The exchange item's values.	

1.11 IBaseLinkableComponent

Type: **Interface** **IIdentifiable**

The IBaseLinkableComponent is the key interface in the OpenMI standard.
OpenMI-compliance definition (The compliancy refers to a set of basic interfaces as well as to optional extension interfaces, e.g. for time and space dependent components):
§ 1) An OpenMI-compliant component must implement the IBaseLinkableComponent interface according to specifications provided as comments in the OpenMI.Standard2 source code.
§ 2) An OpenMI compliant component can also comply to one ore more extensions, by implementing the IBaseLinkableComponent interface and the extension interfaces which it wishes to comply to, according to the specifications provided as comments in the OpenMI.Standard2 source code.
§ 3) An OpenMI-compliant component including its extensions must, when compiled, reference the OpenMI.Standard2*.dlls/jars, which are compiled and and released by the OpenMI Association.
§ 4) An OpenMI-compliant component must be associated with an XML file, the so called OMI file, which

complies to (can be validated with) the LinkableComponent.xsd schema.
§ 5) An OpenMI-compliant component must be associated with an XML file, the so called compliancy info file, which complies to (can be validated with) the OpenMICompliancyInfo.xsd schema. This file must be submitted to the OpenMI Association.
§ 6) The OpenMI Association provides two additional interfaces that OpenMI-compliant components may or may not implement: the "IManageState"interface and the "IByteStateConverter"interface. However, if these interfaces are implemented, each method and property must be implemented according to the comments given in the OpenMI.Standard2 source code.
§ 7) The OpenMI Association's downloadable standard zip file provides the only recognized version of source files, XML schemas and assembly files.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IBaseLinkableComponent	Public IIdentifiable	
Generalization Source -> Destination	Public ITimeSpaceComponent	Public IBaseLinkableComponent	

Operations

Method	Notes	Parameters
AdaptedOutputFactories() List<IAdaptedOutputFactory> Public	Get a list of "IAdaptedOutputFactory" , each allowing to create "IBaseAdaptedOutput"item for making outputs fit to inputs in case they do not already do so. Factories can be added to and removed from the list so that third-party factories and IBaseAdaptedOutput classes can be introduced.	
Arguments() IList<IArgument> Public	Arguments needed to let the component do its work. An unmodifiable list of (modifiable) arguments must be returned that is to be used to get information about the arguments and to set argument values. Validation of changes can be done either when they occur (e.g. using notifications) or when the initialize method is called. Initialize will always be called before any call to the update method of the IBaseLinkableComponent. This property must be available as soon is the linkable component instance is created. Arguments describes the arguments that can be set before the Initialize() method is called.	
Finish() void Public	This method is and must be invoked as the last of any methods in the ILinkableComponent interface. This method must be accessible after the	

Method	Notes	Parameters
	"Prepare"method has been invoked. If this method is invoked before the "Prepare"method has been invoked and the LinkableComponent cannot handled this, an exception must be thrown. Immediatly after the method is invoked, it changes the linkable component's Status to "LinkableComponentStatus.Finishing" . Once the finishing is completed, the component changes its status to "LinkableComponentStatus.Finished"if it can not be restarted, or "LinkableComponentStatus.Created"if it can.	
Initialize() void Public	Initializes the LinkableComponent. The "Initialize()"method will and must be invoked before any other method or property in the ILinkableComponent interface is invoked or accessed, except for the "Arguments"property. Immediatly after the method is invoked, it changes the linkable component's Status to "LinkableComponentStatus.Initializing" . When the method is executed and an error occurs, the Status of the component will change to "LinkableComponentStatus.Failed" , and an exception will be thrown. If the component initializes succesfully, the status is changed to "LinkableComponentStatus.Initialized" . When the "Initialize()"method has been finished and the Status is "LinkableComponentStatus.Initialized" , the properties Id, Caption, Description, "Inputs" , "Outputs" , have been set, and the method "Validate"can be called. It is only required that the method "Initialize()"can be invoked once. If the "Initialize()"method is invoked more than once and the LinkableComponent cannot handle this; an exception must be thrown. REMARKS: The method will typically populate the component based on the values specified in its arguments, which can be retrieved with getArguments. Settings can be used to read input files, allocate memory, and organize input and output exchange items.	
Inputs() IList<IBaseInput> Public	The list of input items for which a component can recieve values. @remark This property must be accessible after the "Initialize()"method has been invoked and until the "Validate"method has been invoked. If this property is accessed before the	

