ECSS-E-TM-40-07 Volume 3A
25 January 2011

Space engineering

Simulation modelling platform - Volume 3: Component model

ECSS Secretariat
ESA-ESTEC
Requirements & Standards Division
Noordwijk, The Netherlands

Foreword

This document is one of the series of ECSS Technical Memoranda. Its Technical Memorandum status indicates that it is a non-normative document providing useful information to the space systems developers' community on a specific subject. It is made available to record and present non-normative data, which are not relevant for a Standard or a Handbook**.** Note that these data are non-normative even if expressed in the language normally used for requirements.

Therefore, a Technical Memorandum is not considered by ECSS as suitable for direct use in Invitation To Tender (ITT) or business agreements for space systems development.

Disclaimer

Published by:	ESA Requirements and Standards Division ESTEC, P.O. Box 299, 2200 AG Noordwijk The Netherlands
Copyright:	2011© by the European Space Agency for the members of ECSS

Change log

ECSS-E-TM-40-07 Volume 3A 25 January 2011	First issue

Table of contents

Figures

Tables

Introduction

Space programmes have developed simulation software for a number of years, and which are used for a variety of applications including analysis, engineering operations preparation and training. Typically different departments perform developments of these simulators, running on several different platforms and using different computer languages. A variety of subcontractors are involved in these projects and as a result a wide range of simulation models are often developed. This Technical Memorandum addresses the issues related to portability and reuse of simulation models. It builds on the work performed by ESA in the development of the Simulator Portability Standards SMP1 and SMP2.

This Technical Memorandum is complementary to ECSS-E-ST-40 because it provides the additional requirements which are specific to the development of simulation software. The formulation of this Technical Memorandum takes into account the Simulation Model Portability specification version 1.2. This Technical Memorandum has been prepared by the ECSS-E-40-07 Working Group.

This Technical Memorandum comprises of a number of volumes.

The intended readership of Volume 1 of this Technical Memorandum are the simulator software customer and all suppliers.

The intended readership of Volume 2, 3 and 4 of this Technical Memorandum is the Infrastructure Supplier.

The intended readership of Volume 5 of this Technical Memorandum is the simulator developer.

Note: Volume 1 contains the list of terms and abbreviations used in this document

- **Volume 1 – Principles and requirements**

  This document describes the Simulation Modelling Platform (**SMP**) and the special principles applicable to simulation software. It provides an interpretation of the ECSS-E-ST-40 requirements for simulation software, with additional specific provisions.

- **Volume 2 - Metamodel**

  This document describes the Simulation Model Definition Language (**SMDL**), which provides platform independent mechanisms to design models (Catalogue), integrate model instances (Assembly), and schedule them (Schedule). SMDL supports design and integration techniques for class-based, interface-based, component-based, event-based modelling and dataflow-based modelling.

- **Volume 3 - Component Model**

  This document provides a platform independent definition of the components used within an SMP simulation, where components include models and services, but also the simulator itself. A set of mandatory interfaces that every model has to implement is defined by the document, and a number of optional interfaces for advanced component mechanisms are specified.

  Additionally, this document includes a chapter on Simulation Services. Services are components that the models can use to interact with a Simulation Environment. SMP defines interfaces for mandatory services that every SMP compliant simulation environment must provide.

- **Volume 4 - C++ Mapping**

  This document provides a mapping of the platform independent models (Metamodel, Component Model and Simulation Services) to the ANSI/ISO C++ target platform. Further platform mappings are foreseen for the future.

  The intended readership of this document is the simulator software customer and supplier. The software simulator customer is in charge of producing the project Invitation to Tender (ITT) with the Statement of Work (SOW) of the simulator software. The customer identifies the simulation needs, in terms of policy, lifecycle and programmatic and technical requirements. It may also provide initial models as inputs for the modelling activities. The supplier can take one or more of the following roles:

  — Infrastructure Supplier - is responsible for the development of generic infrastructure or for the adaptation of an infrastructure to the specific needs of a project. In the context of a space programme, the involvement of Infrastructure Supplier team(s) may not be required if all required simulators are based on full re-use of exiting infrastructure(s), or where the infrastructure has open interfaces allowing adaptations to be made by the Simulator Integrator.

  — Model Supplier - is responsible for the development of project specific models or for the adaptation of generic models to the specific needs of a project or project phase.

  — Simulator Integrator – has the function of integrating the models into a simulation infrastructure in order to provide a full system simulation with the appropriate services for the user (e.g. system engineer) and interfaces to other systems.

- **Volume 5 – SMP usage**

  This document provides a user-oriented description of the general concepts behind the SMP documents Volume 1 to 4, and provides instructions for the accomplishment of the main tasks involved in model and simulator development using SMP.

1 Scope

The Platform Independent Model (**PIM**) of the Simulation Modelling Platform (**SMP**) consists of two parts:

1. The SMP Metamodel, also called Simulation Model Definition Language (**SMDL**).
2. The SMP Component Model, which includes SMP Simulation Services.

This document introduces the platform independent component model used by SMP components. It introduces interfaces for all components, models, services and simulators. As this document does not define classes (i.e. an implementation), but only interfaces, most names in this document start with an upper-case "I", which stands for "Interface".

1.1 Objects

In order to allow harmonisation of components with other SMP elements, this document first defines what an object is (IObject). Although most elements of the SMP component model are components, these components expose some objects, which are derived directly from the IObject interface. These objects include fields, entry points, event sources and event sinks, containers and references, and failures. See section 3.2.1 for details on objects.

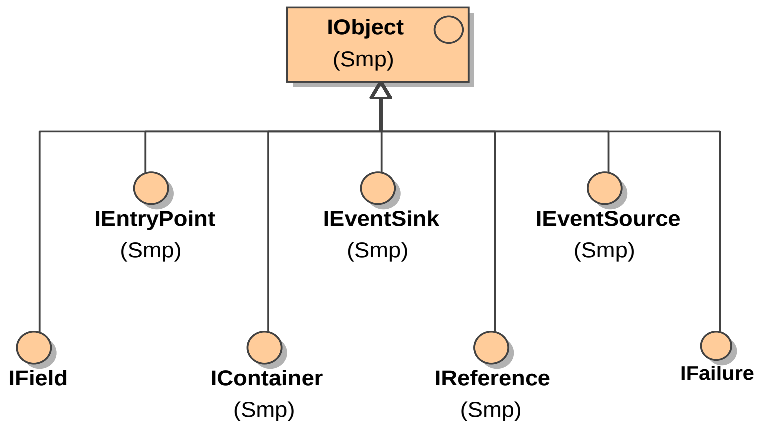


Figure 1-1: Objects

1.2 Components

To allow treating all components in a similar way, it is then defined what a component is (i.e. the IComponent interface). The three most important interfaces derived from this base interface are IModel for models, IService for services and ISimulator for the simulation environment (where ISimulator is not derived immediately from IComponent, but from IComposite). See section 3.2.2 for details on Components, and section 3.6 for details on Simulators.

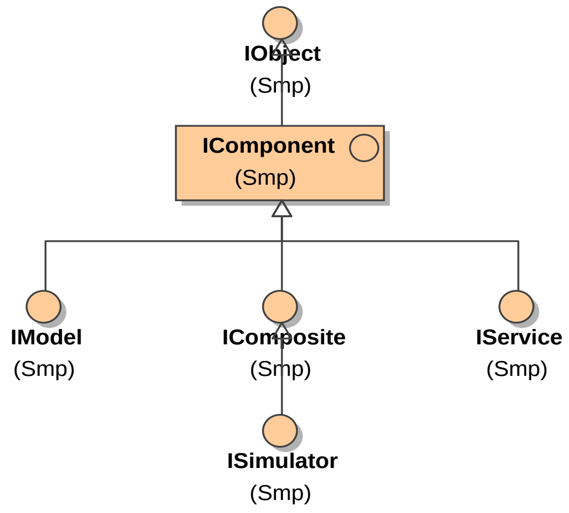


Figure 1-2: Components

1.3 Component Mechanisms

Further, SMP defines standard mechanisms how to enhance components, for example for component aggregation (IAggregate), component composition (IComposite), inter-component events (IEventProvider and IEventConsumer), entry point publication (IEntryPointPublisher), managed components (IManagedComponent), dynamic invocation (IDynamicInvocation) and self-persistence (IPersist). See section 3.3 for details on Component Mechanisms, and section 3.5.2 for Managed Component Mechanisms.

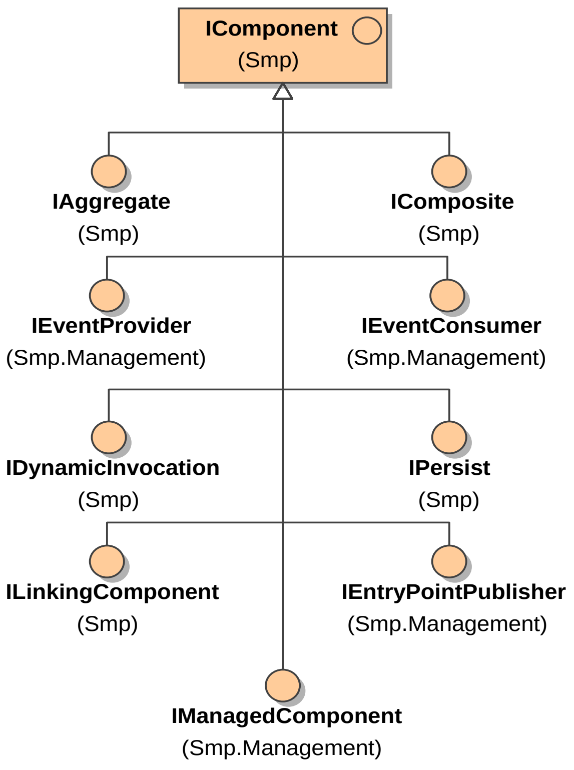


Figure 1-3: Component Mechanisms

1.4 Model Mechanisms

The most important component type of SMP is the Model. While implementing the mandatory IModel interface is sufficient to make a model SMP compliant, this does not allow using most of the SMP mechanisms. Therefore, optional extensions of the IModel interface do exist providing optional features for models that go beyond those for components. An example is the fallible model (IFallibleModel). See section 3.4 for details on Model Mechanisms, and section 3.5.3 for Managed Model Mechanisms.

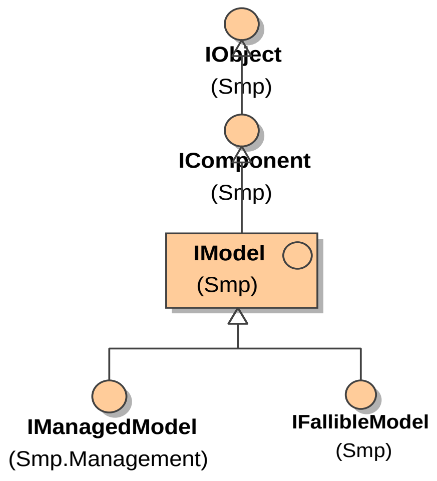


Figure 1-4: Model Mechanisms

1.5 Management Interfaces

To support managed simulations, where all information about the models, their links and initial values is read from an external source (typically XML files), SMP defines optional mechanisms for managed components. These allow setting their properties (name, description, and parent) from external applications, and provide mechanisms to query their elements (fields, entry points, event sources and sinks) by name. See section 3.5 for details on Management Interfaces.

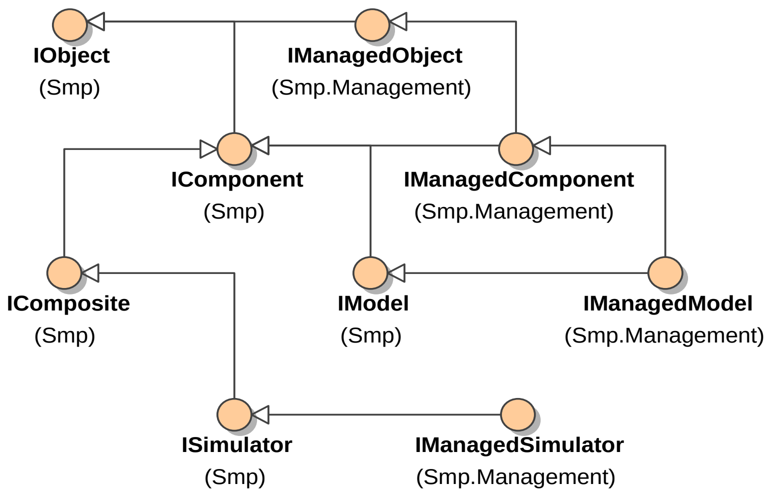


Figure 1-5: Management Interfaces

1.6 Simulation Environments

Finally, the Simulator or Simulation Environment is specified in this document. Its base inter-face ISimulator allows accessing both the simulation services and the models, but the simu-lation environment also has to provide a publication mechanism to the models. See section 3.6 for details on Simulation Environments.

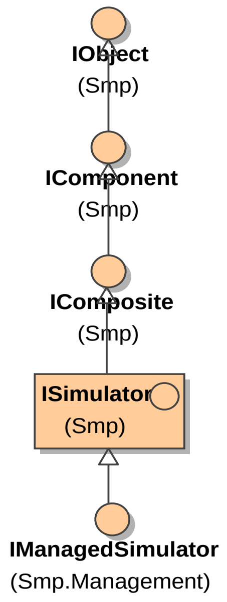


Figure 1-6: Simulation Environments

1.7 Simulation Services

In order to facilitate the inter-operability between SMP compliant simulation environments (i.e. run-time simulation kernels), several Simulation Services are defined in the SMP specification. A mechanism of how models can acquire services is part of the document. See section 4 for de-tails on Simulation Services.

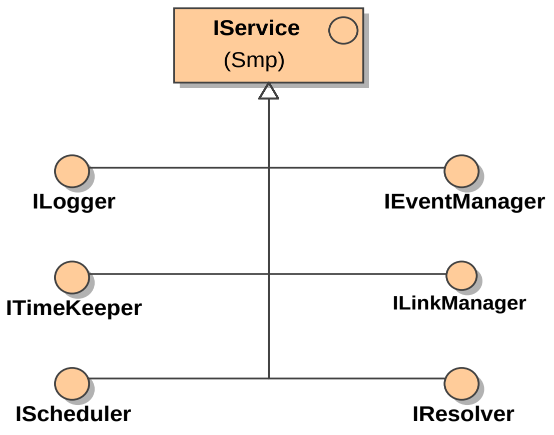


Figure 1-7: Simulation Services

1.8 Exceptions

SMP defines specific exceptions, and links them to the operations of the interfaces. All exceptions are expressed in UML, and derive from a common Exception base class.

The rationale of the SMP Component Model is to use exceptions carefully, following these rules:

- Exceptions are only used for unexpected behaviour. For expected behaviour (for example no service found with given name), the return value is used (typically a return value of null).
- Exceptions are not only used to indicate *that* an operation has failed, but also to indicate *why* it failed. Most exceptions carry useful, precise, complete information. In most cases, the name of the exception clearly tells why an operation has failed.
- Exceptions are typically used when different error sources exist, or when additional in-formation about the error source (for example the expected number of parameters) is passed back to the caller.

Further exceptions may be added to the document in later revisions based on a first reference implementation of SMP.

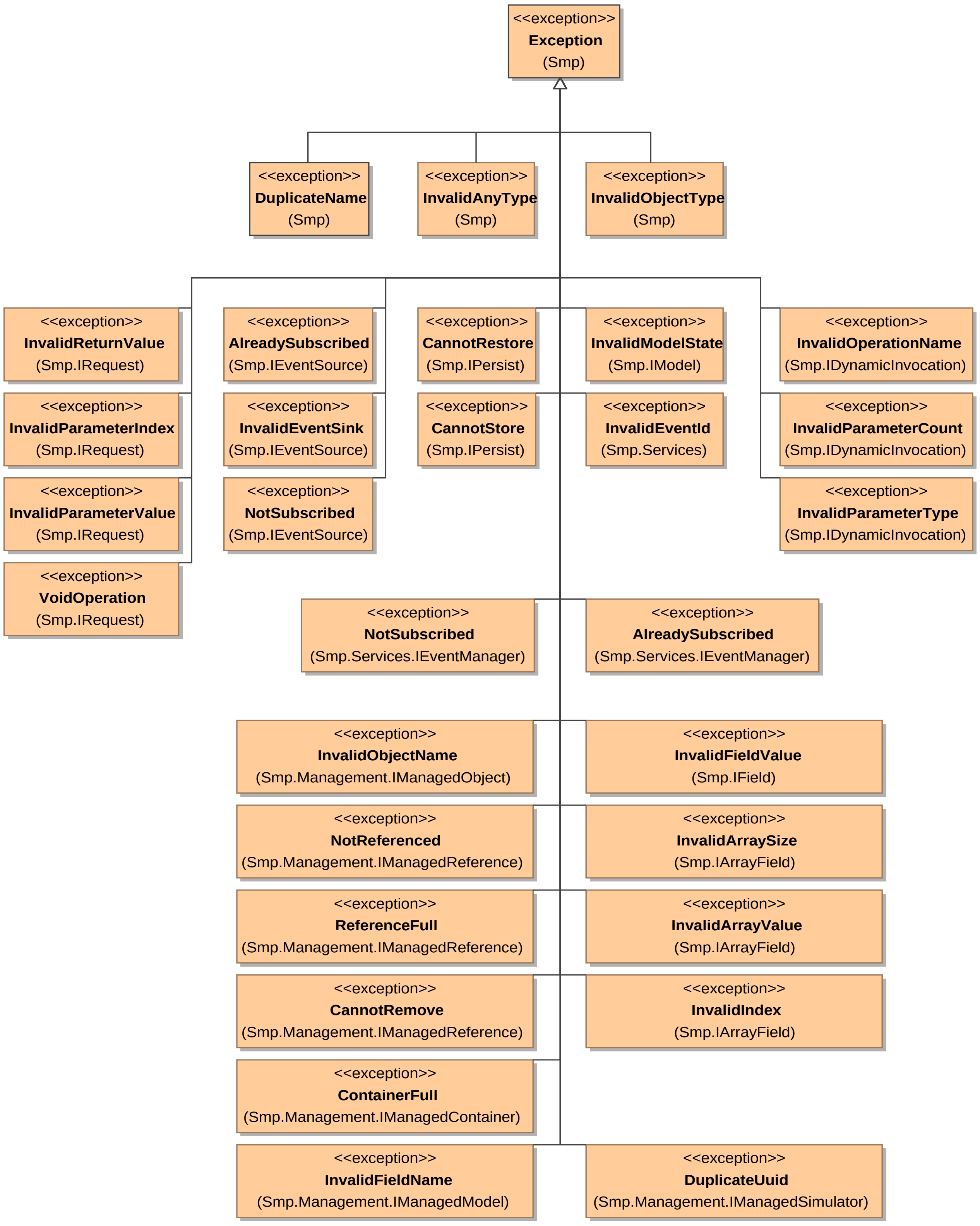


Figure 1-8: Exceptions

2 Base Types

2.1 Primitive Types

The type system is based on a number of pre-defined primitive types. For each platform, a mapping of these types to native platform types has to be provided. The semantics of these types is defined in this Component Model, so that they have a consisting semantics on all platforms.

2.1.1 Char8

8 bit character type.

2.1.2 Bool

Boolean with true and false.

2.1.3 Int8

8 bit signed integer type.

2.1.4 UInt8

8 bit unsigned integer type.

2.1.5 Int16

16 bit signed integer type.

2.1.6 UInt16

16 bit unsigned integer type.

2.1.7 Int32

32 bit signed integer type.

2.1.8 UInt32

32 bit unsigned integer type.

2.1.9 Int64

64 bit signed integer type.

2.1.10 UInt64

64 bit unsigned integer type.

2.1.11 Float32

32-bit single-precision float type.

2.1.12 Float64

64-bit double-precision float type.

2.1.13 Duration

Duration in nanoseconds.

This type is used for relative time values. It specifies a duration in nanoseconds. The following holds for Duration:

- Duration is a value measured in nanoseconds, which is the lowest level of granularity supported for time in SMP.
- Duration is stored as a signed 64-bit integer value.
- Positive values correspond to positive durations, and negative values correspond to negative durations.

This allows specifying duration values roughly between -290 years and 290 years. With this definition, the Duration type is compatible with the DateTime type.

2.1.14 Date Time

Absolute time in nanoseconds.

This type is used for absolute time values. It specifies a time in nanoseconds, relative to a reference time.

The following holds for DateTime:

- Time is a value measured in nanoseconds, which is the lowest level of granularity supported for time in SMP.
- Time is stored as a signed 64-bit integer value, relative to the reference time (01.01.2000, 12:00, Modified Julian Date (MJD) 2000+0.5).

- Positive values correspond to times after the reference time, and negative values correspond to time values before the reference time.

This allows specifying time values roughly between 1710 and 2290. With this definition, the DateTime type is compatible with the Duration type.

2.1.15 String8

8 bit character string.

2.2 Simple Type Union

Some of the interfaces defined in the Component Model make use of the AnySimple type. This data type represents a type that can hold any of the simple types defined in SMP, which includes all primitive types defined above. As for the primitive types, a native platform mapping has to be provided for each platform.

2.2.1 Primitive Type Kind

This is an enumeration of the available primitive types.

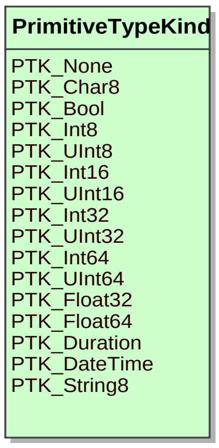


Figure 2-1: Primitive Type Kind

Table 2-1: Enumeration Literals of PrimitiveTypeKind

Name	Description
PTK_None	No type, e.g. for void.
PTK_Char8	8 bit character type.
PTK_Bool	Boolean with true and false.
PTK_Int8	8 bit signed integer type.
PTK_UInt8	8 bit unsigned integer type.
PTK_Int16	16 bit signed integer type.
PTK_UInt16	16 bit unsigned integer type.
PTK_Int32	32 bit signed integer type.
PTK_UInt32	32 bit unsigned integer type.
PTK_Int64	64 bit signed integer type.
PTK_UInt64	64 bit unsigned integer type.
PTK_Float32	32 bit single-precision floating-point type.
PTK_Float64	64 bit double-precision floating-point type.
PTK_Duration	Duration in nanoseconds.
PTK_DateTime	Absolute time in nanoseconds.
PTK_String8	8 bit character string.

2.2.2 Any Simple Array

Array of AnySimple values.

2.2.3 Any Simple

Variant that can store a value of any of the simple types. The type attribute defines the type used to represent the value, while the value attribute contains the actual value.

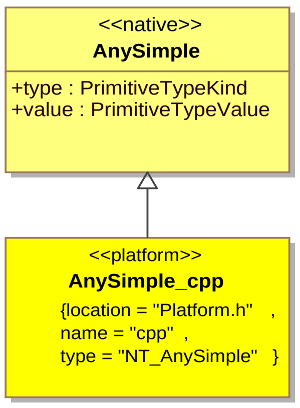


Figure 2-2: Any Simple

2.2.4 Primitive Type Value

Union of primitive type values, which is used for the value field of AnySimple.

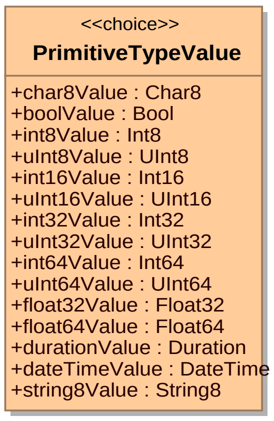


Figure 2-3: Primitive Type Value

Table 2-2: Alternatives

Name	Type	Description
char8Value	Char8	8 bit character value.
boolValue	Bool	Boolean with true and false.
int8Value	Int8	8 bit signed integer value.
uInt8Value	UInt8	8 bit unsigned integer value.
int16Value	Int16	16 bit signed integer value.
uInt16Value	UInt16	16 bit unsigned integer value.
int32Value	Int32	32 bit signed integer value.
uInt32Value	UInt32	32 bit unsigned integer value.
int64Value	Int64	64 bit signed integer value.
uInt64Value	UInt64	64 bit unsigned integer value.
float32Value	Float32	32 bit single-precision floating-point value.
float64Value	Float64	64 bit double-precision floating-point value.
durationValue	Duration	Duration in nanoseconds.
dateTimeValue	DateTime	Absolute time in nanoseconds.
string8Value	String8	8 bit character string value.

2.3 Universally Unique Identifiers

For a unique identification of types (and hence models), SMP uses Universally Unique Identifiers with the format specified by the Open Group (http://www.opengroup.org).

2.3.1 Uuid

Universally Unique Identifier.

A Universally Unique Identifier is 128 bit long, separated into 32 hex nibbles (where each hex nibble stands for 4 bits).

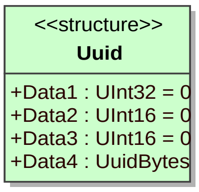


Figure 2-4: Uuid

Table 2-3: Fields of Uuid

Name	Type	Description
Data1	UInt32	8 hex nibbles.
Data2	UInt16	4 hex nibbles.
Data3	UInt16	4 hex nibbles.
Data4	UuidBytes	4+12 hex nibbles.

2.3.2 Uuid Bytes

Final 8 bytes of Uuid.

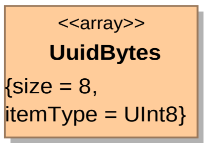


Figure 2-5: Uuid Bytes

Table 2-4: Specification of UuidBytes

Item type	Size
UInt8	8

3 Component Model

3.1 Exceptions

SMP defines some basic exceptions which are used in several interfaces, and which are therefore defined outside of an individual interface. For each exception, see the detailed specification of the interfaces to find out which methods actually may raise this exception.

3.1.1 Exception

This is the base class for all SMP exceptions.

This exception is the base class for all other SMP exceptions. It provides Name, Description and Message.

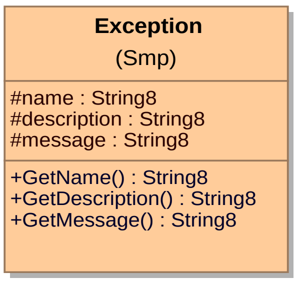


Figure 3-1: Exception

Base Exceptions

None

Fields

Table 3-1: Fields of Exception

Name	Type	Description
description	String8	Description of the exception that is returned by GetDescription.
message	String8	Description of the problem that is returned by GetMessage.
name	String8	Name of the exception that is returned by GetName.

3.1.2 Duplicate Name

This exception is raised when trying to add an object to a collection of objects, which have to have unique names, but another object with the same name does exist already in this collection. This would lead to duplicate names.

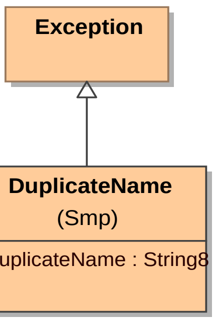


Figure 3-2: Duplicate Name

Base Exceptions

Smp::Exception

Fields

Table 3-2: Fields of DuplicateName

Name	Type	Description
duplicateName	String8	Name that already exists in the collection.

3.1.3 Invalid Any Type

This exception is raised when trying to use an AnySimple argument of wrong type.

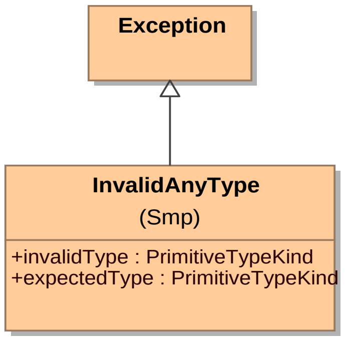

Exception
InvalidAnyType
(Smp)
+invalidType : PrimitiveTypeKind
+expectedType : PrimitiveTypeKind

Figure 3-3: Invalid Any Type

Remark: This can happen when assigning a value to an AnySimple instance, but as well when e.g. registering an event sink with an event source of another event argument type.

Base Exceptions

Smp::Exception

Fields

Table 3-3: Fields of InvalidAnyType

Name	Type	Description
expectedType	PrimitiveTypeKind	Type that was expected.
invalidType	PrimitiveTypeKind	Type that is not valid.

3.1.4 Invalid Event Id

This exception is raised when an invalid event id is provided, e.g. when calling Subscribe(), Unsubscribe() or Emit() of the Event Manager (using an invalid global event id), or when calling SetEventSimulationTime(), SetEventMissionTime(), SetEventEpochTime(), SetEventZuluTime(), SetEventCycleTime(), SetEventCount() or RemoveEvent() of the Scheduler (using an invalid scheduler event id).

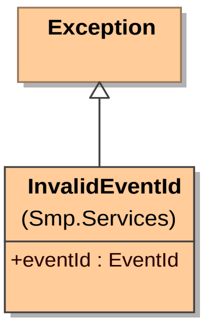


Figure 3-4: Invalid Event Id

Base Exceptions

Smp::Exception

Fields

Table 3-4: Fields of InvalidEventId

Name	Type	Description
eventId	EventId	Invalid event identifier.

3.1.5 Invalid Object Type

This exception is raised when trying to pass an object of wrong type.

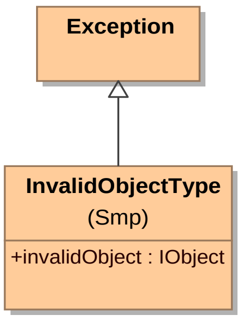


Figure 3-5: Invalid Object Type

Remark: This can happen when adding a component to a container or reference which is semantically typed by a specific type implementing IComponent.

Base Exceptions

Smp::Exception

Fields

Table 3-5: Fields of InvalidObjectType

Name	Type	Description
invalidObject	IObject	Object that is not of valid type.

3.2 Objects and Components

In SMP, a simulation is composed of components, where models, services, and the simulation environment all implement a common base interface. Other elements in SMP are not components, but only objects.

3.2.1 Objects

Objects are elements of a simulation which provide name and description.

3.2.1.1 IObject

This interface is the base interface for almost all other SMP interfaces. While most interfaces derive from IComponent, which itself is derived from IObject, some objects (including IField, IFailure, IEntryPoint, IEventSink, IEventSource, IContainer and IReference) are directly derived from IObject.

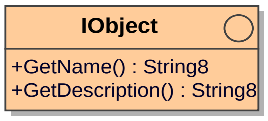


Figure 3-6: IObject

Remark: The two methods of this interface ensure that all SMP objects can be shown with a name, and with an optional description.

Base Interfaces

None

Operations

Table 3-6: Operations of IObject

Name	Description
GetDescription	Return the description of the object ("property getter").
GetName	Return the name of the object ("property getter").

3.2.1.1.1 Get Description

Return the description of the object ("property getter").

Descriptions are optional and may be empty.

Remark: Applications may display the description as additional information about the object.

Prototype

public String8 GetDescription();

Parameters

Table 3-7: Parameters of GetDescription

Name	Dir.	Type	Description
	return	String8	Description of object.

Exceptions

None

3.2.1.1.2 Get Name

Return the name of the object ("property getter").

Names must

- be unique within their context,
- not be empty,
- start with a letter, and

- only contain letters, digits, the underscore ("_") and brackets ("[" and "]").

Remark: Applications may display the name as user readable object identification.

Remark: It is recommended that names do not exceed 32 characters in size.

Prototype

public String8 GetName();

Parameters

Table 3-8: Parameters of GetName

Name	Dir.	Type	Description
	return	String8	Name of object.

Exceptions

None

3.2.2 Components

Most elements in SMP are components, which implement the IComponent interface.

The three most important component types are models, services, and the simulator. The first two of these interfaces are introduced in this section, while the ISimulator interface is explained in section 3.6.1.2.

3.2.2.1 IComponent

This is the base interface for all SMP components.

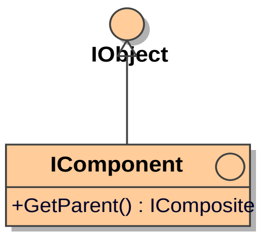


Figure 3-7: IComponent

Remark: SMP components are typically models and services.

Base Interfaces

Smp::IObject

Operations

Table 3-9: Operations of IComponent

Name	Description
GetParent	Return the parent component of the component ("property getter").

3.2.2.1.1 Get Parent

Return the parent component of the component ("property getter").

Components link to their parent to allow traversing the tree of components upwards to the root component of a composition.

Remark: Typically, only the simulator itself is a root component, so all other components should have a parent component.

Prototype

public IComposite GetParent();

Parameters

Table 3-10: Parameters of GetParent

Name	Dir.	Type	Description
	return	IComposite	Parent component of component or null if component has no parent.

Exceptions

None

3.2.2.2 Component Collection

A component collection is an ordered collection of components, which allows iterating all members.

This type is platform specific. For details see the SMP Platform Mappings.

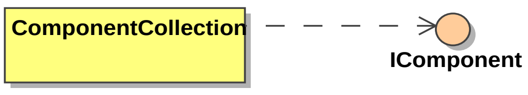


Figure 3-8: Component Collection

3.2.2.3 Model State Kind

This is an enumeration of the available states of a model. Each model is always in one of these four model states.

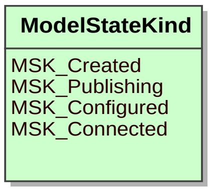


Figure 3-9: Model State Kind

Table 3-11: Enumeration Literals of ModelStateKind

Name	Description
MSK_Created	The Created state is the initial state of a model. Model creation is done by an external mechanism, e.g. by factories. This state is entered automatically after the model has been created. This state is left via the Publish() state transition.
MSK_Publishing	In Publishing state, the model is allowed to publish features. This includes publication of fields, operations and properties. In addition, the model is allowed to create other models. This state is entered via the Publish() state transition. This state is left via the Configure() state transition.
MSK_Configured	In Configured state, the model has been fully configured. This configuration may be done by external components, or internally by the model itself, e.g. by reading data from an external source. This state is entered via the Configure() state transition. This state is left via the Connect() state transition.
MSK_Connected	In Connected state, the model is connected to the simulator. In this state, neither publication nor creation of other models is allowed anymore. This state is entered via the Connect() state transition. This is the final state of a model, and only left on termination.

3.2.2.4 IModel

Interface for a model.

All SMP models implement this interface. As models interface to the simulation environment, they have a dependency to it via the two interfaces IPublication and ISimulator.

This is the only mandatory interface models have to implement. All other interfaces (component and model mechanisms and managed interfaces) are optional.

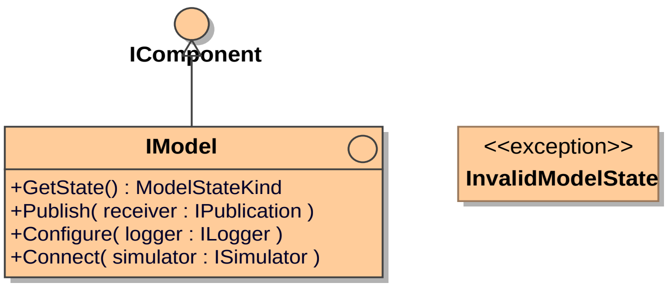

IComponent
IModel
+GetState() : ModelStateKind
+Publish(receiver : IPublication)
+Configure(logger : ILogger)
+Connect(simulator : ISimulator)
<<exception>>
InvalidModelState

Figure 3-10: IModel

Base Interfaces

Smp::IComponent

Operations

Table 3-12: Operations of IModel

Name	Description
Configure	Request the model to perform any custom configuration. The model can create and configure other models using the field values of its published fields.
Connect	Allow the model to connect to the simulator.
GetState	Returns the state the model is currently in.
Publish	Request the model to publish its fields, properties and operations against the provided publication receiver.

3.2.2.4.1 Configure

Request the model to perform any custom configuration. The model can create and configure other models using the field values of its published fields.

This method can only be called once for each model, and only when the model is in Publishing state. The method raises an InvalidModelState exception if the model is not in Publishing state.

The model can still publish further features in this call, and can even create other models, but at the end of this call, it needs to enter the Configured state.

Remark: The simulation environment typically calls this method in the Building state.

Prototype

public void Configure(in ILogger logger) raises (InvalidModelState);

Parameters

Table 3-13: Parameters of Configure

Name	Dir.	Type	Description
logger	in	ILogger	Logger service for logging of error messages during configuration.

Exceptions

Smp::IModel::InvalidModelState

3.2.2.4.2 Connect

Allow the model to connect to the simulator.

This method can only be called once for each model, and only when the model is in the Configured state. The method raises an InvalidModelState exception if the model is not in Configured state.

When this operation is called, the model immediately enters the Connected state, before it uses any of the simulator methods and services.

In this method, the model may query for and use any of the available simulation services, as they are all guaranteed to be fully functional at that time. It may as well connect to other models' functionality (e.g. to event sources), as it is guaranteed that all models have been created and configured before the Connect() method of any model is called.

Remark: The simulation environment typically calls this method in the Connecting state.

Prototype

public void Connect(in ISimulator simulator) raises (InvalidModelState);

Parameters

Table 3-14: Parameters of Connect

Name	Dir.	Type	Description
simulator	in	ISimulator	Simulation Environment that hosts the model.

Exceptions

Smp::IModel::InvalidModelState

3.2.2.4.3 Get State

Returns the state the model is currently in.

The model state can be changed using the Publish(), Configure() and Connect() state transition methods.

Prototype

public ModelStateKind GetState();

Parameters

Table 3-15: Parameters of GetState

Name	Dir.	Type	Description
	return	ModelStateKind	Current model state.

Exceptions

None

3.2.2.4.4 Publish

Request the model to publish its fields, properties and operations against the provided publication receiver.

This method can only be called once for each model, and only when the model is in the Created state. The method raises an InvalidModelState exception if the model is not in Created state.

When this operation is called, the model immediately enters the Publishing state, before it publishes any of its features.

Remark: The simulation environment typically calls this method in the Building state.

Prototype

public void Publish(in IPublication receiver) raises (InvalidModelState);

Parameters

Table 3-16: Parameters of Publish

Name	Dir.	Type	Description
receiver	in	IPublication	Publication receiver.

Exceptions

Smp::IModel::InvalidModelState

3.2.2.4.5 Invalid Model State

This exception is raised by a model when one of the state transition commands is called in an invalid state.

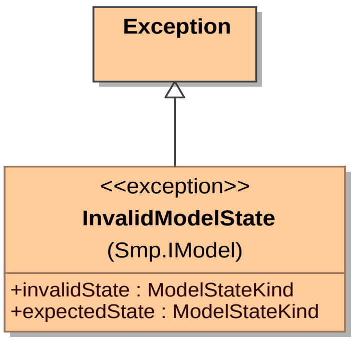

Exception
<<exception>>
InvalidModelState
(Smp.IModel)
+invalidState : ModelStateKind
+expectedState : ModelStateKind

Figure 3-11: Invalid Model State

Fields

Table 3-17: Fields of InvalidModelState

Name	Type	Description
expectedState	ModelStateKind	State that was expected.
invalidState	ModelStateKind	State that is not valid.

3.2.2.5 Model Collection

A model collection is an ordered collection of models, which allows iterating all members.

This type is platform specific. For details see the SMP Platform Mappings.

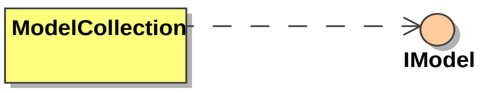


Figure 3-12: Model Collection

3.2.2.6 IService

Base interface for all SMP services.

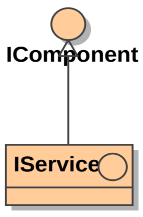


Figure 3-13: IService

Remark: Currently, this interface does not add any functionality.

Base Interfaces

Smp::IComponent

Operations

None

3.2.2.7 Service Collection

A service collection is an ordered collection of services, which allows iterating all members.

This type is platform specific. For details see the SMP Platform Mappings.

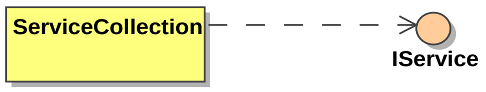


Figure 3-14: Service Collection

3.3 Component Mechanisms

While the IComponent base interface provides mechanisms to get name, description, and parent, it does not allow specifying further relations between components. The mechanisms supported by SMP are aggregation, composition, entry points, inter-component events via event sources and event sinks, dynamic invocation and persistence.

3.3.1 Aggregation

Via aggregation, a component can ***reference*** other components in the component hierarchy to use their methods. As opposed to composition, an aggregated component is not owned, but only referenced.

3.3.1.1 IAggregate

Interface for an aggregate component.

A component with references to other components implements this interface. Referenced components are held in named references.

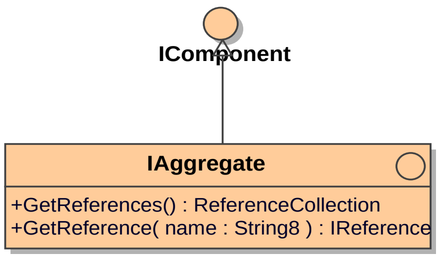

IComponent
IAggregate
+GetReferences() : ReferenceCollection
+GetReference(name : String8) : IReference

Figure 3-15: IAggregate

Remark: This interface represents the Aggregation mechanism in the SMP Metamodel (via References). In UML 2.0, this is represented by a required interface.

Base Interfaces

Smp::IComponent

Operations

Table 3-18: Operations of IAggregate

Name	Description
GetReference	Query for a reference of this aggregate component by its name.
GetReferences	Query for the collection of all references of the aggregate component.

3.3.1.1.1 Get Reference

Query for a reference of this aggregate component by its name.

The returned reference may be null if no reference with the given name could be found. If more than one reference with this name exists, it is not defined which one is returned.

Prototype

public IReference GetReference(in String8 name);

Parameters

Table 3-19: Parameters of GetReference

Name	Dir.	Type	Description
	return	IReference	Reference queried for by name, or null if no reference with this name exists.
name	in	String8	Reference name.

Exceptions

None

3.3.1.1.2 Get References

Query for the collection of all references of the aggregate component.

The returned collection may be empty if no references exist for the aggregate.

Prototype

public ReferenceCollection GetReferences();

Parameters

Table 3-20: Parameters of GetReferences

Name	Dir.	Type	Description
	return	ReferenceCollection	Collection of references.

Exceptions

None

3.3.1.2 IReference

Interface for a reference.

A reference allows querying for the referenced components.

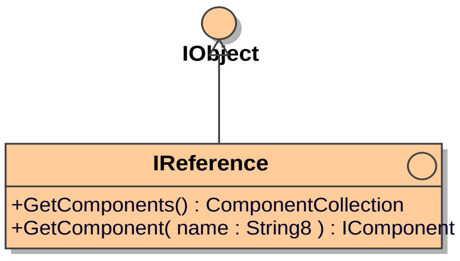

IObject
IReference
+GetComponents() : ComponentCollection
+GetComponent(name : String8) : IComponent

Figure 3-16: IReference

Remark: References are used together with the IAggregate interface for aggregation.

Base Interfaces

Smp::IObject

Operations

Table 3-21: Operations of IReference

Name	Description
GetComponent	Query for a referenced component by its name.
GetComponents	Query for the collection of all referenced components.

3.3.1.2.1 Get Component

Query for a referenced component by its name.