Method	Notes	Parameters
	"Initialize()"method has been invoked or after the "Validate"method has been invoked and the LinkableComponent cannot handle this an exception must be thrown. This method basically returns references to "IBaseInput"items. There is no guarantee that the list of objects is not altered by other components after it has been returned. It is the responsibility of the LinkableComponent to make sure that such possible alterations do not subsequently corrupt the LinkableComponent.	
Outputs() IList<IBaseOutput> Public	The list of output items for which a component can produce results. @remark This property must be accessible after the "Initialize()"method has been invoked and until the "Validate"method has been invoked. If this property is accessed before the "Initialize()"method has been invoked or after the "Validate"method has been invoked and the LinkableComponent cannot handle this an exception must be thrown. The list only contains the core IBaseOutput of the component, not the IBaseAdaptedOutput derived from each IBaseOutput (etc.). To get a complete list of outputs traverse the chain of IBaseAdaptedOutput that start with the IOutputs returned in the list. This method basically returns references to "IBaseOutput"items. There is no guarantee that the list of objects is not altered by other components after it has been returned. It is the responsibility of the LinkableComponent to make sure that such possible alterations do not subsequently corrupt the LinkableComponent.	
Prepare() void Public	Prepares the IBaseLinkableComponent for calls to the "Update"method Before Prepare is called, the component are not required to honor any type of action that retrieves values from the component. After Prepare is called, the component must be ready for providing values. This method must be accessible after the "Initialize()"method has been invoked and until the "Finish"method has been invoked. If this property is accessed before the "Initialize()"method has been invoked or after the "Finish"method has been invoked and the	

Method	Notes	Parameters
	LinkableComponent cannot handle this an exception must be thrown. Immediatly after the method is invoked, it changes the linkable component's Status to "LinkableComponentStatus.Preparing" . When the method has finished, the Status of the component has changed to either "LinkableComponentStatus.Updated"or "LinkableComponentStatus.Failed" . It is only required that the Prepare( ) method can be invoked once. If the Prepare method is invoked more that once and the LinkableComponent cannot handle this an exception must be thrown.	
Status() LinkableComponentStatus Public	Defines current status of the linkable component. See "LinkableComponentStatus"for the possible values. The first Status that a component sets is "LinkableComponentStatus.Created" , as soon after it has been created. In this Status, "Arguments"is the only property that may be accessed.	
StatusChanged() EventHandler<LinkableComponentStatusChangeEventArgs> Public	The StatusChanged event is fired when Status of the component changes. See "LinkableComponentStatus"for the possible states, and see the documentation, e.g. the OpenMI Standard 2 Specification, for possible state changes.	
Update() void Public	This method is called to let the component update itself, thus reaching its next state. Immediately after the method is invoked, it changes the linkable component's Status to "LinkableComponentStatus.Updating" . The type of actions a component takes during the "Update"method depends on the type of component. A numerical model that progresses in time will typically compute a time step. A database whould typically look at the consumers of its output items, and perform one or more queries to be able to provide the values that the consumers require. A GIS system would typically re-evaluate the values in a grid coverage, so that its output output items can provide up-to-date values. If the Update method is performed succesfully, the component sets its state to "LinkableComponentStatus.Updated" , unless after this Update action the component is at the end of its computation, in which case it will be set its State to "LinkableComponentStatus.Done" .	**IBaseOutput[]** [in] requiredOutput This optional parameter lets the caller specify the specific output items that should be updated. If it is omitted or if the length is 0, the component will at least update its output items that have consumers, or all its output items, depending on the component's implementation.

Method	Notes	Parameters
	If during the Update method a problem arises, the component sets its state to "LinkableComponentStatus.Failed" , and throws an exception.	
Validate() string Public	Validates the populated instance of the LinkableComponent This method must be accessible after the "Initialize()"method has been invoked and until the "Finish"method has been invoked. If this property is accessed before the "Initialize()"method has been invoked or after the "Finish"method has been invoked and the LinkableComponent cannot handle this an exception must be thrown. The method will and must be invoked after the various provider/consumer relations between this component's exchange items and the exchange items of other components present in the composition. Immediatly after the method is invoked, it changes the linkable component's Status to "LinkableComponentStatus.Validating" . When the Validate method has finished, the Status of the component has changed to either "LinkableComponentStatus.Valid"or "LinkableComponentStatus.Invalid" . @returns Returns null or an array of strings of length null if there are no messages at all. If there are messages while the components Status is "LinkableComponentStatus.Valid" , the messages are purely informative. If there are messages while the components Status is "LinkableComponentStatus.Invalid" , at least one of the messages indicates a fatal error.	

1.12 IBaseOutput

Type: **Interface** **IBaseExchangeItem**

An output exchange item that can deliver values from an "IBaseLinkableComponent" .
If an output does not provide the data in the way a consumer would like to have it the output can be adapted by an "IBaseAdaptedOutput" , which can transform the data according to the consumer's wishes. E.g. by performing interpolation in time, spatial aggregation, etc.).