The returned component may be null if no component with the given name could be found.

Prototype

public IComponent GetComponent(in String8 name);

Parameters

Table 3-22: Parameters of GetComponent

Name	Dir.	Type	Description
	return	IComponent	Referenced component with the given name, or null if no referenced component with the given name could be found.
name	in	String8	Component name.

Exceptions

None

3.3.1.2.2 Get Components

Query for the collection of all referenced components.

The returned collection may be empty if no components are referenced.

Prototype

public ComponentCollection GetComponents();

Parameters

Table 3-23: Parameters of GetComponents

Name	Dir.	Type	Description
	return	ComponentCollection	Collection of referenced components.

Exceptions

None

3.3.1.3 Reference Collection

A reference collection is an ordered collection of references, which allows iterating all members.

This type is platform specific. For details see the SMP Platform Mappings.

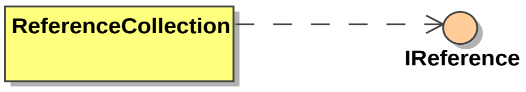


Figure 3-17: Reference Collection

3.3.2 Composition

Via composition, a component can ***contain*** other components in the component hierarchy. As opposed to aggregation, a component is owned, and its life-time coincides with its parent component. Composition is the counter-part to the GetParent() method of the IComponent interface and allows traversing the tree of components in downward direction.

3.3.2.1 IComposite

Interface for a composite component.

A component with children implements this interface. Child components are held in named containers.

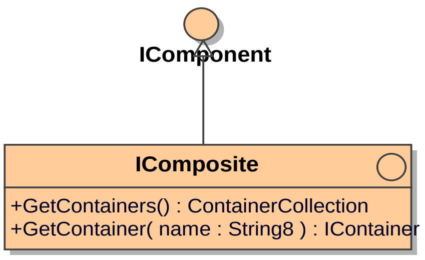

IComponent
IComposite
+GetContainers() : ContainerCollection
+GetContainer(name : String8) : IContainer

Figure 3-18: IComposite

Remark: This interface represents the Composition mechanism in the SMP Metamodel (via Containers). In UML 2.0, this is represented by composite aggregation.

Base Interfaces

Smp::IComponent

Operations

Table 3-24: Operations of IComposite

Name	Description
GetContainer	Query for a container of this composite component by its name.
GetContainers	Query for the collection of all containers of the composite component.

3.3.2.1.1 Get Container

Query for a container of this composite component by its name.

The returned container may be null if no container with the given name could be found.

Prototype

public IContainer GetContainer(in String8 name);

Parameters

Table 3-25: Parameters of GetContainer

Name	Dir.	Type	Description
	return	IContainer	Container queried for by name, or null if no container with this name exists.
name	in	String8	Container name.

Exceptions

None

3.3.2.1.2 Get Containers

Query for the collection of all containers of the composite component.

The returned collection may be empty if no containers exist for the composite.

Prototype

public ContainerCollection GetContainers();

Parameters

Table 3-26: Parameters of GetContainers

Name	Dir.	Type	Description
	return	ContainerCollection	Collection of containers.

Exceptions

None

3.3.2.2 IContainer

Interface for a container.

A container allows querying for its children.

Containers are used together with the IComposite interface for composition.

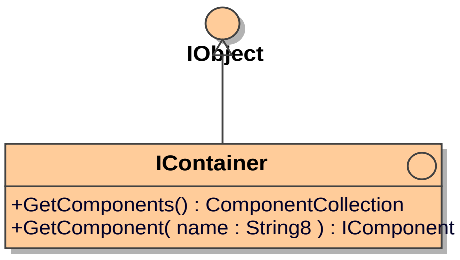

IObject
IContainer
+GetComponents() : ComponentCollection
+GetComponent(name : String8) : IComponent

Figure 3-19: IContainer

Base Interfaces

Smp::IObject

Operations

Table 3-27: Operations of IContainer

Name	Description
GetComponent	Query for a component contained in the container by name.
GetComponents	Query for the collection of all components in the container.

3.3.2.2.1 Get Component

Query for a component contained in the container by name.

The returned component may be null if no child with the given name could be found.

Prototype

public IComponent GetComponent(in String8 name);

Parameters

Table 3-28: Parameters of GetComponent

Name	Dir.	Type	Description
	return	IComponent	Child component, or null if no child component with the given name exists.
name	in	String8	Child name.

Exceptions

None

3.3.2.2.2 Get Components

Query for the collection of all components in the container.

The returned collection may be empty if no components exist for the container.

Prototype

public ComponentCollection GetComponents();

Parameters

Table 3-29: Parameters of GetComponents

Name	Dir.	Type	Description
	return	ComponentCollection	Collection of contained components.

Exceptions

None

3.3.2.3 Container Collection

A container collection is an ordered collection of containers, which allows iterating all members.

This type is platform specific. For details see the SMP Platform Mappings.

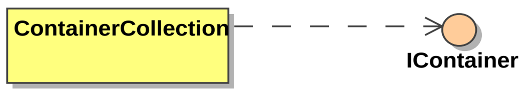


Figure 3-20: Container Collection

3.3.3 Events

Events are used in event-based programming. Event-based programming works via event sources and event sinks that can be registered to and unregistered from event sources. When an event source emits an event, it notifies all subscribed event sinks.

3.3.3.1 IEvent Sink

Interface of an event sink that can be subscribed to an event source (IEventSource).

This interface provides a notification method (event handler) that can be called by event sources when an event is emitted.

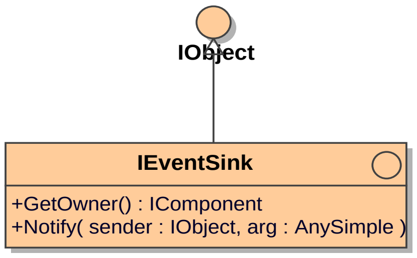


Figure 3-21: IEvent Sink

Base Interfaces

Smp::IObject

Operations

Table 3-30: Operations of IEventSink

Name	Description
GetOwner	This method returns the Component that owns the event sink.
Notify	This event handler method is called when an event is emitted.

3.3.3.1.1 Get Owner

This method returns the Component that owns the event sink.

Remark: This is required to be able to store and restore event sinks.

Prototype

public IComponent GetOwner();

Parameters

Table 3-31: Parameters of GetOwner

Name	Dir.	Type	Description
	return	IComponent	Owner of event sink.

Exceptions

None

3.3.3.1.2 Notify

This event handler method is called when an event is emitted.

Components providing event sinks must ensure that these event sinks do not throw exceptions.

Prototype

public void Notify(in IObject sender, in AnySimple arg);

Parameters

Table 3-32: Parameters of Notify

Name	Dir.	Type	Description
sender	in	IObject	Object emitting the event.
arg	in	AnySimple	Event argument.

Exceptions

None

3.3.3.2 Event Sink Collection

An event sink collection is an ordered collection of event sinks, which allows iterating all members.

This type is platform specific. For details see the SMP Platform Mappings.

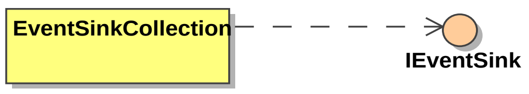


Figure 3-22: Event Sink Collection

3.3.3.3 IEvent Source

Interface of an event source that event sinks (IEventSink) can subscribe to.

This interface allows event consumers to subscribe to or unsubscribe from an event.

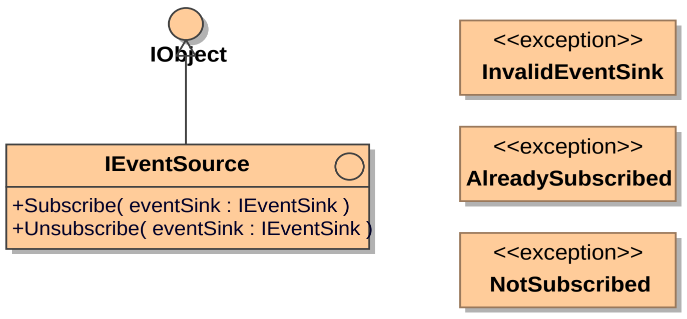


Figure 3-23: IEvent Source

Base Interfaces

Smp::IObject

Operations

Table 3-33: Operations of IEventSource

Name	Description
Subscribe	Subscribe to the event source, i.e. request notifications.
Unsubscribe	Unsubscribe from the event source, i.e. cancel notifications.

3.3.3.3.1 Subscribe

Subscribe to the event source, i.e. request notifications.

This method raises the AlreadySubscribed exception, if the given event sink is already subscribed to the event source. In addition, an exception of type InvalidEventSink may be raised when the event argument of the event sink is not of the type the event source expects. This exception depends on additional metadata that is not defined in this component model.

An event sink can only be subscribed once to each event source. Event sinks will be called in the order they have been subscribed to the event source.

Models providing event sinks must ensure that these event sinks do not throw exceptions.

Remark: Implementations may perform type checking on the optional event argument of the event source and event sink.

Prototype

public void Subscribe(in IEventSink eventSink) raises (AlreadySubscribed, InvalidEventSink);

Parameters

Table 3-34: Parameters of Subscribe

Name	Dir.	Type	Description
eventSink	in	IEventSink	Event sink to subscribe to event source.

Exceptions

Smp::IEventSource::AlreadySubscribed, Smp::IEventSource::InvalidEventSink

3.3.3.3.2 Unsubscribe

Unsubscribe from the event source, i.e. cancel notifications.

This method raises the NotSubscribed exception if the given event sink is not subscribed to the event source.

An event sink can only be unsubscribed if it has been subscribed before.

Prototype

public void Unsubscribe(in IEventSink eventSink) raises (NotSubscribed);

Parameters

Table 3-35: Parameters of Unsubscribe

Name	Dir.	Type	Description
eventSink	in	IEventSink	Event sink to unsubscribe from event source.

Exceptions

Smp::IEventSource::NotSubscribed

3.3.3.3.3 Already Subscribed

This exception is raised when trying to subscribe an event sink to an event source that is already subscribed.

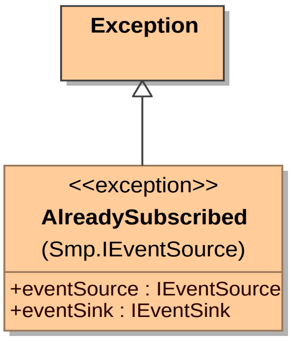


Figure 3-24: Already Subscribed

Fields

Table 3-36: Fields of AlreadySubscribed

Name	Type	Description
eventSink	IEventSink	Event sink that is already subscribed.
eventSource	IEventSource	Event source the event sink is subscribed to.

3.3.3.3.4 Invalid Event Sink

This exception is raised when trying to subscribe an event sink to an event source that has a different event type.

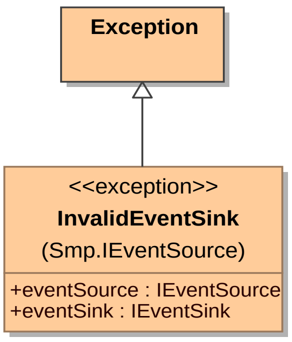


Figure 3-25: Invalid Event Sink

Fields

Table 3-37: Fields of InvalidEventSink

Name	Type	Description
eventSink	IEventSink	Event sink that is not of valid type.
eventSource	IEventSource	Event source the event sink is subscribed to.

3.3.3.3.5 Not Subscribed

This exception is raised when trying to unsubscribe an event sink from an event source that is not subscribed to it.

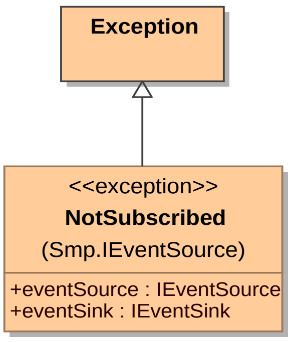


Figure 3-26: Not Subscribed

Fields

Table 3-38: Fields of NotSubscribed

Name	Type	Description
eventSink	IEventSink	Event sink that is not subscribed.
eventSource	IEventSource	Event source the event sink is not subscribed to.

3.3.3.4 Event Source Collection

An event source collection is an ordered collection of event sources, which allows iterating all members.

This type is platform specific. For details see the SMP Platform Mappings.

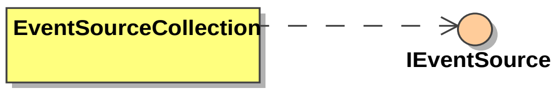


Figure 3-27: Event Source Collection

3.3.4 Entry Points

An entry point is an interface that exposes a void function with no return value that can be called by the scheduler or event manager service.

3.3.4.1 ITask

Interface for a task, which is an ordered collection of entry points.

This interface extends IEntryPoint to allow executing a number of entry points in one operation.

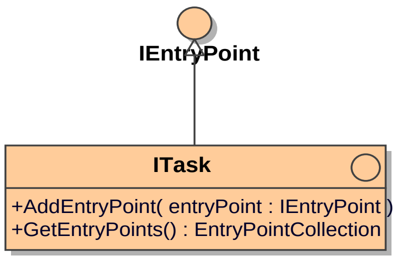

IEntryPoint
ITask
+AddEntryPoint(entryPoint : IEntryPoint)
+GetEntryPoints() : EntryPointCollection

Figure 3-28: ITask

Base Interfaces

Smp::IEntryPoint

Operations

Table 3-39: Operations of ITask

Name	Description
AddEntryPoint	Add an entry point to the task.
GetEntryPoints	Query for the collection of all entry points. The order of entry points in the collection is the order in which they have been added to the task.

3.3.4.1.1 Add Entry Point

Add an entry point to the task.

Entry points in a task will be executed in the order they have been added.

Prototype

public void AddEntryPoint(in IEntryPoint entryPoint);

Parameters

Table 3-40: Parameters of AddEntryPoint

Name	Dir.	Type	Description
entryPoint	in	IEntryPoint	Entry point to add to task.

Exceptions

None

3.3.4.1.2 Get Entry Points

Query for the collection of all entry points. The order of entry points in the collection is the order in which they have been added to the task.

The returned collection may be empty if no entry points have been added to the task.

Prototype

public EntryPointCollection GetEntryPoints();

Parameters

Table 3-41: Parameters of GetEntryPoints

Name	Dir.	Type	Description
	return	EntryPointCollection	Collection of entry points.

Exceptions

None

3.3.4.2 IEntry Point

Interface of an entry point.

This interface provides a notification method (event handler) that can be called e.g. by the Scheduler or Event Manager when an event is emitted.

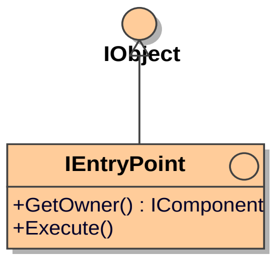


Figure 3-29: IEntry Point

Base Interfaces

Smp::IObject

Operations

Table 3-42: Operations of IEntryPoint

Name	Description
Execute	This method is called when an associated event is emitted.
GetOwner	This method returns the Component that owns the entry point.

3.3.4.2.1 Execute

This method is called when an associated event is emitted.

Components providing entry points must ensure that these entry points do not throw exceptions.

Prototype

public void Execute();

Parameters

None

Exceptions

None

3.3.4.2.2 Get Owner

This method returns the Component that owns the entry point.

Remark: This is required to be able to store and restore entry points.

Prototype

public IComponent GetOwner();

Parameters

Table 3-43: Parameters of GetOwner

Name	Dir.	Type	Description
	return	IComponent	Owner of entry point.

Exceptions

None

3.3.4.3 Entry Point Collection

An entry point collection is an ordered collection of entry points, which allows iterating all members.

This type is platform specific. For details see the SMP Platform Mappings.

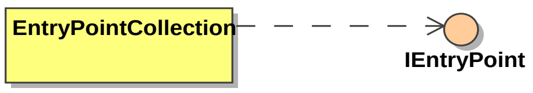


Figure 3-30: Entry Point Collection

3.3.5 Dynamic Invocation

Dynamic invocation is a mechanism that makes the operations of a component available via a standardised interface (as opposed to a custom interface of the component which is not known at compile time of the simulation environment). In order to allow calling a named method with any number of parameters, a request object has to be created which contains all information needed for the method invocation. This request object is as well used to transfer back a return value of the operation.

The dynamic invocation concept presented here standardises the request objects (IRequest interface). In addition, two methods are provided as part of IDynamicInvocation to create and delete request objects. However, it is not mandatory to use these methods, as request objects can well be created and deleted using another implementation. A reason for doing this could be to minimise the number of round-trips between a client (that calls a method) and a component that implements IDynamicInvocation. The sequence diagram in Figure 3-31 shows all steps involved when using the CreateRequest() and DeleteRequest() methods.

Remark: Especially when running distributed components, this implementation is slow. In these cases, the request object should be created by the client, and only passed to the component on Invoke().

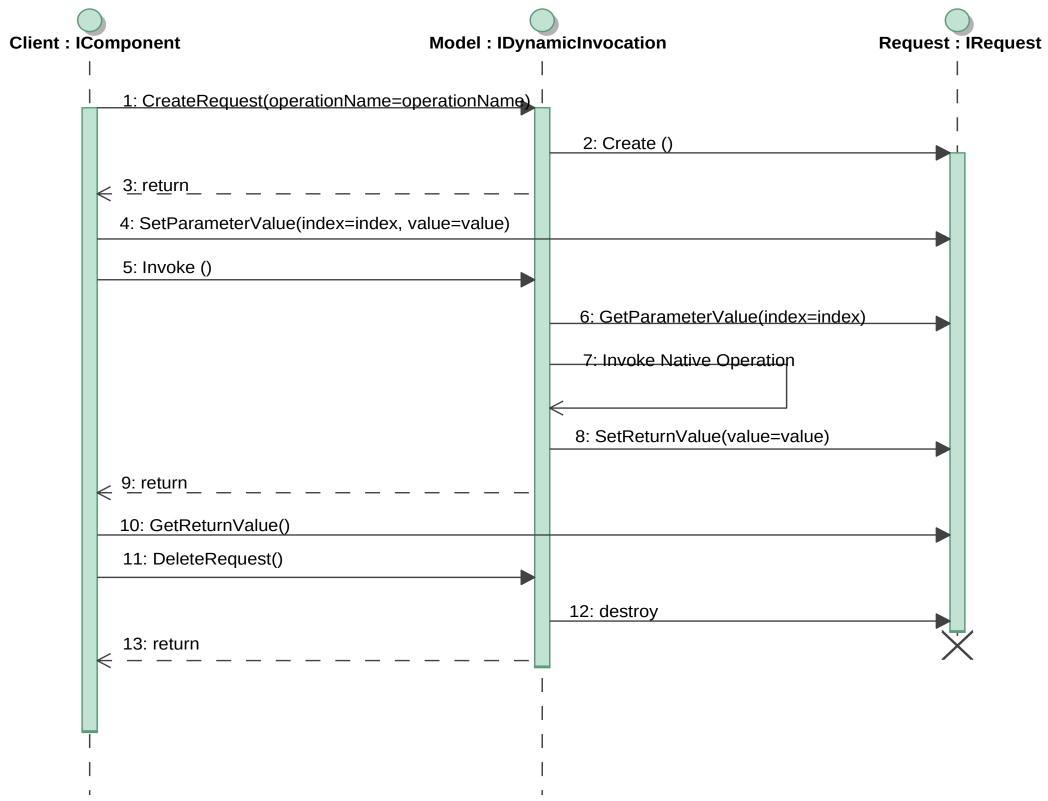

Client : IComponent
Model : IDynamicInvocation
Request : IRequest
1: CreateRequest(operationName=operationName)
2: Create ()
3: return
4: SetParameterValue(index=index, value=value)
5: Invoke ()
6: GetParameterValue(index=index)
7: Invoke Native Operation
8: SetReturnValue(value=value)
9: return
10: GetReturnValue()
11: DeleteRequest()
12: destroy
13: return

Figure 3-31: Sequence of calls for dynamic invocation

The sequence diagram in Figure 3-31, using a Client component and a Model implementing IDynamicInvocation, contains the following steps:

1. The client calls the CreateRequest() operation of the component to create a request object for the operation, passing it the name of the operation.
2. The component creates a request object for the operation, using the default values of all parameters.
3. The component returns the Request object via its IRequest interface to the client.
4. The client calls the SetParameterValue() operation of the Request object to set parameters to non-default values.
5. The client calls the Invoke() operation of the component to invoke the corresponding operation.
6. The component calls the GetParameterValue() operation of the Request object to get parameters.
7. The component calls its internal operation that corresponds to the invoked operation.
8. The component calls the SetReturnValue() operation of the Request object to set the return value.
9. The component returns control to the client.
10. The client calls the GetReturnValue() operation of the Request object to get the return value.

11. The client calls the DeleteRequest() operation of the component to delete the Request object.
12. The component destroys the request object.
13. The component returns control to the client.

Like all mechanisms in this section, dynamic invocation is an optional mechanism.

3.3.5.1 IDynamic Invocation

Interface for a component that supports dynamic invocation of operations.

A component may implement this interface in order to allow dynamic invocation of its operations.

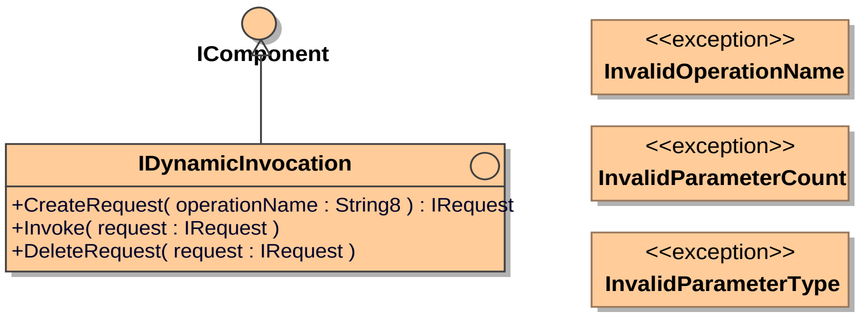


Figure 3-32: IDynamic Invocation

Remark: Dynamic invocation is typically used for scripting.