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IBaseAdaptedOutput	Public IBaseOutput	
Generalization Source -> Destination	Public IBaseOutput	Public IBaseExchangeItem	
Realisation Source -> Destination	Public ITimeSpaceOutput	Public IBaseOutput	

Operations

Method	Notes	Parameters
AdaptedOutputs() IList<IBaseAdaptedOutput > Public	The adaptedOutputs that have this current output item as Adaptee . As soon as the output item's values have been updated, for each adaptedOutput its "IBaseAdaptedOutput.Refresh"method must be called. The list is readonly. Add and remov from the list by using "AddAdaptedOutput"and "RemoveAdaptedOutput" .	
AddAdaptedOutput() void Public	Add a "IBaseAdaptedOutput"to this output item. Every adaptedOutput that uses data from this output item, needs to add itself as a consumer first. If a adaptedOutput is added that can not be handled, or that is incompatible with the already added adaptedOutputs, an exception will be thrown. "adaptedOutput” consumer that has to be added	**IBaseAdaptedOutput** [in] adaptedOutput
AddConsumer() void Public	Add a consumer to this output item. Every input item that wants to call the GetValues() method, needs to add itself as a consumer first. If a consumer is added that can not be handled, or that is incompatible with the already added consumers, an exception will be thrown. The AddConsumer method must and will automatically set the consumer's Provider (see "IBaseInput.Provider") "consumer” consumer that has to be added	**IBaseInput** [in] consumer
Consumers() IList<IBaseInput> Public	Input items that will consume the values, by calling the GetValues() method. Every input item that will call this method, needs to call the AddConsumer method first. If the input item is not interested any longer in calling the GetValues, it should remove itself by calling the RemoveConsumer method. The list is readonly. Add and remove from the list by using "AddConsumer"and "RemoveConsumer" .	

Method	Notes	Parameters
	@remark Please be aware that the "unadulterated" values in the output item, provided by the read only Values property, may be called anyway, even if there are no values available.	
GetValues() IBaseValueSet Public	Provides the values matching the value definition specified by the "querySpecifier” . Extensions can overwrite this base version to include more details in the query, e.g. time and space. @remark One might expect to be the querySpecifier to be of the type IInput, because every input item that calls the GetValues method needs to add itself as a consumer first. However, the "IBaseExchangeItem"suffices to specify what is required. Therefore, to have the flexibility to loosen the "always register as consumer" approach, it is chosen to provide an "IBaseExchangeItem"as argument.	**IBaseExchangeItem** [in] querySpecifier
RemoveAdaptedOutput() void Public	Remove a "IBaseAdaptedOutput" . If a adaptedOutput is not interested any longer in this output item data, it should remove itself by calling RemoveConsumer. "adaptedOutput” consumer that has to be removed	**IBaseAdaptedOutput** [in] adaptedOutput
RemoveConsumer() void Public	Remove a consumer. If an input item is not interested any longer in calling the GetValues method, it should remove itself by calling RemoveConsumer. "consumer” consumer that has to be removed	**IBaseInput** [in] consumer
Values() IBaseValueSet Public	The exchange item's values	

1.13 IBaseValueSet

Type: **Interface**

The "IBaseValueSet"represents a general multi-dimensional set of values. Each value is of type "ValueType"
The size of each dimension can vary, depending on the indices provided, e.g. in a 2D matrix each row can have different lengths. Example, assuming the data is stored as a double[][] matrix, then matrix[1].Length need not equal matrix[2].Length.

Connections

Connector	Source	Target	Notes

Connector	Source	Target	Notes
Generalization Source -> Destination	Public ITimeSpaceValueSet	Public IBaseValueSet	

Operations

Method	Notes	Parameters
GetIndexCount() int Public	Returns the length (max index count) of the dimension specified by the given indices. To get the size of the first dimension, use a zero-length integer array as input argument. Length of indices must be a least one smaller than the "NumberOfIndices" @returns length of the specified dimension	**int[]** [in] indices indices of the dimension to get the length of
GetValue() Object Public	Returns the value object specified by the given array of indices. The length of the array of indices is N, so that the index for each dimension is specified. Otherwise an IllegalArgumentException must be thrown. @returns the value object for the given indices	**int[]** [in] indices indices index value for each dimension
NumberOfIndices() int Public	Returns the number of possible indices (dimensions) for the value set. @returns number of indices, zero based	
SetValue() void Public	Set the value object specified by the given array of indices. The length of the array of indices is N, so that the index for each dimension is specified. Otherwise an IllegalArgumentException must be thrown.	**int[]** [in] indices indices index value for each dimension **Object** [in] value value the value object for the given indices
ValueType() Type Public	The object type of the values that will be available in the value set that is returned by the Values property and the GetValues function.	

1.14 IByteStateConverter

Type: **Interface**

This interface is an optional complement to the "IManageState"interface. Both are extensions to "IBaseLinkableComponent" , meant to provide state management. It defines methods for converting a state into a byte stream and reading in a state from a byte stream. This facilitates external modules, e.g. a GUI or an operational control system, to save a model's state somewhere as persistent state.

Operations

Method	Notes	Parameters
ConvertFromByteArray( ) IIdentifiable Public	Creates a state from a byte stream and returns the identifier of this state. <remarks>The state does not become the current state of the "IBaseLinkableComponent" . For state management the "IManageState"interface is to be used.</remarks> @returns "IIdentifiable"identifying the state.	**byte[]** [in] byteArray State as a byte stream.
ConvertToByteArray() byte Public	Converts the state with the "stateId"into a byte stream. @returns The state identified by "stateId"as an array of bytes.	**IIdentifiable** [in] stateId id of the state.

1.15 ICategory

Type: **Interface** **IDescribable**

The ICategory describes one item of a possible categorization. It is used by the "IQuality"interface for describing qualitative data.
For qualitative data the "IBaseValueSet"exchanged between "IBaseLinkableComponent" s contains one of the possible ICategory instances per data element.

A category defines one "class" within a "set of classes".

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public ICategory	Public IDescribable	

Operations

Method	Notes	Parameters
Value() object Public	Value for this category. @example "blue" in a "red"/"green"/"blue" set.	

1.16 IDescribable

Type: **Interface**

Provides descriptive information on an OpenMI entity.
An entity that is describable has a caption (title or heading) and a description. These are not to be used for

identification (see "IIdentifiable").

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IUnit	Public IDescribable	
Generalization Source -> Destination	Public ICategory	Public IDescribable	
Generalization Source -> Destination	Public IIdentifiable	Public IDescribable	
Generalization Source -> Destination	Public ISpatialDefinition	Public IDescribable	
Generalization Source -> Destination	Public IValueDefinition	Public IDescribable	

Operations

Method	Notes	Parameters
Caption() string Public	Caption string (not to be used as an id)	
Description() string Public	Additional descriptive information about the entity.	

1.17 IDimension

Type: **Interface**

Defines the order of dimension in each "DimensionBase"for a unit

Operations

Method	Notes	Parameters
GetPower() double Public	Returns the power for the requested dimension @example For a quantity such as flow, which may have the unit m3/s, the GetPower method must work as follows: myDimension.GetPower(DimensionBase.AmountOfSubstance) --> returns 0 myDimension.GetPower(DimensionBase.Currency) --> returns 0 myDimension.GetPower(DimensionBase.ElectricCurrent) --> returns 0	**DimensionBase** [in] baseQuantity

Method	Notes	Parameters
	myDimension.GetPower(DimensionBase.Length) --> returns 3 myDimension.GetPower(DimensionBase.LuminousIntensity) --> returns 0 myDimension.GetPower(DimensionBase.Mass) --> returns 0 myDimension.GetPower(DimensionBase.Temperature) --> returns 0 myDimension.GetPower(DimensionBase.Time) --> returns -1	

1.18 IElementSet

Type: **Interface** **ISpatialDefinition**

Data exchange between components in OpenMI is nearly always related to one or more elements in a space, either geo-referenced or not. An element set in OpenMI can be a list of 2D or 3D spatial elements or, as a special case, a list of ID based (non spatial) elements. The latter is supported to allow the exchange of arbitrary data that is not related to space in any way. Possible element types are defined in "ElementType" .
An IElementSet is composed of an ordered list of elements having a common type. The geometry of each element is described by an ordered list of vertices. For 3D elements (i.e. polyhedrons) the shape can be queried by face. When the element set is geo-referenced co-ordinates (X,Y,Z,M) can be obtained for each vertex of an element.

A geo-referenced element set needs to have a valid SpatialReferenceSystemWkt property set in the "ISpatialDefinition" . This is a string that specifies the OGC Well-Known Text representation of the spatial reference. An empty string indicates that there in no spatial reference, which is only valid if the ElementType is ID_BASED.

While an IElementSet can be used to query the geometric description of a model schematization, it does not necessarily provide all topological knowledge on inter-element connections.

Although most models encapsulate static element sets, some advanced models might contain dynamic elements (e.g. waves). A version number has been introduced to enable tracking of element set changes over time. If the version changes, the element set might need to be queried again during the computation process.