Base Interfaces

Smp::IComponent

Operations

Table 3-44: Operations of IDynamicInvocation

Name	Description
CreateRequest	Return a request object for the given operation that describes the parameters and the return value.
DeleteRequest	Destroy a request object that has been created with the CreateRequest() method before.
Invoke	Dynamically invoke an operation using a request object that has been created and filled with parameter values by the caller.

3.3.5.1.1 Create Request

Return a request object for the given operation that describes the parameters and the return value.

The request object may be null if no operation with the given name could be found, or if the operation with the given name does not support dynamic invocation.

Prototype

public IRequest CreateRequest(in String8 operationName);

Parameters

Table 3-45: Parameters of CreateRequest

Name	Dir.	Type	Description
	return	IRequest	Request object for operation, or null if either no operation with the given name could be found, or the operation with the given name does not support dynamic invocation.
operationName	in	String8	Name of operation.

Exceptions

None

3.3.5.1.2 Delete Request

Destroy a request object that has been created with the CreateRequest() method before.

The request object must not be used anymore after DeleteRequest() has been called for it.

Prototype

public void DeleteRequest(in IRequest request);

Parameters

Table 3-46: Parameters of DeleteRequest

Name	Dir.	Type	Description
request	in	IRequest	Request object to destroy.

Exceptions

None

3.3.5.1.3 Invoke

Dynamically invoke an operation using a request object that has been created and filled with parameter values by the caller.

This method raises the InvalidOperationName exception if the request object passed does not name an operation that allows dynamic invocation. When calling invoke with a wrong number of parameters, the InvalidParameterCount exception is raised. When passing a parameter of wrong type, the InvalidParameterType exception is raised.

Remark: The same request object can be used to invoke a method several times.

Prototype

public void Invoke(in IRequest request) raises (InvalidOperationName, InvalidParameterCount, InvalidParameterType);

Parameters

Table 3-47: Parameters of Invoke

Name	Dir.	Type	Description
request	in	IRequest	Request object to invoke.

Exceptions

Smp::IDynamicInvocation::InvalidOperationName,
Smp::IDynamicInvocation::InvalidParameterCount,
Smp::IDynamicInvocation::InvalidParameterType

3.3.5.1.4 Invalid Operation Name

This exception is raised by the Invoke() method when trying to invoke a method that does not exist, or that does not support dynamic invocation.

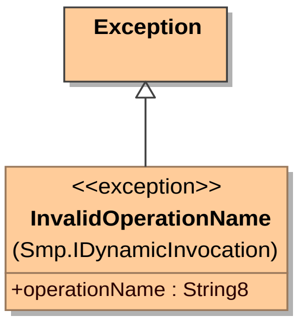


Figure 3-33: Invalid Operation Name

Fields

Table 3-48: Fields of InvalidOperationName

Name	Type	Description
operationName	String8	Operation name of request passed to the Invoke() method.

3.3.5.1.5 Invalid Parameter Count

This exception is raised by the Invoke() method when trying to invoke a method with a wrong number of parameters.

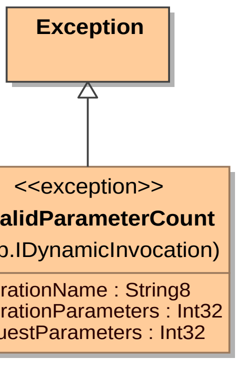


Figure 3-34: Invalid Parameter Count

Fields

Table 3-49: Fields of InvalidParameterCount

Name	Type	Description
operationName	String8	Operation name of request passed to the Invoke() method.
operationParameters	Int32	Correct number of parameters of operation.
requestParameters	Int32	Wrong number of parameters of operation.

3.3.5.1.6 Invalid Parameter Type

This exception is raised by the Invoke() method when trying to invoke a method passing a parameter of wrong type.

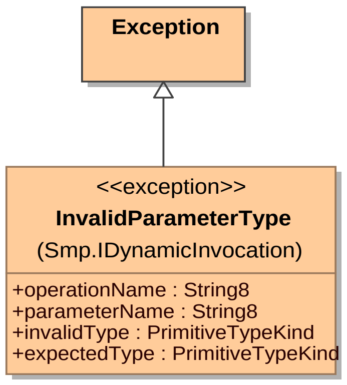


Figure 3-35: Invalid Parameter Type

Remark: The index of the parameter of wrong type can be extracted from the request using the method GetParameterIndex().

Fields

Table 3-50: Fields of InvalidParameterType

Name	Type	Description
expectedType	PrimitiveTypeKind	Type that was expected.
invalidType	PrimitiveTypeKind	Type that is not valid.
operationName	String8	Operation name of request passed to the Invoke() method.
parameterName	String8	Name of parameter of wrong type.

3.3.5.2 IRequest

A request holds information, which is passed between a client invoking an operation via the IDynamicInvocation interface and a component being invoked.

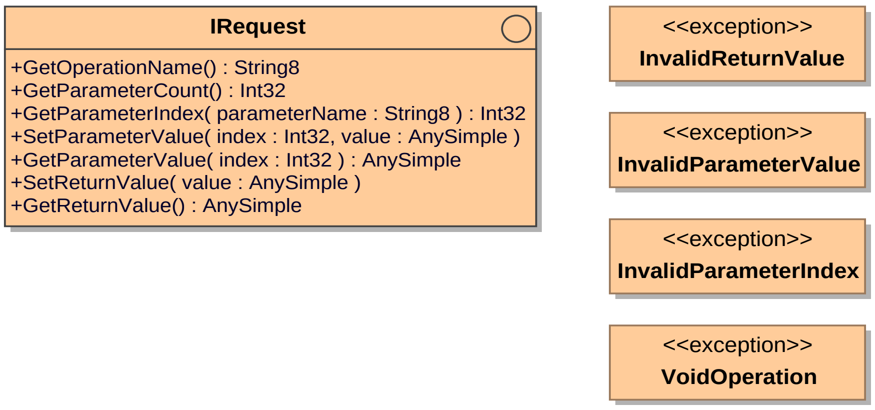

IRequest
+GetOperationName() : String8
+GetParameterCount() : Int32
+GetParameterIndex(parameterName : String8) : Int32
+SetParameterValue(index : Int32, value : AnySimple)
+GetParameterValue(index : Int32) : AnySimple
+SetReturnValue(value : AnySimple)
+GetReturnValue() : AnySimple
<<exception>> InvalidReturnValue
<<exception>> InvalidParameterValue
<<exception>> InvalidParameterIndex
<<exception>> VoidOperation

Figure 3-36: IRequest

Base Interfaces

None

Operations

Table 3-51: Operations of IRequest

Name	Description
GetOperationName	Return the name of the operation that this request is for.
GetParameterCount	Return the number of parameters stored in the request.
GetParameterIndex	Query for a parameter index by parameter name.
GetParameterValue	Query a value of a parameter at a given position.
GetReturnValue	Query the return value of the operation.
SetParameterValue	Assign a value to a parameter at a given position.
SetReturnValue	Assign the return value of the operation.

3.3.5.2.1 Get Operation Name

Return the name of the operation that this request is for.

Remark: A request is typically created using the CreateRequest() method to dynamically call a specific method of a component implementing the IDynamicInvocation interface. This method returns the name passed to it, to allow finding out which method is actually called on Invoke().

Prototype

public String8 GetOperationName();

Parameters

Table 3-52: Parameters of GetOperationName

Name	Dir.	Type	Description
	return	String8	Name of the operation.

Exceptions

None

3.3.5.2.2 Get Parameter Count

Return the number of parameters stored in the request.

Parameters can be accessed by their 0-based index. This index

- must not be negative,
- must be smaller than the parameter count.

Remark: The GetParameterIndex() method may be used to access parameters by name.

Prototype

public Int32 GetParameterCount();

Parameters

Table 3-53: Parameters of GetParameterCount

Name	Dir.	Type	Description
	return	Int32	Number of parameters in request object.

Exceptions

None

3.3.5.2.3 Get Parameter Index

Query for a parameter index by parameter name.

The index values are 0-based. An index of -1 indicates a wrong parameter name.

Prototype

public Int32 GetParameterIndex(in String8 parameterName);

Parameters

Table 3-54: Parameters of GetParameterIndex

Name	Dir.	Type	Description
	return	Int32	Index of parameter with the given name, or -1 if no parameter with the given name could be found.
parameterName	in	String8	Name of parameter.

Exceptions

None

3.3.5.2.4 Get Parameter Value

Query a value of a parameter at a given position.

This method raises an exception of type InvalidParameterIndex if called with an illegal parameter index.

Prototype

public AnySimple GetParameterValue(in Int32 index) raises (InvalidParameterIndex);

Parameters

Table 3-55: Parameters of GetParameterValue

Name	Dir.	Type	Description
	return	AnySimple	Value of parameter.
index	in	Int32	Index of parameter (0-based).

Exceptions

Smp::IRequest::InvalidParameterIndex

3.3.5.2.5 Get Return Value

Query the return value of the operation.

This method raises an exception of type VoidOperation if called for a request object of a void operation.

Prototype

public AnySimple GetReturnValue() raises (VoidOperation);

Parameters

Table 3-56: Parameters of GetReturnValue

Name	Dir.	Type	Description
	return	AnySimple	Return value of the operation.

Exceptions

Smp::IRequest::VoidOperation

3.3.5.2.6 Set Parameter Value

Assign a value to a parameter at a given position.

This method raises an exception of type InvalidParameterIndex if called with an illegal parameter index. If called with an invalid parameter type, it raises an exception of type InvalidAnyType. If called with an invalid value for the parameter, it raises an exception of type InvalidParameterValue.

Prototype

public void SetParameterValue(in Int32 index, in AnySimple value) raises (InvalidAnyType, InvalidParameterIndex, InvalidParameterValue);

Parameters

Table 3-57: Parameters of SetParameterValue

Name	Dir.	Type	Description
index	in	Int32	Index of parameter (0-based).
value	in	AnySimple	Value of parameter.

Exceptions

Smp::InvalidAnyType, Smp::IRequest::InvalidParameterIndex, Smp::IRequest::InvalidParameterValue

3.3.5.2.7 Set Return Value

Assign the return value of the operation.

This method raises an exception of type VoidOperation if called for a request object of a void operation. If called with an invalid return type, it raises an exception of type InvalidAnyType. If called with an invalid value for the return type, this method raises an exception of type InvalidReturnValue.

Prototype

public void SetReturnValue(in AnySimple value) raises (InvalidAnyType, InvalidReturnValue, VoidOperation);

Parameters

Table 3-58: Parameters of SetReturnValue

Name	Dir.	Type	Description
value	in	AnySimple	Return value.

Exceptions

Smp::InvalidAnyType, Smp::IRequest::InvalidReturnValue, Smp::IRequest::VoidOperation

3.3.5.2.8 Invalid Parameter Index

This exception is raised when using an invalid parameter index to set (SetParameterValue()) or get (GetParameterValue()) a parameter value of an operation in a request.

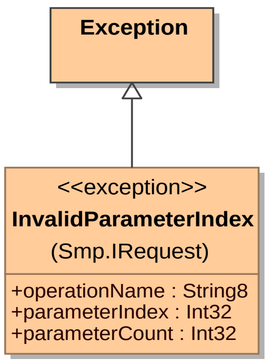


Figure 3-37: Invalid Parameter Index

Fields

Table 3-59: Fields of InvalidParameterIndex

Name	Type	Description
operationName	String8	Name of operation.
parameterCount	Int32	Number of parameters of the operation.
parameterIndex	Int32	Invalid parameter index used.

3.3.5.2.9 Invalid Parameter Value

This exception is raised when trying to assign an illegal value to a parameter of an operation in a request using SetParameterValue().

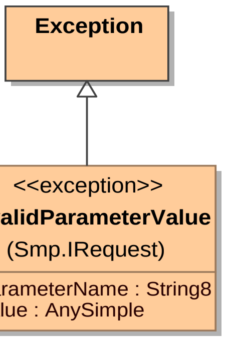


Figure 3-38: Invalid Parameter Value

Fields

Table 3-60: Fields of InvalidParameterValue

Name	Type	Description
parameterName	String8	Name of parameter value was assigned to.
value	AnySimple	Value that was passed as parameter.

3.3.5.2.10 Invalid Return Value

This exception is raised when trying to assign an invalid return value of an operation in a request using SetReturnValue().

Fields

Table 3-61: Fields of InvalidReturnValue

Name	Type	Description
operationName	String8	Name of operation the return value was assigned to.
value	AnySimple	Value that was passed as return value.

3.3.5.2.11 Void Operation

This exception is raised when trying to read (GetReturnValue()) or write (SetReturnValue()) the return value of a void operation.

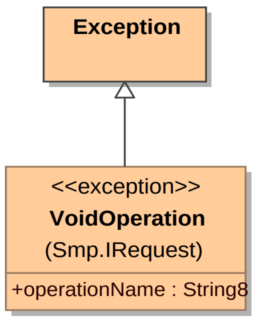


Figure 3-39: Void Operation

Fields

Table 3-62: Fields of VoidOperation

Name	Type	Description
operationName	String8	Name of operation.

3.3.6 Persistence

Persistence of SMP components can be handled in one of two ways:

1. **External Persistence**: The simulation environment stores and restores the model's state by directly accessing the fields that are published to the simulation environment, i.e. via the IPublication interface.

   Remark: This should be the preferred mechanism for the majority of models.

2. **Self-Persistence**: The component may implement the IPersist interface, which allows it to store and restore (part of) its state into or from storage that is provided by the simulation environment.

   Remark: This mechanism is usually only needed by specialised models, for example embedded models that need to load on-board software from a specific file. Further, this mechanism can be used by simulation services if desired. For example, the Scheduler service may use it to store and restore its current state.

Like all mechanisms in this section, self-persistence of components is an optional mechanism, while external persistence (via the Store() and Restore() methods of the ISimulator interface) is a mandatory mechanism of every SMP simulation environment.

3.3.6.1 IPersist

Interface of a self-persisting component that provides operations to allow for storing and restoring its state.

A component may implement this interface if it wants to have control over storing and restoring of its state. This is an optional interface which needs to be implemented by components with self-persistence only.

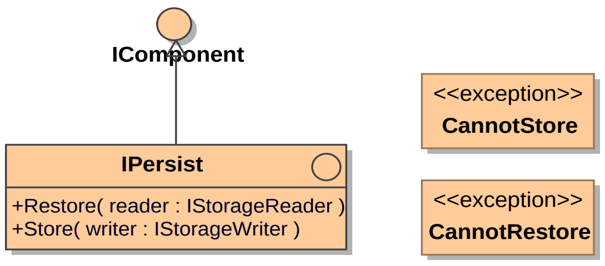

IComponent
IPersist
+Restore(reader : IStorageReader)
+Store(writer : IStorageWriter)
<<exception>>
CannotStore
<<exception>>
CannotRestore

Figure 3-40: IPersist

Base Interfaces

Smp::IComponent

Operations

Table 3-63: Operations of IPersist

Name	Description
Restore	Restore component state from storage.
Store	Store component state to storage.

3.3.6.1.1 Restore

Restore component state from storage.

This method raises an exception of type CannotRestore if reading data from the storage reader fails.

Prototype

public void Restore(in IStorageReader reader) raises (CannotRestore);

Parameters

Table 3-64: Parameters of Restore

Name	Dir.	Type	Description
reader	in	IStorageReader	Interface that allows reading from storage.

Exceptions

Smp::IPersist::CannotRestore

3.3.6.1.2 Store

Store component state to storage.

This method raises an exception of type CannotStore if writing data to the storage writer fails.

Prototype

public void Store(in IStorageWriter writer) raises (CannotStore);

Table 3-65: Parameters of Store

Name	Dir.	Type	Description
writer	in	IStorageWriter	Interface that allows writing to storage.

Exceptions

Smp::IPersist::CannotStore

3.3.6.1.3 Cannot Restore

This exception is raised when the content of the storage reader passed to the Restore() method contains invalid data.

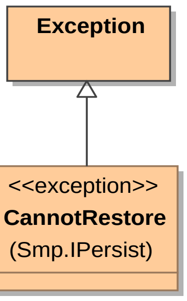


Figure 3-41: Cannot Restore

Remark: This typically happens when a Store() has been created from a different configuration of components.

Fields

None

3.3.6.1.4 Cannot Store

This exception is raised when the component cannot store its data to the storage writer given to the Store() method.

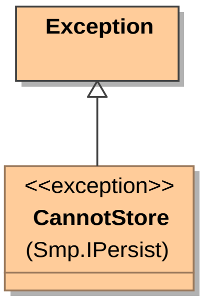


Figure 3-42: Cannot Store

Remark: This may e.g. be if there is no disk space left.

Fields

None

3.3.6.2 IStorage Reader

This interface provides functionality to read data from storage.

This interface is platform specific. For details see the SMP Platform Mappings.

Figure 3-43: IStorage Reader

Remark: A client (typically the simulation environment) provides this interface to allow components implementing the IPersist interface to restore their state. It is passed to the Restore() method of every model implementing IPersist.

Base Interfaces

None

Operations

None

3.3.6.3 IStorage Writer

This interface provides functionality to write data to storage.

This interface is platform specific. For details see the SMP Platform Mappings.

Figure 3-44: IStorage Writer

Remark: A client (typically the simulation environment) provides this interface to allow components implementing the IPersist interface to store their state. It is passed to the Store() method of every model implementing IPersist.

Base Interfaces

None

Operations

None

3.4 Model Mechanisms

While the IModel interface defines the mandatory functionality every SMP model has to provide, this section introduces additional mechanisms available for more advanced use.

3.4.1 Fallible Models

Fallible models expose their failure state and a collection of failures.

3.4.1.1 IFailure

Interface for a failure.

A Failure allows to query and to set its state to Failed or Unfailed.

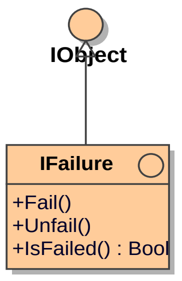


Figure 3-45: IFailure

Base Interfaces

Smp::IObject

Table 3-66: Operations of IFailure

Name	Description
Fail	Sets the state of the failure to failed.
IsFailed	Returns whether the failure's state is set to failed.
Unfail	Sets the state of the failure to unfailed.

3.4.1.1.1 Fail

Sets the state of the failure to failed.

Prototype

public void Fail();

Parameters

None

Exceptions

None

3.4.1.1.2 Is Failed

Returns whether the failure's state is set to failed.

Prototype

public Bool IsFailed();

Table 3-67: Parameters of IsFailed

Name	Dir.	Type	Description
	return	Bool	Returns true if the failure state is Failed, false otherwise.

Exceptions

None

3.4.1.1.3 Unfail

Sets the state of the failure to unfailed.

Prototype

public void Unfail();

Parameters

None

Exceptions

None

3.4.1.2 IFallible Model

Interface for a fallible model that exposes its failure state and a collection of failures.

A fallible model allows querying for its failures by name.

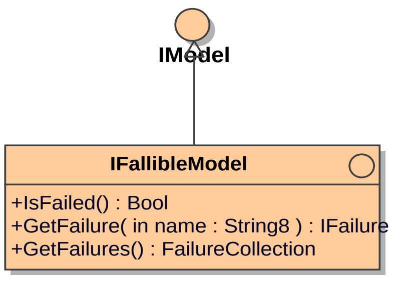

IModel
IFallibleModel
+IsFailed() : Bool
+GetFailure(in name : String8) : IFailure
+GetFailures() : FailureCollection

Figure 3-46: IFallible Model

Base Interfaces

Smp::IModel

Table 3-68: Operations of IFaillibleModel

Name	Description
GetFailure	Get a failure by name.
GetFailures	Query for the collection of all failures.
IsFailed	Query for whether the model is failed. A model is failed when at least one of its failures is failed.

3.4.1.2.1 Get Failure

Get a failure by name.

The returned failure may be null if no child with the given name could be found.

Prototype

public IFailure GetFailure(in String8 name);

Table 3-69: Parameters of IFailure

Name	Dir.	Type	Description
name	in	String8	Name of the failure to return.
	return	IFailure	Failure queried for by name, or null if no failure with this name exists.

Exceptions

None

3.4.1.2.2 Get Failures

Query for the collection of all failures.

The returned collection may be empty if no failures exist for the fallible model.

Prototype

public FailureCollection GetFailures();

Table 3-70: Parameters of GetFailures

Name	Dir.	Type	Description
	return	FailureCollection	Failure collection of the model.

Exceptions

None

3.4.1.2.3 Is Failed

Query for whether the model is failed. A model is failed when at least one of its failures is failed.

Prototype

public Bool IsFailed();

Table 3-71: Parameters of IsFailed

Name	Dir.	Type	Description
	return	Bool	Whether the model is failed or not.

Exceptions

None

3.5 Management Interfaces

Managed interfaces allow external components to access all mechanisms by name. This includes the basic component features, optional component mechanisms and optional model mechanisms.

Managed interfaces allow full access to all functionality of components. For composition and aggregation, they extend the existing interfaces by methods to add new components or references, respectively. For entry points, event sources and event sinks, the managed interfaces provide access to the elements by name. For fields, access by name is provided by an extended model interface.

All management interfaces are optional, and only need to be provided for models used in a managed environment. Typically, in a managed environment a model configuration is built from an XML document (namely an SMDL Assembly) during the Creating phase.