@remark For ElementSets of type ElementType.IdBased the SpatialReferenceSystemWkt property an empty string.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IElementSet	Public ISpatialDefinition	

Operations

Method	Notes	Parameters
ElementType() ElementType Public	"ElementType"of the elementset. All elements in the set are of his type.	
GetElementId()	Returns Id of the ' "index" -th' element in the	**int** [in] index

Method	Notes	Parameters
IIdentifiable Public	ElementSet. Indexes start from zero. If the ElementType of the ElementSet is not IdBased, a null or an empty string may be returned. @returns Index of the element with the specified id, -1 if the id was not found	The element index for which the element Caption is requested. If the element index is outside the range [0, number of elements -1], an exception must be thrown.
GetElementIndex() int Public	Index of element with id "elementId"in the elementset. Indexes start from zero. There are not restrictions to how elements are ordered.	**IIdentifiable** [in] elementId Identification string for the element for which the element index is requested. If no element in the ElementSet has the specified elementId, -1 must be returned.
GetFaceCount() int Public	Returns the number of faces in a 3D element. For 2D elements this returns 0. @returns Number of faces.	**int** [in] elementIndex <para>Index for the element <para>If the element index is outside the range [0, number of elements -1], an exception must be thrown.
GetFaceVertexIndices() int Public	Gives an array with the vertex indices for a face. @returns The vertex indices for this face. @remark The vertex indices for a face must be locally numbered for the element (containing numbers in the range [0;"GetVertexCount" (elementIndex)-1]).	**int** [in] elementIndex Element index. **int** [in] faceIndex Face index.
GetVertexCount() int Public	Number of vertices for the element specified by the elementIndex. If the GetVertexCount()method is invoked for element sets of type "TimeSpace.ElementType.IdBased" , an exception must be thrown. @returns Number of vertices in element defined by the elementIndex.	**int** [in] elementIndex <para>The element index for the element for which the number of vertices is requested. <para>If the element index is outside the range [0, number of elements -1], an exception must be thrown.
GetVertexMCoordinate() double Public	M co-ordinate for the vertex with VertexIndex of the element with elementIndex.	**int** [in] elementIndex Element index. **int** [in] vertexIndex Vertex index in the element with index elementIndex.
GetVertexXCoordinate() double Public	X co-ordinate for the vertex with vertexIndex of the element with elementIndex.	**int** [in] elementIndex Element index. **int** [in] vertexIndex Vertex index in the element with index elementIndex.
GetVertexYCoordinate() double Public	Y co-ordinate for the vertex with vertexIndex of the element with elementIndex.	**int** [in] elementIndex Element index. **int** [in] vertexIndex

Method	Notes	Parameters
		Vertex index in the element with index elementIndex.
GetVertexZCoordinate() double Public	Z co-ordinate for the vertex with vertexIndex of the element with elementIndex.	**int** [in] elementIndex Element index. **int** [in] vertexIndex Vertex index in the element with index elementIndex.
HasM() bool Public	True if the element set supports M co-ordinates.	
HasZ() bool Public	True if the element set supports Z co-ordinates.	

1.19 IIdentifiable

Type: **Interface** **IDescribable**

Defines a method to get the Id of an OpenMI entity. The "IIdentifiable"extends the "IDescribable" , and therefore has, next to the id, a caption and a description.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IArgument	Public IIdentifiable	
Generalization Source -> Destination	Public IBaseExchangeItem	Public IIdentifiable	
Generalization Source -> Destination	Public IBaseLinkableComponent	Public IIdentifiable	
Generalization Source -> Destination	Public IIdentifiable	Public IDescribable	
Generalization Source -> Destination	Public IAdaptedOutputFactory	Public IIdentifiable	
Generalization Source -> Destination	Public IArgument	Public IIdentifiable	

Operations

Method	Notes	Parameters
Id() string Public	Returns the Id as a String. The Id must be unique within its context but does not need to be globally unique. E.g. the id of an input	

Method	Notes	Parameters
	exchange item must be unique in the list of inputs of a ILinkableComponent, but a similar Id might be used by an exchange item of another ILinkableComponent.	

1.20 IManageState

Type: **Interface**

Optional interface to be implemented by components in addition to the "IBaseLinkableComponent"interface. It provides additional methods for handling component state so it can be saved, restored and cleared. It can be left completely to the component to handle persistence of state or it can also implement "IByteStateConverter"and provide ways for state to be converted to and from an array of bytes. A third-party could then handle the saving and loading of state data.

Operations

Method	Notes	Parameters
ClearState() void Public	Clears a state from the linkable component's memory. If the state identifier identified by stateID is not known by the linkable component an IllegalArgumentException exception should be thrown. @code ArgumentException @paramref ="stateId" is specified. </remark	**IIdentifiable** [in] stateId Identifier of the state to be cleared.
KeepCurrentState() IIdentifiable Public	Store the linkable component's current State @returns Identifier of the stored state.	
RestoreState() void Public	Restores the state identified by the parameter stateID. If the state identifier identified by stateID is not known by the linkable component an IllegalArgumentException exception should be trown. @code ArgumentException @paramref ="stateId" is specified. </remark	**IIdentifiable** [in] stateId Identifier of the state to be restored.