Remark: Typically, a Loader or Model Manager component calls these interfaces to push configuration information into the models, which has been read from an SMDL Assembly file. Initialisation of models from an SMDL Configuration file does not require managed models.

3.5.1 Managed Components

Managed components provide write access to their properties, i.e. they provide corresponding "setter" methods for the Name, Description, and Parent properties. This allows putting them into a hierarchy with a given name and description.

3.5.1.1 IManaged Object

Interface of a managed object.

A managed object additionally allows assigning name and description.

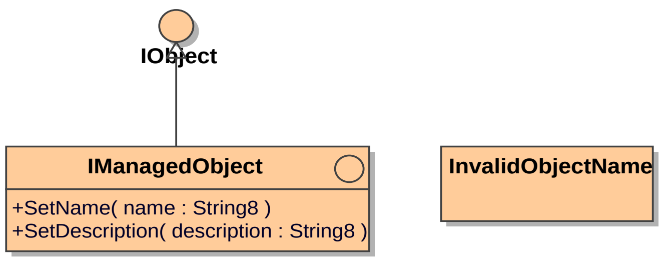

IObject
IManagedObject
+SetName(name : String8)
+SetDescription(description : String8)
InvalidObjectName

Figure 3-47: IManaged Object

Base Interfaces

Smp::IObject

Table 3-72: Operations of IManagedObject

Name	Description
SetDescription	Define the description of the managed object ("property setter").
SetName	Define the name of the managed object ("property setter").

3.5.1.1.1 Set Description

Define the description of the managed object ("property setter").

Prototype

public void SetDescription(in String8 description);

Table 3-73: Parameters of SetDescription

Name	Dir.	Type	Description
description	in	String8	Description of object.

Exceptions

None

3.5.1.1.2 Set Name

Define the name of the managed object ("property setter").

This method throws an exception of type InvalidObjectName when the given name is not valid.

Names must

- be unique within their context,
- not be empty,
- start with a letter, and
- only contain letters, digits, the underscore ("_") and brackets ("[" and "]").

Remark: Except for the first rule (uniqueness of names), the SetName() method should test for all other rules.

Remark: It is recommended that names do not exceed 32 characters in size.

Prototype

public void SetName(in String8 name) raises (InvalidObjectName);

Table 3-74: Parameters of SetName

Name	Dir.	Type	Description
name	in	String8	Name of object.

Exceptions

Smp::Management::IManagedObject::InvalidObjectName

3.5.1.1.3 Invalid Object Name

This exception is raised when trying to set an object's name to an invalid name. Names

- must not be empty,
- must start with a letter, and
- must only contain letters, digits, the underscore ("_") and brackets ("[" and "]").

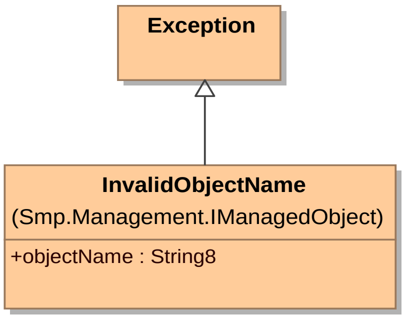


Figure 3-48: Invalid Object Name

Table 3-75: Fields of InvalidObjectName

Name	Type	Description
objectName	String8	Invalid object name passed to SetName().

3.5.1.2 IManaged Component

Interface of a managed component.

A managed component additionally allows assigning the parent.

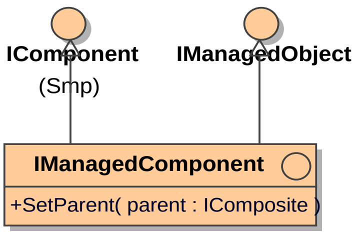


Figure 3-49: IManaged Component

Base Interfaces

Smp::Management::IManagedObject, Smp::IComponent

Table 3-76: Operations of IManagedComponent

Name	Description
SetParent	Define the parent component ("property setter"). Components link to their parent to allow traversing the tree of components upwards.

3.5.1.2.1 Set Parent

Define the parent component ("property setter"). Components link to their parent to allow traversing the tree of components upwards.

Prototype

public void SetParent(in IComposite parent);

Table 3-77: Parameters of SetParent

Name	Dir.	Type	Description
parent	in	IComposite	Parent composite of component.

Exceptions

None

3.5.2 Managed Component Mechanisms

The component mechanisms introduced in section 3.3 (Component Mechanisms) do not provide full access for external components, but only limited access:

- Aggregation provides collections of references, but does not allow adding new references to a Reference.
- Composition provides a tree of components, but does not allow adding new components to a Container.
- Event sinks can be connected by models if they have access to event sources, but an external component can not query for event sinks and event sources by name.
- Entry points can be registered with services by models, but an external component can not query for entry points by name.

To overcome these limitations, managed interfaces are provided with full access to all functionality. For composition and aggregation, these extend the existing interfaces by methods to add new components or references, respectively. For event sources and event sinks, the managed interfaces provide access to the elements by name. For entry points, the managed interface provides access to the entry points by name.

3.5.2.1 IComponent Collection

Base interface for managed collections, which are either references or containers.

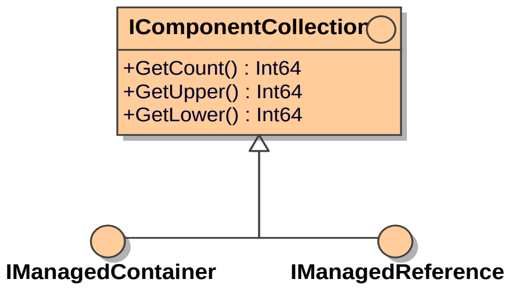


Figure 3-50: IComponent Collection

Base Interfaces

None

Table 3-78: Operations of IComponentCollection

Name	Description
GetCount	Query for the number of components in the collection.
GetLower	Query the minimum number of components in the collection.
GetUpper	Query the maximum number of components in the collection.

3.5.2.1.1 Get Count

Query for the number of components in the collection.

Prototype

public Int64 GetCount();

Table 3-79: Parameters of GetCount

Name	Dir.	Type	Description
	return	Int64	Current number of components in the collection.

Exceptions

None

3.5.2.1.2 Get Lower

Query the minimum number of components in the collection.

Remark: This information can be used to validate a model hierarchy. If a collection specifies a Lower value above its current Count, then it is not properly configured. An external component may use this information to validate the configuration before executing it.

Prototype

public Int64 GetLower();

Table 3-80: Parameters of GetLower

Name	Dir.	Type	Description
	return	Int64	Minimum number of components in the collection.

Exceptions

None

3.5.2.1.3 Get Upper

Query the maximum number of components in the collection.

A return value of -1 indicates that the collection has no upper limit.

Remark: This information can be used to check whether another component can be added to the collection.

Remark: This is consistent with the use of upper bounds in UML, where a value of -1 represents no limit (typically shown as *).

Prototype

public Int64 GetUpper();

Table 3-81: Parameters of GetUpper

Name	Dir.	Type	Description
	return	Int64	Maximum number of components in the collection. (-1 = unlimited).

Exceptions

None

3.5.2.2 IManaged Reference

Interface of a managed reference.

A managed reference additionally allows querying the size limits and adding and removing referenced components.

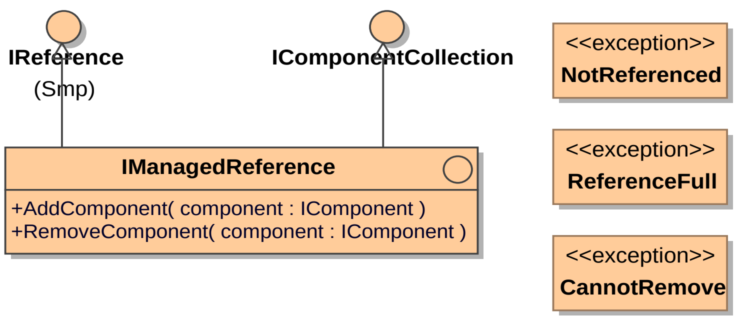


Figure 3-51: IManaged Reference

Base Interfaces

Smp::Management::IComponentCollection, Smp::IReference

Table 3-82: Operations of IManagedReference

Name	Description
AddComponent	Add a referenced component.
RemoveComponent	Remove a referenced component.

3.5.2.2.1 Add Component

Add a referenced component.

This method raises an exception of type ReferenceFull if called for a full reference, i.e. when the Count has reached the Upper limit. This method may raise an exception of type InvalidObjectType when it expects the given component to implement another interface as well.

Adding a component with a name that already exists in the reference does not throw an exception, although GetComponent() will no longer allow to return both referenced components by name.

Prototype

public void AddComponent(in IComponent component) raises (InvalidObjectType, ReferenceFull);

Table 3-83: Parameters of AddComponent

Name	Dir.	Type	Description
component	in	IComponent	New referenced component.

Exceptions

Smp::InvalidObjectType,
Smp::Management::IManagedReference::ReferenceFull

3.5.2.2.2 Remove Component

Remove a referenced component.

This method raises an exception of type NotReferenced if called with a component that is not referenced. If the number of referenced components is less than or equal to the Lower limit, this method raises an exception of type CannotRemove.

Prototype

public void RemoveComponent(in IComponent component) raises (CannotRemove, NotReferenced);

Table 3-84: Parameters of RemoveComponent

Name	Dir.	Type	Description
component	in	IComponent	Referenced component to remove.

Exceptions

Smp::Management::IManagedReference::CannotRemove,
Smp::Management::IManagedReference::NotReferenced

3.5.2.2.3 Cannot Remove

This exception is thrown when trying to remove a component from a reference when the number of referenced components is lower than or equal to the Lower limit.

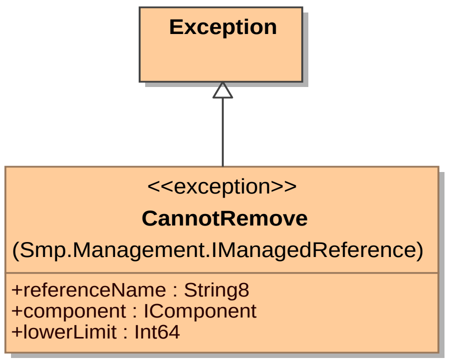


Figure 3-52: Cannot Remove

Table 3-85: Fields of CannotRemove

Name	Type	Description
component	IComponent	Component that could not be removed.
lowerLimit	Int64	Lower limit of the reference.
referenceName	String8	Name of reference.

3.5.2.2.4 Not Referenced

This exception is thrown when trying to remove a component from a reference which was not referenced before.

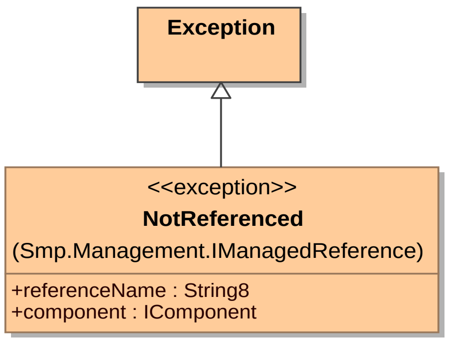


Figure 3-53: Not Referenced

Table 3-86: Fields of NotReferenced

Name	Type	Description
component	IComponent	Component that is not referenced.
referenceName	String8	Name of reference.

3.5.2.2.5 Reference Full

This exception is raised when trying to add a component to a reference that is full, i.e. where the Count has reached the Upper limit.

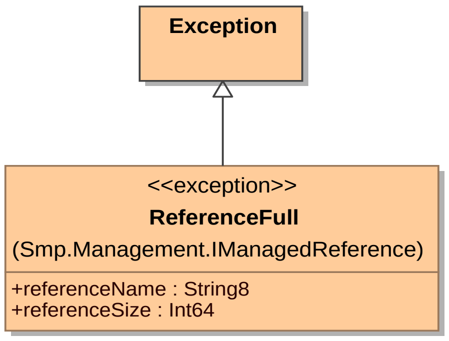


Figure 3-54: Reference Full

Table 3-87: Fields of ReferenceFull

Name	Type	Description
referenceName	String8	Name of reference.
referenceSize	Int64	Number of components in the reference, which is its Upper limit when the reference is full.

3.5.2.3 IManaged Container

Interface of a managed container.

A managed container additionally allows querying the size limits and adding contained components.

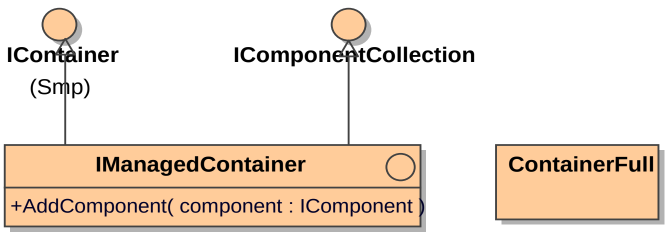


Figure 3-55: IManaged Container

Base Interfaces

Smp::Management::IComponentCollection, Smp::IContainer

Table 3-88: Operations of IManagedContainer

Name	Description
AddComponent	Add a contained component to the container.

3.5.2.3.1 Add Component

Add a contained component to the container.

This method raises an exception of type ContainerFull if called for a full container, i.e. when the Count has reached the Upper limit. It raises an exception of type DuplicateName when trying to add a component with a name that is already contained in the container, as this would lead to duplicate names in the container. This method may raise an exception of type InvalidObjectType when it expects the given component to implement another interface as well.

Prototype

public void AddComponent(in IComponent component) raises (DuplicateName, InvalidObjectType, ContainerFull);

Table 3-89: Parameters of AddComponent

Name	Dir.	Type	Description
component	in	IComponent	New contained component.

Exceptions

Smp::DuplicateName, Smp::InvalidObjectType, Smp::Management::IManagedContainer::ContainerFull

3.5.2.3.2 Container Full

This exception is raised when trying to add a component to a container that is full, i.e. where the Count has reached the Upper limit.

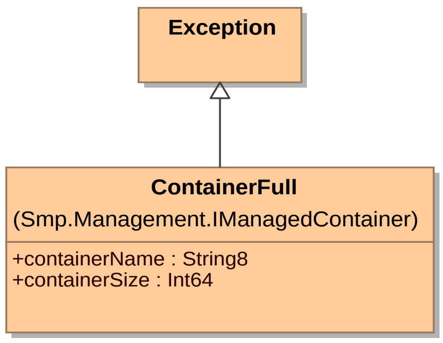


Figure 3-56: Container Full

Table 3-90: Fields of ContainerFull

Name	Type	Description
containerName	String8	Name of full container.
containerSize	Int64	Number of components in the container, which is its Upper limit when the container is full.

3.5.2.4 IEvent Consumer

Interface of an event consumer.

An event sonsumer is a omponent that holds event sinks, which may be subscribed to other component's event sources.

This is an optional interface. It needs to be implemented for managed components only, which want to allow access to event sinks by name.

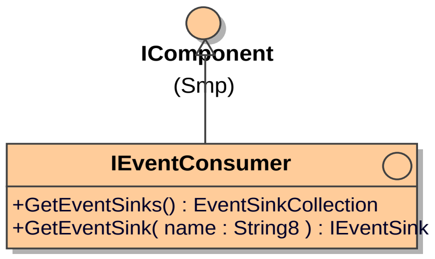

IComponent
(Smp)
IEventConsumer
+GetEventSinks() : EventSinkCollection
+GetEventSink(name : String8) : IEventSink

Figure 3-57: IEvent Consumer

Base Interfaces

Smp::IComponent

Table 3-91: Operations of IEventConsummer

Name	Description
GetEventSink	Query for an event sink of this component by its name.
GetEventSinks	Query for the collection of all event sinks of the component.

3.5.2.4.1 Get Event Sink

Query for an event sink of this component by its name.

The returned event sink may be null if no event sink with the given name could be found.

Prototype

public IEventSink GetEventSink(in String8 name);

Table 3-92: Parameters of GetEventSink

Name	Dir.	Type	Description
	return	IEventSink	Event sink with the given name, or null if no event sink with the given name could be found.
name	in	String8	Event sink name.

Exceptions

None

3.5.2.4.2 Get Event Sinks

Query for the collection of all event sinks of the component.

The collection may be empty if no event sinks exist.

Prototype

public EventSinkCollection GetEventSinks();

Table 3-93: Parameters of GetEventSinks

Name	Dir.	Type	Description
	return	EventSinkCollection	Collection of event sinks.

Exceptions

None

3.5.2.5 IEvent Provider

Interface of an event provider.

An event provider is a omponent that holds event sources, which allow other components to subscribe their event sinks.

This is an optional interface. It needs to be implemented for managed components only, which want to allow access to event sources by name.

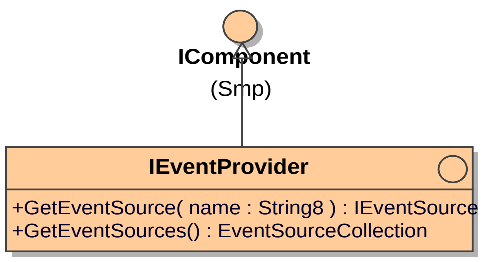

IComponent
(Smp)
IEventProvider
+GetEventSource(name : String8) : IEventSource
+GetEventSources() : EventSourceCollection

Figure 3-58: IEvent Provider

Base Interfaces

Smp::IComponent

Table 3-94: Operations of IEventProvider

Name	Description
GetEventSource	Query for an event source of this component by its name.
GetEventSources	Query for the collection of all event sources of the component.

3.5.2.5.1 Get Event Source

Query for an event source of this component by its name.

The returned event source may be null if no event source with the given name could be found.

Prototype

public IEventSource GetEventSource(in String8 name);

Table 3-95: Parameters of GetEventSource

Name	Dir.	Type	Description
	return	IEventSource	Event source with the given name or null if no event source with the given name could be found.
name	in	String8	Event source name.

Exceptions

None

3.5.2.5.2 Get Event Sources

Query for the collection of all event sources of the component.

The collection may be empty if no event sources exist.

Prototype

public EventSourceCollection GetEventSources();

Table 3-96: Parameters of GetEventSources

Name	Dir.	Type	Description
	return	EventSourceCollection	Collection of event sources.

Exceptions

None

3.5.2.6 IEntry Point Publisher

Interface of an entry point publisher.

An entry point publisher is a model that publishes entry points.

This is an optional interface. It needs to be implemented for managed models only, which want to provide access to their entry points by name.

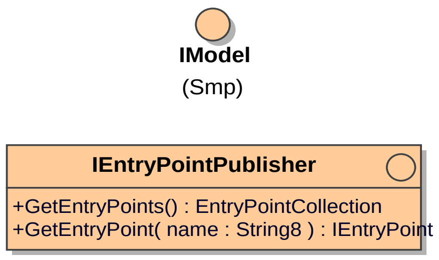

IModel
(Smp)
IEntryPointPublisher
+GetEntryPoints() : EntryPointCollection
+GetEntryPoint(name : String8) : IEntryPoint

Figure 3-59: IEntry Point Publisher

Remark: The entry points may be registered, for example, with the Scheduler or the Event Manager services.

Base Interfaces

Smp::IComponent

Table 3-97: Operations of IEntryPointPublisher

Name	Description
GetEntryPoint	Query for an entry point of this model by its name.
GetEntryPoints	Query for the collection of all entry points of the model.

3.5.2.6.1 Get Entry Point

Query for an entry point of this model by its name.

The returned entry point may be null if no entry point with the given name could be found.

Prototype

public IEntryPoint GetEntryPoint(in String8 name);

Table 3-98: Parameters of GetEntryPoint

Name	Dir.	Type	Description
	return	IEntryPoint	Entry point with given name, or null if no entry point with given name could be found.
name	in	String8	Entry point name.

Exceptions

None

3.5.2.6.2 Get Entry Points

Query for the collection of all entry points of the model.

The collection may be empty if no entry points exist.

Prototype

public EntryPointCollection GetEntryPoints();

Table 3-99: Parameters of GetEntryPoints

Name	Dir.	Type	Description
	return	EntryPointCollection	Collection of entry points.

Exceptions

None

3.5.3 Managed Model Mechanisms

The model mechanisms introduced in section 3.4 (Model Mechanisms) do not provide full access for external components, but only limited access:

- Fields are published against the simulation environment, but an external component can not query for fields by name.

To overcome these limitations, managed interfaces are provided with full access to all functionality. For fields, the managed interface allows for querying a dedicate field of a model by name. Via the returned ISimpleField and the IArrayField interface it is possible to read and write field values.

Field Names:

In SMP, models can have fields of various types, including not only the base types introduced in section 2 (Base Types), but also complex types (strings, arrays, structures, classes). The available types are documented in the Metamodel. As the methods GetValue() and SetValue() of ISimpleField only support fields of simple types, structured types have to be accessed by accessing their individual members.

Rule 1: Members of structures and classes are separated with one of the following characters: "\", "/", "!", "."

Rule 2: Members of strings and arrays with non-simple item type are addressed by their 0-based index enclosed in brackets ("[" and "]").

Examples:

MyField.Position[2]

Structure!Field

Class/Array[1]/Structure/Field

3.5.3.1 IManaged Model

Interface of a managed model.

A managed model additionally allows querying for fields by name.

This is an optional interface. It needs to be implemented for managed models only, which want to allow access to fields by name.