1.21 IQuality

Type: **Interface** **IValueDefinition**

Qualitative data described items in terms of some quality or categorization that may be 'informal' or may use relatively ill-defined characteristics such as warmth and flavour. However, qualitative data can include well-defined

aspects such as gender, nationality or commodity type. OpenMI defines the IQuality interface for working with qualitative data.
An IQuality describes qualitative data, where a value is specified as one category within a number of predefined (possible) categories. These categories can be ordered or not.
For qualitative data the IValueSet exchanged between ILinkableComponents contains one of the possible ICategory instances per element in the ElementSet involved.

<example> Examples:
<list>
•Colors: red, green, blue
•Land use: nature, recreation, industry, infrastructure
•Rating: worse, same, better
</list>
</example>

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IQuality	Public IValueDefinition	

Operations

Method	Notes	Parameters
Categories() IList<ICategory> Public	Returns a list of the possible "ICategory"allowed for this IQuality. If the quality is not ordered the list contains the ICategory's in an unspecified order. When it is ordered the list contains the ICategory's in the same sequence.	
IsOrdered() bool Public	Checks if the IQuality is defined by an ordered set of ICategory or not.	

1.22 IQuantity

Type: **Interface** **IValueDefinition**

A Quantity specifies values as an amount of some unit, usually as a floating point number.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IQuantity	Public IValueDefinition	

Operations

Method	Notes	Parameters
Unit() IUnit Public	Unit of quantity	

1.23 ISpatialDefinition

Type: **Interface** **IDescribable**

Data in components in OpenMI is often related to spatial coordinates, either geo-referenced or not. The "ISpatialDefinition"is the general spatial construct that all other spatial constructions extend from.
The currently most noticable extending interfaces is the "IElementSet" , which in previous versions of the standard was the only spatial construction, and which all other spatial constructions had to be wrapped into, whereas in the current version the "IElementSet"is an extension of the "ISpatialDefinition" .

Although most models encapsulate data with a static spatial definition, some advanced models might contain dynamic spatial definitions (e.g. waves, moving grids). The
"Version"number has been introduced to enable tracking of spatial changes over time. If the version changes, the spatial definition might need to be queried again during the computation process.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IElementSet	Public ISpatialDefinition	
Generalization Source -> Destination	Public ISpatialDefinition	Public IDescribable	

Operations

Method	Notes	Parameters
ElementCount() int Public	Number of data elements in the spatial axis.	
SpatialReferenceSystem Wkt() string Public	The SpatialReferenceSystemWkt speficies the OGC Well-Known Text representation of the spatial reference system to be used in association with the coordinates in the "ISpatialDefinition" . For the list of WKT strings see "http://spatialreference.org/" . @example For all spatial axis a spatial reference can be defined in a form: PROJCS["Mercator Spheric", GEOGCS["WGS84based_GCS", DATUM["WGS84based_Datum", SPHEROID["WGS84based_Sphere", 6378137, 0], TOWGS84[0, 0, 0, 0, 0, 0, 0]], PRIMEM["Greenwich", 0,	

Method	Notes	Parameters
	AUTHORITY["EPSG", "8901"]], UNIT["degree", 0.0174532925199433, AUTHORITY["EPSG", "9102"]], AXIS["E", EAST], AXIS["N", NORTH]], PROJECTION["Mercator"], PARAMETER["False_Easting", 0], PARAMETER["False_Northing", 0], PARAMETER["Central_Meridian", 0], PARAMETER["Latitude_of_origin", 0], UNIT["metre", 1, AUTHORITY["EPSG", "9001"]], AXIS["East", EAST], AXIS["North", NORTH]]	
Version() int Public	The current version number for the spatial axis. The version must be incremented if anything inside the spatial axis is changed, or if an entirely new spatial axis is provided.	

1.24 ITime

Type: **Interface**

Time interface based on a Modified Julian Date (number and fraction of days since 00:00 November 17, 1858).

The ITime interface supports a time stamp as well as a time interval. A time stamp will have its "DurationInDays"set to 0, while a time interval will have a positive "DurationInDays"value.

Operations

Method	Notes	Parameters
DurationInDays() double Public	Duration in days. O if time is a time stamp.	
StampAsModifiedJulianDay() double Public	Time stamp as a modified julian day value.	

1.25 ITimeSet

Type: **Interface**

A set of time stamps or time intervals, used to indicate where an output item has values and can provide values, and where an input item does or may require values.