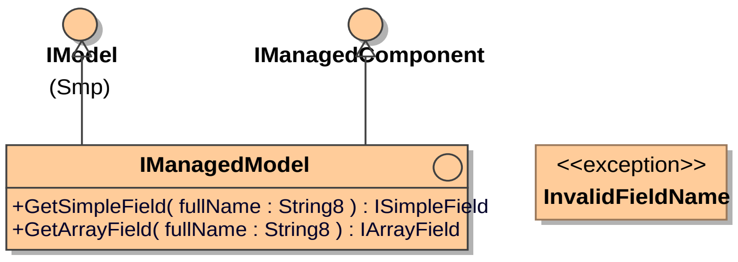

IModel
(Smp)
IManagedComponent
IManagedModel
+GetSimpleField(fullName : String8) : ISimpleField
+GetArrayField(fullName : String8) : IArrayField
<<exception>>
InvalidFieldName

Figure 3-60: IManaged Model

Base Interfaces

Smp::Management::IManagedComponent, Smp::IModel

Table 3-100: Operations of IManagedModel

Name	Description
GetArrayField	Get the field of given name that is an array of a simple type.
GetSimpleField	Get the field of given name that is typed by a simple type.

3.5.3.1.1 Get Array Field

Get the field of given name that is an array of a simple type.

This method raises an exception of type InvalidFieldName if called with a field name for which no corresponding field exists or for which the corresponding field is not an array of simple type.

This method can only be used to get array fields with items of simple type.

Prototype

public IArrayField GetArrayField(in String8 fullName) raises (InvalidFieldName);

Table 3-101: Parameters of GetArrayField

Name	Dir.	Type	Description
fullName	in	String8	Fully qualified array field name (relative to the model)
	return	IArrayField	Array field.

Exceptions

Smp::Management::IManagedModel::InvalidFieldName

3.5.3.1.2 Get Simple Field

Get the field of given name that is typed by a simple type.

This method raises an exception of type InvalidFieldName if called with a field name for which no corresponding field exists or for which the corresponding field is not of simple type.

This method can only be used to get fields of simple types.

Remark: For getting access to fields of structured or array types, this method may be called multiply, for example by specifying a field name "MyField.Position[2]" in order to access an array item of a structure.

Prototype

public ISimpleField GetSimpleField(in String8 fullName) raises (InvalidFieldName);

Table 3-102: Parameters of GetSimpleField

Name	Dir.	Type	Description
	return	ISimpleField	Simple field.
fullName	in	String8	Fully qualified field name (relative to the model).

Exceptions

Smp::Management::IManagedModel::InvalidFieldName

3.5.3.1.3 Invalid Field Name

This exception is raised when an invalid field name is specified.

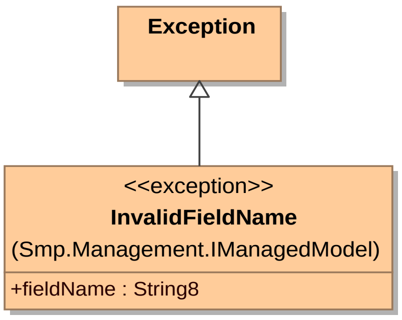


Figure 3-61: Invalid Field Name

Table 3-103: Fields of InvalidFieldName

Name	Type	Description
fieldName	String8	Fully qualified field name that is invalid.

3.5.3.2 IField

Interface of a field.

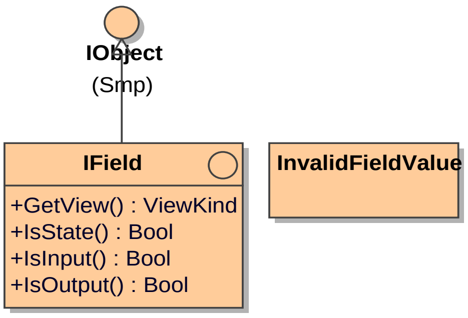


Figure 3-62: IField

Base Interfaces

Smp::IObject

Table 3-104: Operations of IField

Name	Description
GetView	Return View kind of the field.
IsInput	Return Input flag of the field.
IsOutput	Return Output flag of the field.
IsState	Return State flag of the field.

3.5.3.2.1 Get View

Return View kind of the field.

Prototype

public ViewKind GetView();

Table 3-105: Parameters of GetView

Name	Dir.	Type	Description
	return	ViewKind	The View kind of the field. The view kind indicates which user roles have visibility of the field.

Exceptions

None

3.5.3.2.2 Is Input

Return Input flag of the field.

Prototype

public Bool IsInput();

Table 3-106: Parameters of IsInput

Name	Dir.	Type	Description
	return	Bool	The Input flag of the field. When true, the field is considered an input into the model and can be used as target of a field link in data flow based design.

Exceptions

None

3.5.3.2.3 Is Output

Return Output flag of the field.

Prototype

public Bool IsOutput();

Table 3-107: Parameters of IsOutput

Name	Dir.	Type	Description
	return	Bool	The Output flag of the field. When true, the field is considered an output of the model and can be used as source of a field link in data flow based design.

Exceptions

None

3.5.3.2.4 Is State

Return State flag of the field.

Prototype

public Bool IsState();

Table 3-108: Parameters of IsState

Name	Dir.	Type	Description
	return	Bool	The State flag of the field. When true, the state of the field shall be stored by the simulation infrastructure persistence mechanism on Store(), and restored on Restore().

Exceptions

None

3.5.3.2.5 Invalid Field Value

This exception is raised when trying to assign an illegal value to a field.

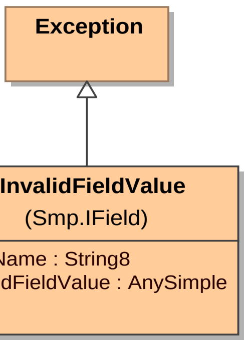


Figure 3-63: Invalid Field Value

Table 3-109: Fields of InvalidFieldValue

Name	Type	Description
fieldName	String8	Fully qualified field name the value was assigned to.
invalidFieldValue	AnySimple	Value that was passed as new field value.

3.5.3.3 IArray Field

Interface of a field which is an array of a simple type.

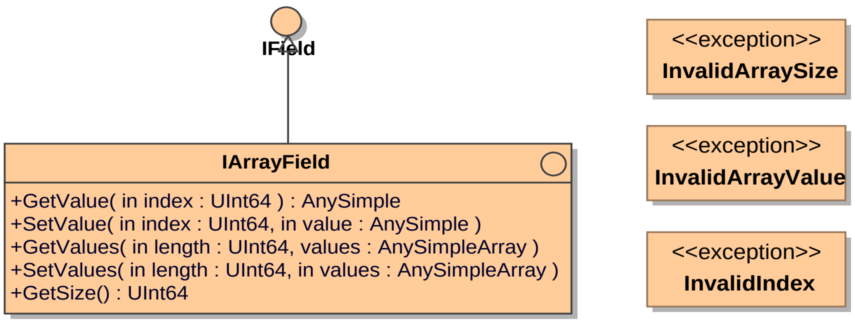

IField
IArrayField
+GetValue(in index : UInt64) : AnySimple
+SetValue(in index : UInt64, in value : AnySimple)
+GetValues(in length : UInt64, values : AnySimpleArray)
+SetValues(in length : UInt64, in values : AnySimpleArray)
+GetSize() : UInt64
<<exception>>
InvalidArraySize
<<exception>>
InvalidArrayValue
<<exception>>
InvalidIndex

Figure 3-64: IArray Field

Base Interfaces

Smp::IField

Table 3-110: Operations of IArrayFiled

Name	Description
GetSize	Get the size (number of array items) of the field.
GetValue	Get a value from a specific index of the array field.
GetValues	Get all values of the array field.
SetValue	Set a value at given index of the array field.
SetValues	Set all values of the array field.

3.5.3.3.1 Get Size

Get the size (number of array items) of the field.

Prototype

public UInt64 GetSize();

Table 3-111: Parameters of GetSize

Name	Dir.	Type	Description
	return	UInt64	Size (number of array items) of the field.

Exceptions

None

3.5.3.3.2 Get Value

Get a value from a specific index of the array field.

This method raises an exception of type InvalidIndex if called with an index value beyond the size of the array.

Prototype

public AnySimple GetValue(in UInt64 index) raises (InvalidIndex);

Table 3-112: Parameters of GetValue

Name	Dir.	Type	Description
	return	AnySimple	Value from given index.
index	in	UInt64	Index of value to get.

Exceptions

Smp::IArrayField::InvalidIndex

3.5.3.3.3 Get Values

Get all values of the array field.

This method raises an exception of type InvalidArraySize if called with an array of wrong size.

The array with the values has to be pre-allocated by the caller, and has to be released by the caller as well. Therefore, an inout parameter is used, not a return value of the method.

Prototype

public void GetValues(in UInt64 length, AnySimpleArray values) raises (InvalidArraySize);

Table 3-113: Parameters of GetValues

Name	Dir.	Type	Description
length	in	UInt64	Size of given values array.
values	inout	AnySimpleArray	Pre-allocated array of values to store result to.

Exceptions

Smp::IArrayField::InvalidArraySize

3.5.3.3.4 Set Value

Set a value at given index of the array field.

This method raises an exception of type InvalidFieldValue if called with an invalid value and an exception of type InvalidIndex if called with an index value beyond the size of the array.

Prototype

public void SetValue(in UInt64 index, in AnySimple value) raises (InvalidFieldValue, InvalidIndex);

Table 3-114: Parameters of SetValue

Name	Dir.	Type	Description
index	in	UInt64	Index of value to set.
value	in	AnySimple	Value to set at given index.

Exceptions

Smp::IField::InvalidFieldValue, Smp::IArrayField::InvalidIndex

3.5.3.3.5 Set Values

Set all values of the array field.

This method raises an exception of type InvalidArraySize if called with an array of wrong size and an exception of type InvalidArrayValue if called with an invalid values array.

Prototype

public void SetValues(in UInt64 length, in AnySimpleArray values) raises (InvalidArraySize, InvalidArrayValue);

Table 3-115: Parameters of SetValues

Name	Dir.	Type	Description
length	in	UInt64	Size of given values array.
values	in	AnySimpleArray	Array of values to store in array field.

Exceptions

Smp::IArrayField::InvalidArraySize, Smp::IArrayField::InvalidArrayValue

3.5.3.3.6 Invalid Array Size

This exception is raised when an invalid array size is specified.

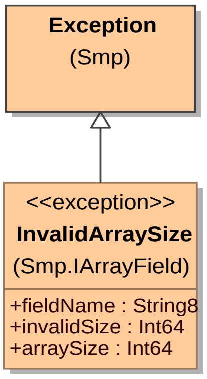


Figure 3-65: Invalid Array Size

Table 3-116: Fields of InvalidArraySize

Name	Type	Description
arraySize	Int64	Real array size.
fieldName	String8	Name of field that has been accessed.
invalidSize	Int64	Invalid array size.

3.5.3.3.7 Invalid Array Value

This exception is raised when trying to assign an illegal value to an array field.

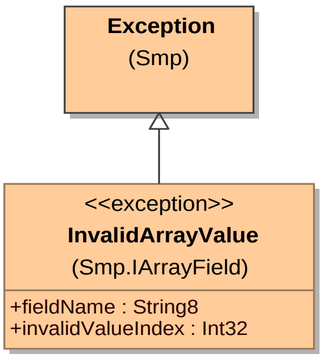


Figure 3-66: Invalid Array Value

Table 3-117: Fields of InvalidArrayValue

Name	Type	Description
fieldName	String8	Fully qualified field name the value was assigned to.
invalidValueIndex	Int32	Index in array where the first invalid value was found.

3.5.3.3.8 Invalid Index

This exception is raised when an invalid index is specified.

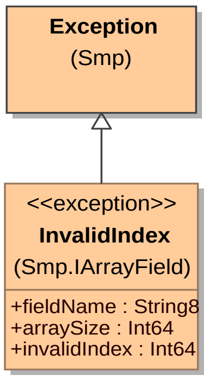


Figure 3-67: Invalid Index

Table 3-118: Fields of InvalidIndex

Name	Type	Description
arraySize	Int64	Real array size.
fieldName	String8	Fully qualified field name the value was assigned to.
invalidIndex	Int64	Invalid Index.

3.5.3.4 ISimple Field

Interface of a field of simple type.

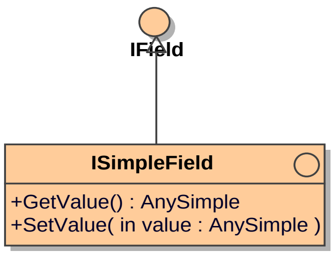


Figure 3-68: ISimple Field

Base Interfaces

Smp::IField

Table 3-119: Operations of ISimpleField

Name	Description
GetValue	Get the value of the simple field.
SetValue	Set the value of the simple field.

3.5.3.4.1 Get Value

Get the value of the simple field.

Prototype

public AnySimple GetValue();

Table 3-120: Parameters of GetValue

Name	Dir.	Type	Description
	return	AnySimple	Field value.

Exceptions

None

3.5.3.4.2 Set Value

Set the value of the simple field.

This method raises an exception of type InvalidFieldValue if called with an invalid value.

Prototype

public void SetValue(in AnySimple value) raises (InvalidFieldValue);

Table 3-121: Parameters of SetValue

Name	Dir.	Type	Description
value	in	AnySimple	Field value.

Exceptions

Smp::IField::InvalidFieldValue

3.5.3.5 IForcible Field

Interface of a forcible field.

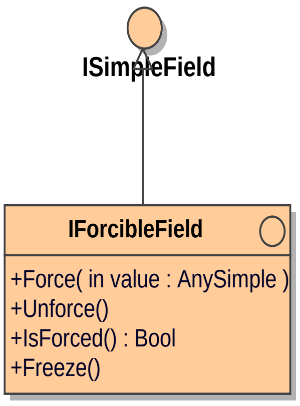


Figure 3-69: IForcible Field

Base Interfaces

Smp::ISimpleField

Table 3-122: Operations of IForcibleField

Name	Description
Force	Force field to given value.
Freeze	Force field to its current value.
IsForced	Query for the forced state of the field.
Unforce	Unforce field.

3.5.3.5.1 Force

Force field to given value.

This method raises an exception of type InvalidFieldValue if called with an invalid value.

Prototype

public void Force(in AnySimple value) raises (InvalidFieldValue);

Table 3-123: Parameters of Force

Name	Dir.	Type	Description
value	in	AnySimple	Value to force field to.

Exceptions

Smp::IField::InvalidFieldValue

3.5.3.5.2 Freeze

Force field to its current value.

Prototype

public void Freeze();

Parameters

None

Exceptions

None

3.5.3.5.3 Is Forced

Query for the forced state of the field.

Prototype

public Bool IsForced();

Table 3-124: Parameters of IsForced

Name	Dir.	Type	Description
	return	Bool	Whether the field is forced or not.

Exceptions

None

3.5.3.5.4 Unforce

Unforce field.

Prototype

public void Unforce();

Parameters

None

Exceptions

None

3.5.3.6 View Kind

This enumeration defines possible options for the View attribute, which can be used to control if and how an element is made visible when published to the simulation infrastructure.

The simulation infrastructure must at least support the "None" and the "EndUser" roles (i.e. hidden or always visible).

The simulation infrastructure may support the selection of different user roles ("Debug", "Expert", "End User"), in which case the "Debug" and the "Expert" role must also be supported as described.

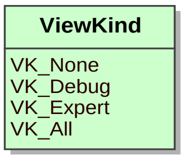


Figure 3-70: View Kind

Table 3-125: Enumeration Literals of ViewKind

Name	Description
VK_None	The element is not made visible to the user (this is the default).
VK_Debug	The element is made visible for debugging purposes. The element is not visible to end users. If the simulation infrastructure supports the selection of different user roles, then the element shall be visible to "Debug" users only.
VK_Expert	The element is made visible for expert users. The element is not visible to end users. If the simulation infrastructure supports the selection of different user roles, then the element shall be visible to "Debug" and "Expert" users.
VK_All	The element is made visible to all users.

3.6 Simulation Environments

A Simulation Environment has to implement the `ISimulator` interface to give access to the models and services. This interface is derived from `IComposite` to give access to at least two managed containers, namely the “Models” and “Services” containers. Finally, a simulation environment has to pass a publication server to all models in the `Publishing` state.

3.6.1 Simulators

The Simulation Environment is always in one of the defined simulator states, with well-defined state transition methods between these states.

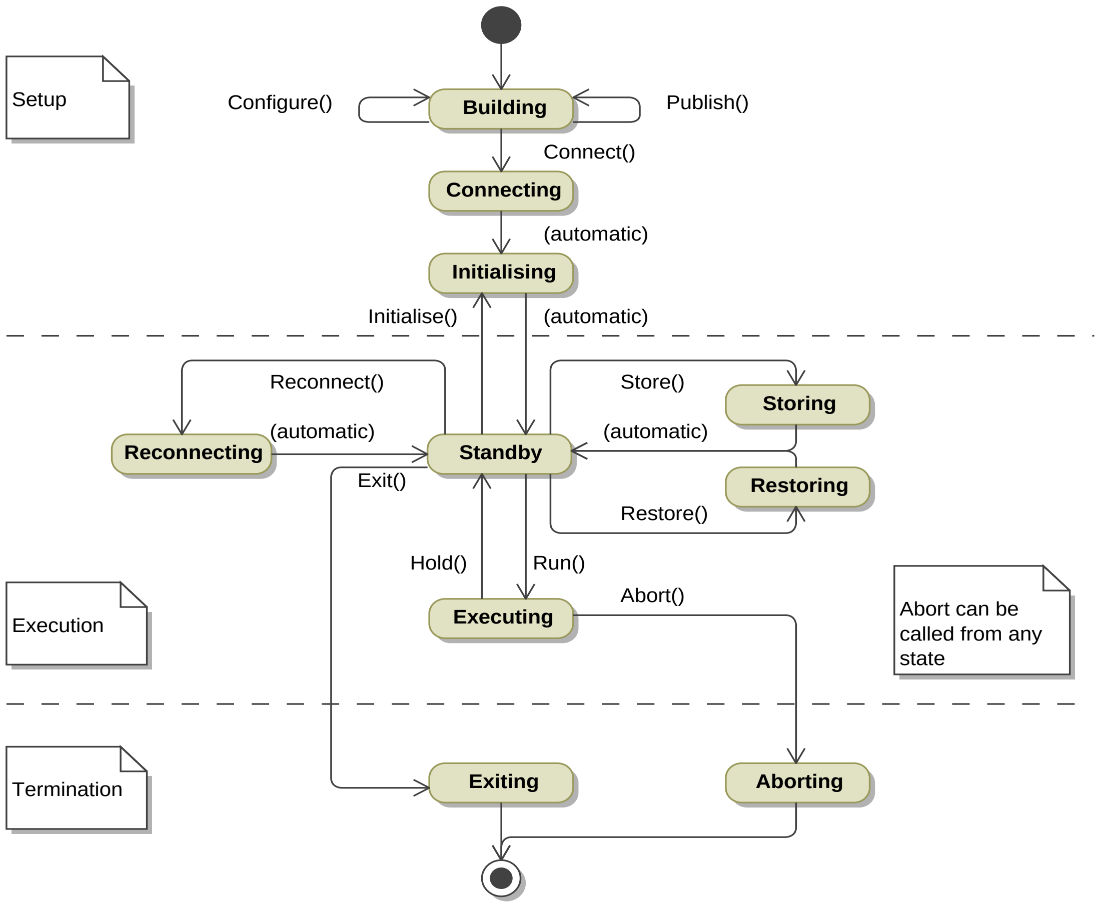


Figure 3-71: Simulation Environment State Diagram with State Transition Methods

The available simulator states are enumerated by the SimulatorStateKind enumeration, while the ISimulator interface provides the corresponding state transition methods. Except for the Abort() state transition, which can be called from any other state, all other state transitions should be called only from the appropriate states, as shown in the Simulation Environment State Diagram in Figure 3-71, and explained in the following subsections. However, when calling a state transition from another state, the simulation environment shall not raise an exception, but ignore the state transition. It may use the Logger service to log a warning message.

The simulator states correspond to dedicated Model states. Therefore a model is always in one of the defined model states too, with well-defined state transition methods between these states. The available model states are enumerated by the ModelStateKind enumeration, while the IModel interface provides the corresponding state transition methods.

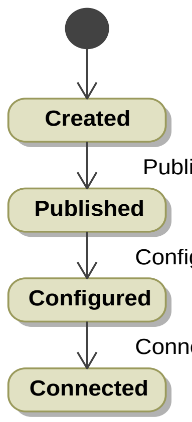


Figure 3-72: Model State Diagram with State Transition Methods

During the Building state the Simulation Environment builds the models and calls Publish() and Configure() to initiate the transition of the models from Created via Published to Configured state. During the Connecting state the Simulation Environment initiates the transition of the models into Connected state.

3.6.1.1 Simulator State Kind

This is an enumeration of the available states of the simulator. The Setup phase is split into three different states, the Execution phase has five different states, and the Termination phase has two states.

Table 3-126: Enumeration Literals of SimulatorStateKind

Name	Description
SSK_Building	In Building state, the model hierarchy is created. This is done by an external component, not by the simulator. This state is entered automatically after the simulation environment has performed its initialisation. This state is left with the Connect() state transition method.
SSK_Connecting	In Connecting state, the simulation environment traverses the model hierarchy and calls the Connect() method of each model. This state is entered using the Connect() state transition. After connecting all models to the simulator, an automatic state transition to the Initialising state is performed.
SSK_Initialising	In Initialising state, the simulation environment executes all initialisation entry points in the order they have been added to the simulator using the AddInitEntryPoint() method. This state is either entered automatically after the simulation environment has connected all models to the simulator, or manually from Standby state using the Initialise() state transition. After calling all initialisation entry points, an automatic state transition to the Standby state is performed.

Name	Description
SSK_Standby	In Standby state, the simulation environment (namely the Time Keeper Service) does not progress simulation time. Only entry points registered relative to Zulu time are executed. This state is entered automatically from the Initialising, Storing, and Restoring states, or manually from the Executing state using the Hold() state transition. This state is left with one of the Run(), Store(), Restore(), Initialise(), Reconnect() or Exit() state transitions.
SSK_Executing	In Executing state, the simulation environment (namely the Time Keeper Service) does progress simulation time. Entry points registered with any of the available time kinds are executed. This state is entered using the Run() state transition. This state is left using the Hold() state transition.
SSK_Storing	In Storing state, the simulation environment first stores the values of all fields published with the State attribute to storage (typically a file). Afterwards, the Store() method of all components (Models and Services) implementing the optional IPersist interface is called, to allow custom storing of additional information. While in this state, fields published with the State attribute must not be modified by the models, to ensure that a consistent set of field values is stored. This state is entered using the Store() state transition. After storing the simulator state, an automatic state transition to the Standby state is performed.
SSK_Restoring	In Restoring state, the simulation environment first restores the values of all fields published with the State attribute from storage. Afterwards, the Restore() method of all components implementing the optional IPersist interface is called, to allow custom restoring of additional information. While in this state, fields published with the State attribute must not be modified by the models, to ensure that a consistent set of field values is restored. This state is entered using the Restore() state transition. After restoring the simulator state, an automatic state transition to the Standby state is performed.
SSK_Reconnecting	In Reconnecting state, the simulation environment makes sure that models that have been added to the simulator after leaving the Building state are properly published, configured and connected. This state is entered using the Reconnect() state transition. After connecting all new models, an automatic state transition to the Standby state is performed.
SSK_Exiting	In Exiting state, the simulation environment is properly terminating a running simulation. This state is entered using the Exit() state transition. After exiting, the simulator is in an undefined state.

Name	Description
SSK_Aborting	In this state, the simulation environment performs an abnormal simulation shut-down. This state is entered using the Abort() state transition. After aborting, the simulator is in an undefined state.

3.6.1.2 ISimulator

This interface gives access to the simulation environment state and state transitions. Further, it provides convenience methods to add models, and to add and retrieve simulation services.

This is a mandatory interface that every SMP compliant simulation environment has to implement.

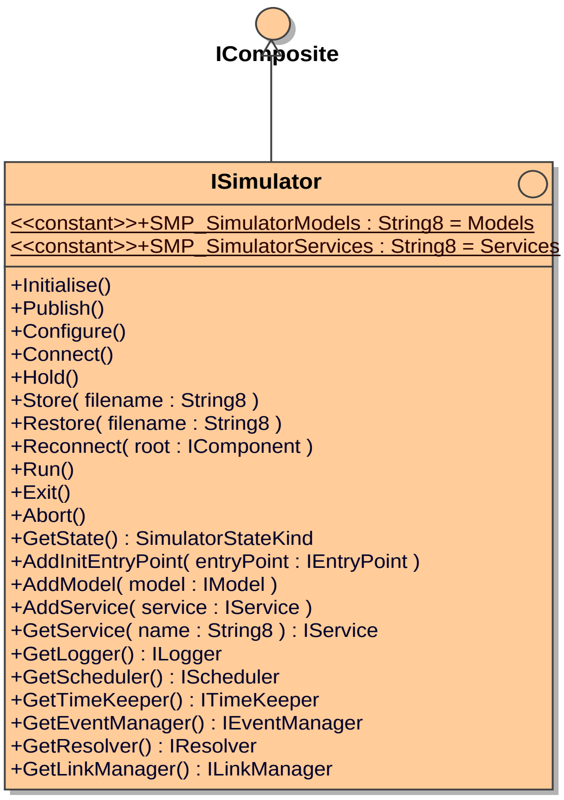

IComposite
ISimulator
<<constant>>+SMP_SimulatorModels : String8 = Models
<<constant>>+SMP_SimulatorServices : String8 = Services
+Initialise()
+Publish()
+Configure()
+Connect()
+Hold()
+Store(filename : String8)
+Restore(filename : String8)
+Reconnect(root : IComponent)
+Run()
+Exit()
+Abort()
+GetState() : SimulatorStateKind
+AddInitEntryPoint(entryPoint : IEntryPoint)
+AddModel(model : IModel)
+AddService(service : IService)
+GetService(name : String8) : IService
+GetLogger() : ILogger
+GetScheduler() : IScheduler
+GetTimeKeeper() : ITimeKeeper
+GetEventManager() : IEventManager
+GetResolver() : IResolver
+GetLinkManager() : ILinkManager

Figure 3-73: ISimulator

Base Interfaces

Smp::IComposite

Table 3-127: Constants of ISimulator

Name	Type	Description	Value
SMP_SimulatorModels	String8	Name of the model container.	Models
SMP_SimulatorServices	String8	Name of the service container.	Services

Table 3-128: Operations of ISimulator

Name	Description
Initialise	This method asks the simulation environment to call all initialisation entry points again.
Publish	This method asks the simulation environment to call the Publish() method of all model instances in the model hierarchy which are still in Created state.
Configure	This method asks the simulation environment to call the Configure() method of all model instances which are still in Publishing state.
Connect	This method informs the simulation environment that the hierarchy of model instances has been configured, and can now be connected to the simulator. Thus, the simulation environment calls the Connect() method of all model instances.
Run	This method changes from Standby to Executing state.
Hold	This method changes from Executing to Standby state.
Store	This method is used to store a state vector to file.
Restore	This method is used to restore a state vector from file.
Reconnect	This method asks the simulation environment to reconnect the component hierarchy starting at the given root component.
Exit	This method is used for a normal termination of a simulation.
Abort	This method is used for an abnormal termination of a simulation.
GetState	Return the current simulator state.
AddInitEntryPoint	This method can be used to add entry points that shall be executed in the Initialising state.
AddModel	This method adds a model to the models collection of the simulator, i.e. to the "Models" container.
AddService	This method adds a user-defined service to the services collection, i.e. to the "Services" container.
GetService	Query for a service component by its name.
GetLogger	Return interface to logger service.
GetTimeKeeper	Return interface to time keeper service.
GetScheduler	Return interface to scheduler service.
GetEventManager	Return interface to event manager service.
GetResolver	Return interface to resolver service.
GetLinkRegistry	Return interface to link registry service.

3.6.1.2.1 Initialise

This method asks the simulation environment to call all initialisation entry points again.

This method can only be called when in Standby state, and enters Initialising state. After completion, the simulator automatically returns to Standby state.

The entry points will be executed in the order they have been added to the simulator using the AddInitEntryPoint() method.

Prototype

public void Initialise();

Parameters

None

Exceptions

None

3.6.1.2.2 Publish

This method asks the simulation environment to call the Publish() method of all model instances in the model hierarchy which are still in Created state.

This method must only be called when in Building state.

Remark: This method is typically called by an external component after creating new model instances, typically from information in an SMDL Assembly.

Prototype

public void Publish();

Parameters

None

Exceptions

None

3.6.1.2.3 Configure

This method asks the simulation environment to call the Configure() method of all model instances which are still in Publishing state.

This method must only be called when in Building state.

Remark: This method is typically called by an external component after setting field values of new model instances, typically using the information in an SMDL Assembly or SMDL Configuration.

Prototype

public void Configure();

Parameters

None

Exceptions

None

3.6.1.2.4 Connect

This method informs the simulation environment that the hierarchy of model instances has been configured, and can now be connected to the simulator. Thus, the simulation environment calls the Connect() method of all model instances.

In this method, the simulation environment first enters Connecting state and calls the Connect() method of every model in the model hierarchy, then enters

Initialising state and calls the initialisation entry points, and finally enters Standby state.

This method must only be called when in Building state.

Remark: This method is typically called by an external component after configuring all model instances.

Prototype

public void Connect();

Parameters

None

Exceptions

None

3.6.1.2.5 Run

This method changes from Standby to Executing state.

This method must only be called when in Standby state, and enters Executing state.

Prototype

public void Run();

Parameters

None

Exceptions

None

3.6.1.2.6 Hold

This method changes from Executing to Standby state.

This method must only be called when in Executing state, and enters Standby state.

Prototype

public void Hold();

Parameters

None

Exceptions

None

3.6.1.2.7 Store

This method is used to store a state vector to file.

This method must only be called when in Standby state, and enters Storing state. On completion, it automatically returns to Standby state.

Prototype

public void Store(in String8 filename);

Table 3-129: Parameters of Store

Name	Dir.	Type	Description
filename	in	String8	Name to use for simulation state vector file.

Exceptions

None

3.6.1.2.8 Restore

This method is used to restore a state vector from file.

This method must only be called when in Standby state, and enters Restoring state. On completion, it automatically returns to Standby state.

Prototype

public void Restore(in String8 filename);

Table 3-130: Parameters of Restore

Name	Dir.	Type	Description
filename	in	String8	Name of simulation state vector file.

Exceptions

None

3.6.1.2.9 Reconnect

This method asks the simulation environment to reconnect the component hierarchy starting at the given root component.

This method must only be called when in Standby state.

The simulation environment has to ensure that all models under the given root (but not the root itself) are published, configured and connected, so that all child models are finally in Connected state.

Remark: This method is typically called after creating additional model instances and adding them to the existing model hierarchy.

Prototype

public void Reconnect(in IComponent root);

Table 3-131: Parameters of Reconnect

Name	Dir.	Type	Description
root	in	IComponent	Root component to start reconnecting from. This can be the parent component of a newly added model, or e.g. the simulator itself.

Exceptions

None

3.6.1.2.10 Exit

This method is used for a normal termination of a simulation.

This method must only be called when in Standby state, and enters Exiting state.

Prototype

public void Exit();

Parameters

None

Exceptions

None

3.6.1.2.11 Abort

This method is used for an abnormal termination of a simulation.

This method can be called from any other state, and enters Aborting state.

Prototype

public void Abort();

Parameters

None

Exceptions

None

3.6.1.2.12 Get State

Return the current simulator state.

Prototype

public SimulatorStateKind GetState();

Table 3-132: Parameters of GetState

Name	Dir.	Type	Description
	return	SimulatorStateKind	Current simulator state.

Exceptions

None

3.6.1.2.13 Add Init Entry Point

This method can be used to add entry points that shall be executed in the Initialising state.

The entry points will be executed in the order they have been added to the simulator.

Remark: The ITask interface (which is derived from IEntryPoint) can be used to add several entry points in a well-defined order.

Prototype

public void AddInitEntryPoint(in IEntryPoint entryPoint);

Table 3-133: Parameters of AddInitEntryPoint

Name	Dir.	Type	Description
entryPoint	in	IEntryPoint	Entry point to execute in Initialising state.

Exceptions

None

3.6.1.2.14 Add Model

This method adds a model to the models collection of the simulator, i.e. to the "Models" container.

This method raises an exception of type DuplicateName if the name of the new model conflicts with the name of an existing component (model or service).

The container for the models has no upper limit and thus the ContainerFull exception will never be thrown.

The method will never throw the InvalidObjectType exception either, as it gets a component implementing the IModel interface.

Prototype

public void AddModel(in IModel model) raises (DuplicateName);

Table 3-134: Parameters of AddModel

Name	Dir.	Type	Description
model	in	IModel	New model to add to collection of root models, i.e. to the "Models" container.

Exceptions

Smp::DuplicateName

3.6.1.2.15 Add Service

This method adds a user-defined service to the services collection, i.e. to the "Services" container.

This method raises an exception of type DuplicateName if the name of the new service conflicts with the name of an existing component (model or service).

The container for the services has no upper limit and thus the ContainerFull exception will never be thrown.

The method will never throw the InvalidObjectType exception either, as it gets a component implementing the IService interface.

The container for the services has no upper limit and thus the ContainerFull exception will never be thrown.

The method will never throw the InvalidObjectType exception, as it expects a component implementing the IService interface.

Remark: It is recommended that custom services include a project or company acronym as prefix in their name, to avoid collision of service names.

Prototype

public void AddService(in IService service) raises (DuplicateName);

Table 3-135: Parameters of AddService

Name	Dir.	Type	Description
service	in	IService	Service to add to services collection.

Exceptions

Smp::DuplicateName

3.6.1.2.16 Get Service

Query for a service component by its name.

The returned component is null if no service with the given name could be found. Standard names are defined for the standardised services, while custom services use custom names.

The existence of custom services is not guaranteed, so models should expect to get a null reference.

Prototype

public IService GetService(in String8 name);

Table 3-136: Parameters of GetService

Name	Dir.	Type	Description
	return	IService	Service with the given name, or null if no service with the given name could be found.
name	in	String8	Service name.

Exceptions

None

3.6.1.2.17 Get Logger

Return interface to logger service.

Remark: This is a type-safe convenience method, to avoid having to use the generic GetService() method. For the standardised services, it is recommended to use the convenience methods, which are guaranteed to return a valid reference.

Prototype

public ILogger GetLogger();

Table 3-137: Parameters of ILogger

Name	Dir.	Type	Description
	return	ILogger	Interface to mandatory logger service.

Exceptions

None

3.6.1.2.18 Get Time Keeper

Return interface to time keeper service.

Remark: This is a type-safe convenience method, to avoid having to use the generic GetService() method. For the standardised services, it is recommended to use the convenience methods, which are guaranteed to return a valid reference.

Prototype

public ITimeKeeper GetTimeKeeper();

Table 3-138: Parameters of GetTime

Name	Dir.	Type	Description
	return	ITimeKeeper	Interface to mandatory time keeper service.

Exceptions

None

3.6.1.2.19 Get Scheduler

Return interface to scheduler service.

Remark: This is a type-safe convenience method, to avoid having to use the generic GetService() method. For the standardised services, it is recommended to use the convenience methods, which are guaranteed to return a valid reference.

Prototype

public IScheduler GetScheduler();

Table 3-139: Parameters of GetScheduler

Name	Dir.	Type	Description
	return	IScheduler	Interface to mandatory scheduler service.

Exceptions

None

3.6.1.2.20 Get Event Manager

Return interface to event manager service.

Remark: This is a type-safe convenience method, to avoid having to use the generic GetService() method. For the standardised services, it is recommended to use the convenience methods, which are guaranteed to return a valid reference.

Prototype

public IEventManager GetEventManager();

Table 3-140: Parameters of GetEventManager

Name	Dir.	Type	Description
	return	IEventManager	Interface to mandatory event manager service.

Exceptions

None

3.6.1.2.21 Get Resolver

Return interface to resolver service.

Remark: This is a type-safe convenience method, to avoid having to use the generic GetService() method. For the standardised services, it is recommended to use the convenience methods, which are guaranteed to return a valid reference.

Prototype

public IResolver GetResolver();

Table 3-141: Parameters of GetResolver

Name	Dir.	Type	Description
	return	IResolver	Interface to mandatory resolver service.

Exceptions

None

3.6.1.2.22 Get Link Registry

Return interface to link registry service.

Remark: This is a type-safe convenience method, to avoid having to use the generic GetService() method. For the standardised services, it is recommended to use the convenience methods, which are guaranteed to return a valid reference.

Prototype

public ILinkRegistry GetLinkRegistry();

Table 3-142: Parameters of GetLinkRegistry

Name	Dir.	Type	Description
	return	ILinkRegistry	Interface to mandatory link registry service.

Exceptions

None

3.6.1.3 IManaged Simulator

This interface gives access to a managed simulator.

This interface extends the ISimulator interface and adds methods to create components (typically models) from component factories. It makes use of the IFactory interface for component factories.

This is an optional interface the simulation environment may implement.

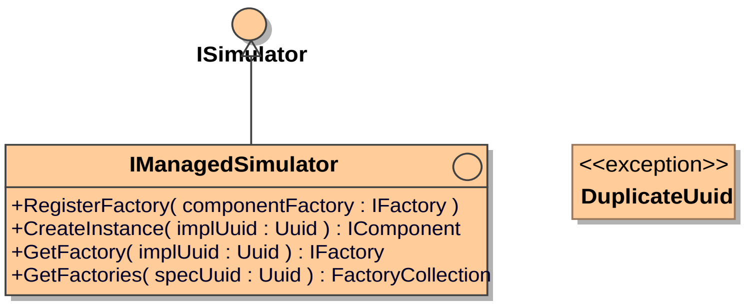

ISimulator
IManagedSimulator
+RegisterFactory(componentFactory : IFactory)
+CreateInstance(implUuid : Uuid) : IComponent
+GetFactory(implUuid : Uuid) : IFactory
+GetFactories(specUuid : Uuid) : FactoryCollection
<<exception>>
DuplicateUuid

Figure 3-74: IManaged Simulator

Base Interfaces

Smp::ISimulator

Table 3-143: Operations of IManagedSimulator

Name	Description
CreateInstance	This method creates an instance of the component with the given implementation identifier.
GetFactories	This method returns all factories of components with the given specification identifier.
GetFactory	This method returns the factory of the component with the given implementation identifier.
RegisterFactory	This method registers a component factory with the managed simulator. The managed simulator can use this factory to create component instances of the component implementation in its CreateInstance() method.

3.6.1.3.1 Create Instance

This method creates an instance of the component with the given implementation identifier.

Remark: This method is typically called during Creating state when building the hierarchy of models.

Prototype

public IComponent CreateInstance(in Uuid implUuid);

Table 3-144: Parameters of CreateInstance

Name	Dir.	Type	Description
	return	IComponent	New instance of the component with the given implementation identifier or null in case no factory for the given implementation identifier has been registered.
implUuid	in	Uuid	Implementation identifier of the component.

Exceptions

None

3.6.1.3.2 Get Factories

This method returns all factories of components with the given specification identifier.

The returned collection may be empty if no factories have been registered for the given specification identifier.

Prototype

public FactoryCollection GetFactories(in Uuid specUuid);

Table 3-145Parameters of GetFactories

Name	Dir.	Type	Description
	return	FactoryCollection	Collection of factories for the given specification identifier.
specUuid	in	Uuid	Specification identifier of the component.

Exceptions

None

3.6.1.3.3 Get Factory

This method returns the factory of the component with the given implementation identifier.

Prototype

public IFactory GetFactory(in Uuid implUuid);

Table 3-146: Parameters of GetFactory

Name	Dir.	Type	Description
	return	IFactory	Factory of the component with the given implementation identifier or null in case no factory for the given implementation identifier has been registered.
implUuid	in	Uuid	Implementation identifier of the component.

Exceptions

None

3.6.1.3.4 Register Factory

This method registers a component factory with the managed simulator. The managed simulator can use this factory to create component instances of the component implementation in its CreateInstance() method.

This method raises an exception of type DuplicateUuid if another factory has been registered using the same implementation identifier already.

Remark: This method is typically called early in the Building state to register the available component factories before the hierarchy of model instances is created.

Prototype

public void RegisterFactory(in IFactory componentFactory) raises (DuplicateUuid);

Table 3-147: Parameters of RegisterFactory

Name	Dir.	Type	Description
componentFactory	in	IFactory	Factory to create instance of the component implementation.

Exceptions

Smp::Management::IManagedSimulator::DuplicateUuid

3.6.1.3.5 Duplicate Uuid

This exception is raised when trying to register a factory under a Uuid that has already been used to register another (or the same) factory.This would lead to duplicate implementation Uuids.

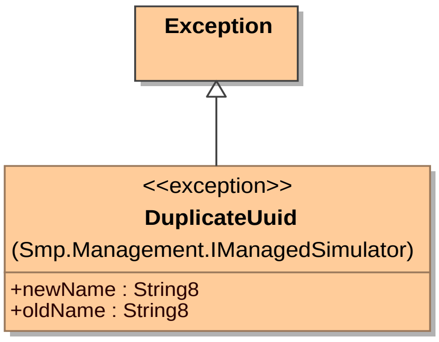


Figure 3-75: Duplicate Uuid

Table 3-148: Fields of DuplicateUuid

Name	Type	Description
newName	String8	Name of factory that tried to register under this Uuid.
oldName	String8	Name of factory already registered under this Uuid.

3.6.1.4 IFactory

Interface for a component factory.

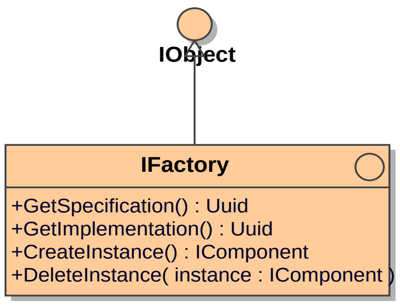

IObject
IFactory
+GetSpecification() : Uuid
+GetImplementation() : Uuid
+CreateInstance() : IComponent
+DeleteInstance(instance : IComponent)

Figure 3-76: IFactory

Base Interfaces

Smp::IObject

Table 3-149: Operations of IFactory

Name	Description
CreateInstance	Create a new instance.
DeleteInstance	Delete an existing instance.
GetImplementation	Get implementation identifier of factory.
GetSpecification	Get specification identifier of factory.

3.6.1.4.1 Create Instance

Create a new instance.

Prototype

public IComponent CreateInstance();

Table 3-150: Parameters of CreateInstance

Name	Dir.	Type	Description
	return	IComponent	New component instance.

Exceptions

None

3.6.1.4.2 Delete Instance

Delete an existing instance.

Prototype

public void DeleteInstance(in IComponent instance);

Table 3-151: Parameters of DeleteInstance

Name	Dir.	Type	Description
instance	in	IComponent	Instance to delete.

Exceptions

None

3.6.1.4.3 Get Implementation

Get implementation identifier of factory.

Prototype

public Uuid GetImplementation();

Table 3-152: Parameters of GetImplementation

Name	Dir.	Type	Description
	return	Uuid	Universally unique identifier of component implementation.

Exceptions

None

3.6.1.4.4 Get Specification

Get specification identifier of factory.

Prototype

public Uuid GetSpecification();

Table 3-153: Parameters of GetSpecification

Name	Dir.	Type	Description
	return	Uuid	Universally unique identifier of component specification.