The "HasDurations"defines whether the set contains stamps or intervals.

Operations

Method	Notes	Parameters
HasDurations() bool Public	True if the "Times"have durations, i.e. are time intervals. In this case, a duration value greater then zero is expected for every ITime in the "Times"list.	
OffsetFromUtcInHours() double Public	Time zone offset from UTC, expressed in the number of hours. Since some of the world's time zones differ half an hour from their neighbours the value is specified as a double.	
TimeHorizon() ITime Public	The time horizon defines for an input item for what time span it may require values. This means the providers of this input can assume that the input item never goes back further in time than the time horizon's begin time, TimeHorizon.StampAsModifiedJulianDay . Also, it will never go further ahead than the time horizon's end time, TimeHorizon.StampAsModifiedJulianDay+TimeHorizon.DurationInDays . For an output item, and thus for an adapted output item, the time horizon indicates in what time span the item can provide values. Specific values: TimeHorizon.StampAsModifiedJulianDay == Double.NegativeInfinity : far back in time TimeHorizon.Duration == Double.PositiveInfinity : far in the future	
Times() IList<ITime> Public	Time stamps or spans as available in the values of an output item, or as required by an input item. Specific values: TimeSet.Times.Count == 0 , in case of output: time dependent item, but no values available yet or required yet TimeSet.Times.Count == 0 , in case of in: time dependent item, but currently no values required	

1.26 ITimeSpaceAdaptedOutput

Type: **Interface** **IBaseAdaptedOutput**

An "ITimeSpaceAdaptedOutput"adds one or more data operations on top of an output item. It is in itself an "IBaseAdaptedOutput" . The adaptedOutput extends an output item with functionality as spatial interpolation, temporal interpolation, unit conversion etc.
ITimeSpaceAdaptedOutput instances are created by means of an "IAdaptedOutputFactory" .

The IAdaptedOutput is based on the adaptor design pattern. It adapts an IOutput or another IAdaptedOutput to make it suitable for new use or purpose. The object being adapted is typically called the "adaptee". The IAdaptedOutput replaces the DataOperation that was used in OpenMI 1.x.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public ITimeSpaceAdaptedOutput	Public IBaseAdaptedOutput	
Realisation Source -> Destination	Public ITimeSpaceAdaptedOutput	Public ITimeSpaceOutput	

1.27 ITimeSpaceComponent

Type: **Interface** **IBaseLinkableComponent, ITimeExtension**

An "IBaseLinkableComponent"providing exchange items of type time-space.
See "IBaseLinkableComponent"for details

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public ITimeSpaceComponent	Public IBaseLinkableComponent	

1.28 ITimeSpaceExchangeItem

Type: **Interface** **IBaseExchangeItem**

A time / space dependent item that can be exchanged, either as input or as output.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public ITimeSpaceExchangeItem	Public IBaseExchangeItem	
Generalization Source -> Destination	Public ITimeSpaceInput	Public ITimeSpaceExchangeItem	

Connector	Source	Target	Notes
Generalization Source -> Destination	Public ITimeSpaceOutput	Public ITimeSpaceExchangeItem	

Operations

Method	Notes	Parameters
SpatialDefinition() ISpatialDefinition Public	Spatial information (usually an element set) on the values that are available in an output exchange item, or required by an input exchange item.	
TimeSet() ITimeSet Public	Time information on the values that are available in an output exchange item, or required by an input exchange item	

1.29 ITimeSpaceExtension

Type: **Interface**

Methods that are specific for an time-space component.

Operations

Method	Notes	Parameters
TimeExtent() ITimeSet Public	The "TimeExtent"property describes in what time span the component can operate. This can be used to support the user when creating a composition.	

1.30 ITimeSpaceInput

Type: **Interface** **ITimeSpaceExchangeItem**

An input item that can accept values for an "ITimeSpaceComponent" .

The item is a combination of an "IValueDefinition" , an "IElementSet" , and an "ITimeSet" . This combination specifies which type of data is required, where and when, as input for an "ITimeSpaceComponent" .

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public ITimeSpaceInput	Public ITimeSpaceExchangeItem	

Connector	Source	Target	Notes
Realisation Source -> Destination	Public ITimeSpaceInput	Public IBaseInput	

Operations

Method	Notes	Parameters
Values() ITimeSpaceValueSet Public	The exchange item's values, as a specialized "ITimeSpaceValueSet"	

1.31 ITimeSpaceOutput

Type: **Interface** **ITimeSpaceExchangeItem**

An output exchange item that can deliver values from a time / space dependent ILinkableComponent. The output is a combination of an "IValueDefinition" , an "IElementSet" , and an "ITimeSet" . This combination specifies which type of data can be provided where and when by the ILinkableComponent.
If an output does not provide the data in the way a consumer would like to have it the output can be adapted by an "ITimeSpaceAdaptedOutput" , which can transform the data according to the consumer's wishes. E.g. by performing interpolation in time, spatial aggregation, etc.).

If a output item does provide the data in the way a consumer would lik to have it the output item can be decorated by an "ITimeSpaceAdaptedOutput" , which can transform the data according to the consumer's wishes (e.g. by performing interpolation in time, spatial interpolation etc.).

Connections

Connector	Source	Target	Notes
Realisation Source -> Destination	Public ITimeSpaceAdaptedOutput	Public ITimeSpaceOutput	
Generalization Source -> Destination	Public ITimeSpaceOutput	Public ITimeSpaceExchangeItem	
Realisation Source -> Destination	Public ITimeSpaceOutput	Public IBaseOutput	

Operations

Method	Notes	Parameters
GetValues() ITimeSpaceValueSet Public	Overridden version of the "IBaseOutput.GetValues"method. "GetValues"now returns an "ITimeSpaceValueSet" , instead of an "IBaseValueSet" .	**IBaseExchangeItem** [in] querySpecifier

Method	Notes	Parameters
Values() ITimeSpaceValueSet Public	The exchange item's values, as a specialized "ITimeSpaceValueSet"	

1.32 ITimeSpaceValueSet

Type: **Interface IBaseValueSet**

The "ITimeSpaceValueSet"represents an ordered two-dimensional list of values. The first dimension stands for the times for which values are available, whereas in the second dimension each value belongs to precisely one element in the corresponding "IElementSet"(that was specified when asking for the values). In other words, the i-th value in that dimension of the value set corresponds to the i-th element in the IElementSet.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public ITimeSpaceValueSet	Public IBaseValueSet	

Operations

Method	Notes	Parameters
GetElementValuesForTime() IList Public	Get values from "Values2D" , for all elements, for the specified "timeIndex". If the data is time independent, timeIndex must be specified as 0.	**int** [in] timeIndex
GetTimeSeriesValuesForElement() IList Public	Get values from "Values2D" , for all times, for the specified "elementIndex". If the data is not related to a location, elementIndex must be specified as 0.	**int** [in] elementIndex
GetValue() object Public	Get the value for the specified "timeIndex"and "elementIndex"from "Values2D" . If the data is time independent, timeIndex must be specified as 0. If the data is not related to a location, elementIndex must be specified as 0.	**int** [in] timeIndex **int** [in] elementIndex
SetElementValuesForTime() void Public	Set values in "Values2D" , for all elements, for the specified "timeIndex" . If the data is time independent, timeIndex must be specified as 0.	**int** [in] timeIndex **IList** [in] values
SetTimeSeriesValuesForElement() void Public	Set values in "Values2D" , for all times, for the specified "elementIndex" . If the data is not related to a location, elementIndex must be specified as 0.	**int** [in] elementIndex **IList** [in] values
SetValue() void	Set the value in "Values2D" , for the specified	**int** [in] timeIndex

Method	Notes	Parameters
Public	"timeIndex"and "elementIndex" . If the data is time independent, timeIndex must be specified as 0. If the data is not related to a location, elementIndex must be specified as 0.	**int** [in] elementIndex **object** [in] value
Values2D() IList<IList> Public	Two-dimensional list of values. The first IList represents time, and the contained IList the element in the IElementSet.	

1.33 IUnit

Type: **Interface** **IDescribable**

Unit interface, describing the physical unit of a "IQuantity" .

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IUnit	Public IDescribable	

Operations

Method	Notes	Parameters
ConversionFactorToSI() double Public	Conversion factor to SI ('A' in: SI-value = A * quant-value + B)	
Dimension() IDimension Public	The unit's dimension	
OffSetToSI() double Public	OffSet to SI ('B' in: SI-value = A * quant-value + B)	

1.34 IValueDefinition

Type: **Interface** **IDescribable**

A ValueDefinition describes a value returned by the Values property and the GetValues function of the "IBaseExchangeItem" .

@remark This interface is not meant to be implemented directly. Instead, implement either "IQuality"or "IQuantity" , or a custom derived vale definition interface.

Connections

Connector	Source	Target	Notes
Generalization Source -> Destination	Public IQuality	Public IValueDefinition	
Generalization Source -> Destination	Public IValueDefinition	Public IDescribable	
Generalization Source -> Destination	Public IQuantity	Public IValueDefinition	

Operations

Method	Notes	Parameters
MissingDataValue() Object Public	The value representing that data is missing	
ValueType() Type Public	The object types of value that will be available in the "IBaseValueSet"that is returned by the Values property and the GetValues function of the "IBaseExchangeItem" .	