Exceptions

None

3.6.1.5 Factory Collection

A factory collection is an ordered collection of factories, which allows iterating all members.

This type is platform specific. For details see the SMP Platform Mappings.

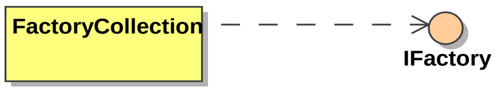


Figure 3-77: Factory Collection

3.6.2 Publication

As part of the initialisation, every model needs to be given access to a publication receiver to publish its fields and operations. While the simulation environment does not have to implement the IPublication interface itself, it has to provide a publication receiver to each model during the Publishing state.

3.6.2.1 IPublication

Interface that provides functionality to allow publishing members, including fields, properties and operations.

This interface is platform specific. For details see the SMP Platform Mappings.

3.6.3 Service Acquisition

Simulation services are closely related to models: Models need a mechanism to acquire simulation services, and most services interact with models. Further, simulation services are themselves components of the SMP Component Model. Therefore, this document puts simulation services into context with models and the simulation environment.

When the simulation environment connects a model, it calls the Connect() method of the IModel interface, passing it a global reference of ISimulator. A model can either immediately use this reference to query services, or store the reference to query services on demand.

Before a Model can call an operation of a Service, the following steps are needed:

1. The Simulator calls the Connect() operation of the Model, passing it a reference to itself.
2. The Model calls the GetService() operation of the Simulator, passing it the name of the required service.
3. The Simulator returns the required Service to the Model.
4. The Model calls the desired operation of the Service.
5. The Service returns the desired value to the Model.
6. The Model returns control to the Simulator.

If the model keeps a reference to the service, it can call the service again at any other time.

7. The Model calls the desired operation of the Service.
8. The Service returns the desired value to the Model.

These steps are shown in a sequence diagram in Figure 3-78.

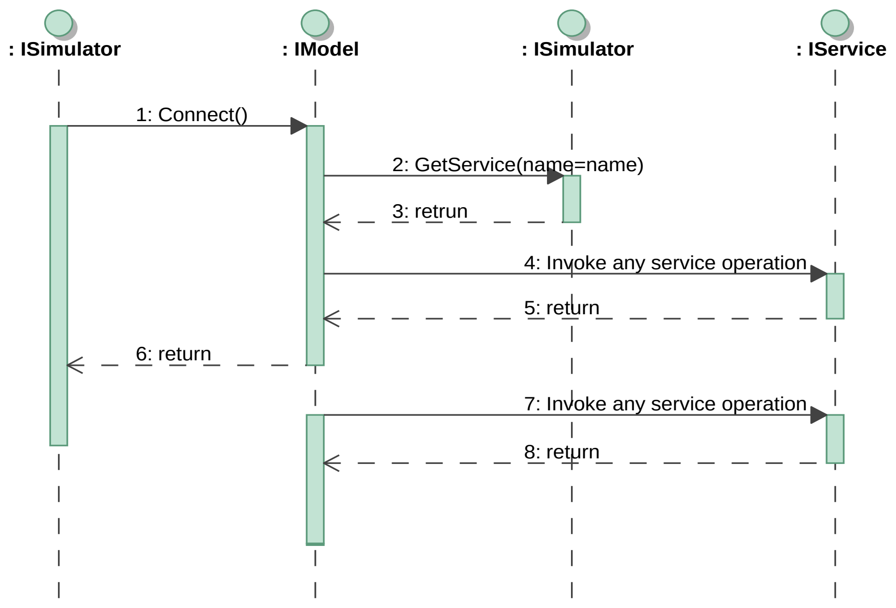


Figure 3-78: Sequence of calls for service acquisition

4 Simulation Services

In order to facilitate the inter-operability between SMP compliant simulation environments (i.e. run-time simulation kernels), several *Simulation Services* are defined in the SMP specification. All services are mandatory.

Any SMP compliant simulation environment *shall* support the following standard services.

4.1 Logger

The Logger service provides a method to send a log message to the simulation log.

4.1.1 ILogger

This interface gives access to the Logger Service.

All objects in a simulation can log messages using this service. Objects can either use pre-defined log message kinds, or define their own message kinds.

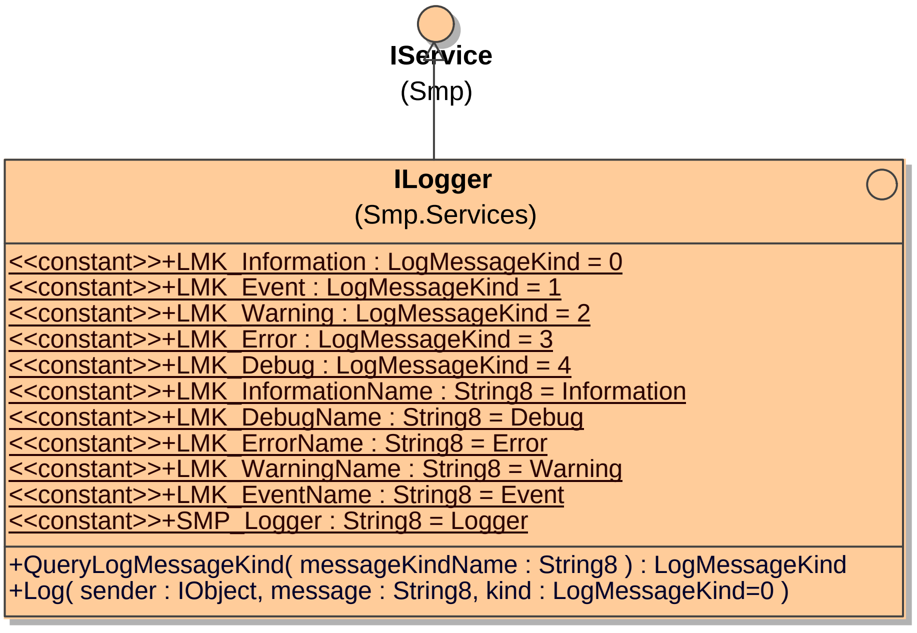

IService
(Smp)
ILogger
(Smp.Services)
<<constant>>+LMK_Information : LogMessageKind = 0
<<constant>>+LMK_Event : LogMessageKind = 1
<<constant>>+LMK_Warning : LogMessageKind = 2
<<constant>>+LMK_Error : LogMessageKind = 3
<<constant>>+LMK_Debug : LogMessageKind = 4
<<constant>>+LMK_InformationName : String8 = Information
<<constant>>+LMK_DebugName : String8 = Debug
<<constant>>+LMK_ErrorName : String8 = Error
<<constant>>+LMK_WarningName : String8 = Warning
<<constant>>+LMK_EventName : String8 = Event
<<constant>>+SMP_Logger : String8 = Logger
+QueryLogMessageKind(messageKindName : String8) : LogMessageKind
+Log(sender : IObject, message : String8, kind : LogMessageKind=0)

Figure 4-1: ILogger

Base Interfaces

Smp::IService

Table 4-1: Constants of ILogger

Name	Type	Description	Value
LMK_Debug	LogMessageKind	The message contains debug information.	4
LMK_DebugName	String8	The message contains debug information.	Debug
LMK_Error	LogMessageKind	The message has been raised because of an error.	3
LMK_ErrorName	String8	The message has been raised because of an error.	Error
LMK_Event	LogMessageKind	The message has been sent from an event, typically from a state transition.	1
LMK_EventName	String8	The message has been sent from an event, typically from a state transition.	Event
LMK_Information	LogMessageKind	The message contains general information.	0
LMK_InformationName	String8	The message contains general information.	Information
LMK_Warning	LogMessageKind	The message contains a warning.	2
LMK_WarningName	String8	The message contains a warning.	Warning
SMP_Logger	String8	Name of the logger service.	Logger

Table 4-2: Operations of ILogger

Name	Description
Log	This function logs a message to the simulation log.
QueryLogMessageKind	Return identifier of log message kind by name.

4.1.1.1 Log

This function logs a message to the simulation log.

Prototype

public void Log(in IObject sender, in String8 message, in LogMessageKind kind);

Table 4-3: Parameters of Log

Name	Dir.	Type	Description
sender	in	IObject	Object that sends the message.
message	in	String8	The message to log.
kind	in	LogMessageKind	Kind of message.

Exceptions

None

4.1.1.2 Query Log Message Kind

Return identifier of log message kind by name.

It is guaranteed that this method always returns the same id for the same messageKindName string.

Remark: This method can be used for predefined log message kinds, but is especially useful for user-defined log message kinds.

Prototype

public LogMessageKind QueryLogMessageKind(in String8 messageKindName);

Table 4-4: Parameters of QueryLogMessageKind

Name	Dir.	Type	Description
	return	LogMessageKind	Identifier of log message kind.
messageKindName	in	String8	Name of log message kind.

Exceptions

None

4.1.2 Log Message Kind

This type is used as identifier of a log message kind.

Table 4-5: Specification of LogMessageKind

Minimum	Maximum	Primitive Type
0	2147483647	Int32

4.1.3 Predefined Log Message Kinds

When logging a message with the logger service, an additional kind parameter is passed to the Log() method to identify the kind of message. The application can use any valid number, for example to allow filtering messages by message kind. However, the document pre-defines a few message kinds that are assumed to be used in most simulations.

Table 4-6 : Predefined Log Message Kinds

Message Kind	Id	Description
Information	0	The message contains general information.
Event	1	The message has been sent from an event, typically from a state transition.
Warning	2	The message contains a warning.
Error	3	The message has been raised because of an error.
Debug	4	The message contains debug information.

4.1.4 User defined Log Message Kinds

With the QueryLogMessageKind() method, it is possible to add a user-defined log message kind to the logger service. The first call of this method with a user-defined message kind name returns a new, unique identifier that can be used as third parameter for the Log() method. Further calls of the method QueryLogMessageKind() with the same user-defined message kind name are guaranteed to return the same identifier again.

Remark: This mechanism allows using a user-defined log message kind within several different models, without the need to store the log message kind identifier into a global variable. Further, it assigns a user-readable name to each log message kind, so that e.g. a log message viewer can show log message kinds by name rather than by identifier.

4.2 Time Keeper

SMP supports four different kinds of time. The time managed by the Time Keeper simulation service is called Simulation Time. The service keeps track of simulation time and puts it into relation with epoch time and mission time. Further, the service provides Zulu time based on the clock of the computer.

4.2.1 ITime Keeper

This interface gives access to the Time Keeper Service.

Components can query for the time (using the four available time kinds), and can change the epoch or mission time.

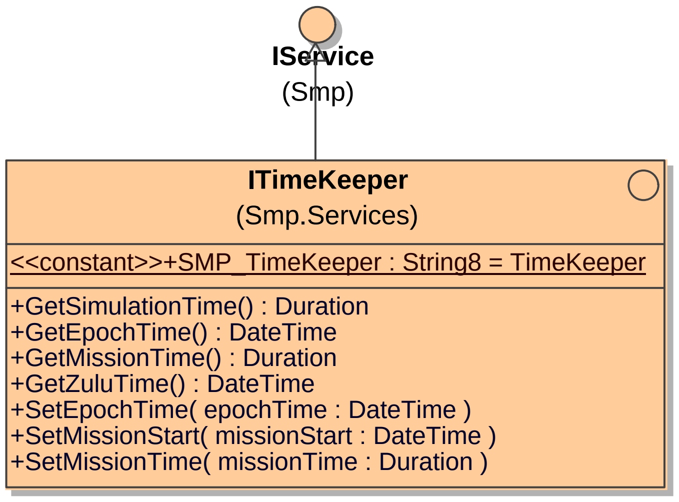

IService
(Smp)
ITimeKeeper
(Smp.Services)
<<constant>>+SMP_TimeKeeper : String8 = TimeKeeper
+GetSimulationTime() : Duration
+GetEpochTime() : DateTime
+GetMissionTime() : Duration
+GetZuluTime() : DateTime
+SetEpochTime(epochTime : DateTime)
+SetMissionStart(missionStart : DateTime)
+SetMissionTime(missionTime : Duration)

Figure 4-2: ITime Keeper

Base Interfaces

Smp::IService

Table 4-7: Constants of ITimeKeeper

Name	Type	Description	Value
SMP_TimeKeeper	String8	Name of the TimeKeeper service.	TimeKeeper

Table 4-8: Operations of ITimeKeeper

Name	Description
GetEpochTime	Return Epoch time.
GetMissionTime	Return Mission time.
GetSimulationTime	Return Simulation time.
GetZuluTime	Return Zulu time.
SetEpochTime	Set Epoch time.
SetMissionStart	Set Mission time by defining the mission start time.
SetMissionTime	Set Mission time.

4.2.1.1 Get Epoch Time

Return Epoch time.

Epoch time is an absolute time with a fixed offset to simulation time. Epoch time typically progresses with simulation time, but can be changed with SetEpochTime.

Prototype

public DateTime GetEpochTime();

Table 4-9: Parameters of GetEpochTime

Name	Dir.	Type	Description
	return	DateTime	Current epoch time.

Exceptions

None

4.2.1.2 Get Mission Time

Return Mission time.

Mission time is a relative time with a fixed offset to simulation time. Mission time typically progresses with simulation time, but can be changed with the two methods SetMissionTime and SetMissionStart. Further, mission time is updated when changing epoch time with SetEpochTime.

Prototype

public Duration GetMissionTime();

Table 4-10: Parameters of GetMissionTime

Name	Dir.	Type	Description
	return	Duration	Current mission time.

Exceptions

None

4.2.1.3 Get Simulation Time

Return Simulation time.

Simulation time is a relative time that starts at 0.

Prototype

public Duration GetSimulationTime();

Table 4-11: Parameters of GetSimulationTime

Name	Dir.	Type	Description
	return	Duration	Current simulation time.

Exceptions

None

4.2.1.4 Get Zulu Time

Return Zulu time.

Zulu time is a system dependent time and not related to simulation time. Zulu time is typically related to the system clock of the computer.

Prototype

public DateTime GetZuluTime();

Table 4-12: Parameters of GetZuluTime

Name	Dir.	Type	Description
	return	DateTime	Current Zulu time.

Exceptions

None

4.2.1.5 Set Epoch Time

Set Epoch time.

Changes the offset between simulation time and epoch time.

Calling this method shall raise a global EpochTimeChanged event in the Event Manager.

Prototype

public void SetEpochTime(in DateTime epochTime);

Table 4-13: Parameters of SetEpochTime

Name	Dir.	Type	Description
epochTime	in	DateTime	New epoch time.

Exceptions

None

4.2.1.6 Set Mission Start

Set Mission time by defining the mission start time.

Changes the offset between simulation time and mission time. The mission time itself will be calculated as the offset between the current epoch time and the given mission start:

MissionTime = EpochTime: MissionStart

Calling this method shall raise a global MissionTimeChanged event in the Event Manager.

Prototype

public void SetMissionStart(in DateTime missionStart);

Table 4-14: Parameters of SetMissionStart

Name	Dir.	Type	Description
missionStart	in	DateTime	New mission start date and time.

Exceptions

None

4.2.1.7 Set Mission Time

Set Mission time.

Changes the offset between simulation time and mission time.

Calling this method shall raise a global MissionTimeChanged event in the Event Manager.

Prototype

public void SetMissionTime(in Duration missionTime);

Table 4-15: Parameters of SetMissionTime

Name	Dir.	Type	Description
missionTime	in	Duration	New mission time.

Exceptions

None

4.2.2 Time Kind

Enumeration of supported time kinds.

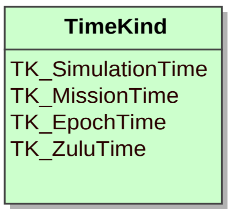


Figure 4-3: Time Kind

Table 4-16: Enumeration Literals of TimeKind

Name	Description
tTK_SimulationTime	Simulation time. Simulation time is a relative time. It does only exist within the time keeper service. The following holds for simulation time: • Simulation time is a non-negative value measured in nanoseconds, which is the lowest level of granularity supported for time in SMP. • Simulation time is stored in a signed 64-bit integer value. This allows specifying time values of more than 290 years. • Simulation time can be queried using the GetSimulationTime() method of the time keeper (via the ITimeKeeper interface). • Simulation time is initialised to 0 at the beginning of the initialisation phase. That is, during initialisation the time keeper service will return a simulation time of 0. • Simulation time is only progressed when the simulation environment is in Executing state. • When storing a state vector, simulation time is stored as well. • When restoring a state vector, simulation time is restored as well. The document does not define how quickly simulation time is progressed when the simulator is in Executing state. Typical examples are: • **Real-Time**: The simulation time progresses with real-time, where real-time is typically defined by the computer clock. Note that two types of real-time simulations exist: hard real-time and soft real-time simulations. In a hard real-time simulation, strict requirements on timing have to be met, while in a soft real-time simulation, the requirements are less demanding such that latencies in a certain range are allowed, which is called real-time slip. • **Accelerated**: The simulation time progresses relative to real-time using a constant acceleration factor. This factor may be larger than 1.0, which relates to "faster than real-time", smaller than 1.0, which means "slower than real-time", or 1.0, which coincides with real-time. • **Free Running**: The simulation time progresses as fast as possible, and is not related to real-time. Typically,

Name	Description
	the speed is coordinated with the timed events of the scheduler, which underlines the close relationship between these two services (Time Keeper and Scheduler). • **Debugging**: The simulation is executed in a step-by-step manner using break points in order to inspect data or trace calls within the simulation. SMP does not mandate which of these modes a simulation environment has to support.
TK_MissionTime	Mission time. Mission time is a relative time, i.e. it measures elapsed time from a definite point in time (called the mission start). Mission time is stored as a number relative to the mission start date. Mission time is maintained using a fixed offset to epoch time, and hence progresses together with simulation and epoch time, except for the case when the offset is changed. The following holds for mission time: • Mission time is a value measured in nanoseconds, which is the lowest level of granularity supported for time in SMP. • Mission time is returned as a signed 64-bit integer value, relative to a mission start date and time (which itself is not stored). • Mission time can be queried using the GetMissionTime() method of the time keeper (via the ITimeKeeperinterface). • Mission time is initialised to 0 at the beginning of the initialisation phase, but can be changed already before entering the execution phase. As epoch time is initialised to 01.01.2000, 12:00, the default mission start is as well 01.01.2000, 12:00. • Mission time is progressed linearly with epoch and simulation time (i.e. with a fixed offset to epoch time). Using either the SetMissionTime() method or the SetMissionStart() method of the time keeper (via the ITimeKeeper interface), the offset between simulation time and mission time can be changed. • When storing a state vector, mission time is stored as well. • When restoring a state vector, mission time is restored as well.

Name	Description
TK_EpochTime	Epoch time. Epoch time is an absolute time, i.e. it defines a definite point in time. It is not only used as a way to express date and time, but as well to determine all time-dependent variables at that time, such as barycentric positions of all solar system bodies. Epoch time is stored as a number relative to a reference date, which has been defined as the 1st of January 2000 mid-day (01.01.2000, 12:00). Epoch time is maintained using a fixed offset to simulation time, and hence progresses together with simulation time, except for the case when the offset is changed. The following holds for epoch time: • Epoch time is a value measured in nanoseconds, which is the lowest level of granularity supported for time in SMP. • Epoch time is returned as a signed 64-bit integer value, relative to the epoch reference time (01.01.2000, 12:00, Modified Julian Date 2000+0.5). This allows specifying time values roughly between 1710 and 2290. • Epoch time can be queried using the GetEpochTime() method of the time keeper (via the ITimeKeeperinterface). • Epoch time is initialised to 0 (i.e. 01.01.2000, 12:00) at the beginning of the initialisation phase, but can be changed already before entering the execution phase. • Epoch time is progressed linearly with simulation time (i.e. with a fixed offset to simulation time). Using the SetEpochTime() method of the time keeper (via the ITimeKeeper interface), the offset between simulation time and epoch time can be changed. • When storing a state vector, epoch time (i.e. its offset to simulation time) is stored as well. • When restoring a state vector, epoch time (i.e. its offset to simulation time) is restored as well.

Name	Description
TK_ZuluTime	Zulu time. From the Mobile Aeronautics Education Laboratory (MAEL) of the NASA, the following definition of Zulu Time is cited (http://www.grc.nasa.gov/WWW/MAEL/ag/zulu.htm): "The world is divided into 24 time zones. For easy reference in communications, a letter of the alphabet has been assigned to each time zone. The "clock" at Greenwich, England is used as the standard clock for international reference of time in communications, military, maritime and other activities that cross time zones. The letter designator for this clock is Z. Times are usually written in military time or 24 hour format such as 1830Z (6:30 pm). To pronounce this, the phonetic alphabet is used for the letter Z, or Zulu. This time is sometimes referred to as Zulu Time because of its assigned letter. Its official name is Coordinated Universal Time or UTC. Previously it had been known as Greenwich Mean Time or GMT but this has been replaced with UTC." In SMP, Zulu time is not related to simulation time, but typically to the computer clock (or to some external clock). The following holds for Zulu time: • Zulu time is measured in nanoseconds. • Zulu time is returned as a signed 64-bit integer value, relative to the epoch reference time. • Zulu time represents the current time at Greenwich, England, called as well UTC or GMT. As Zulu time is not managed by the time keeper service, but provided based on an external clock (typically the computer clock), it is not related to simulation time, and progresses independently of the state of the simulation environment. When a simulator interfaces to an external system, for example a ground station or some Hardware-In-The-Loop (HITL), Zulu time is often used as a time stamp.

4.3 Scheduler

The scheduler service calls entry points of models based on events triggered by one of the four time kinds.

4.3.1 IScheduler

This interface gives access to the Scheduler Service.

Components can register (Add) and unregister (Remove) entry points for scheduling. Further, they can set (Set) individual attributes of events on the scheduler.

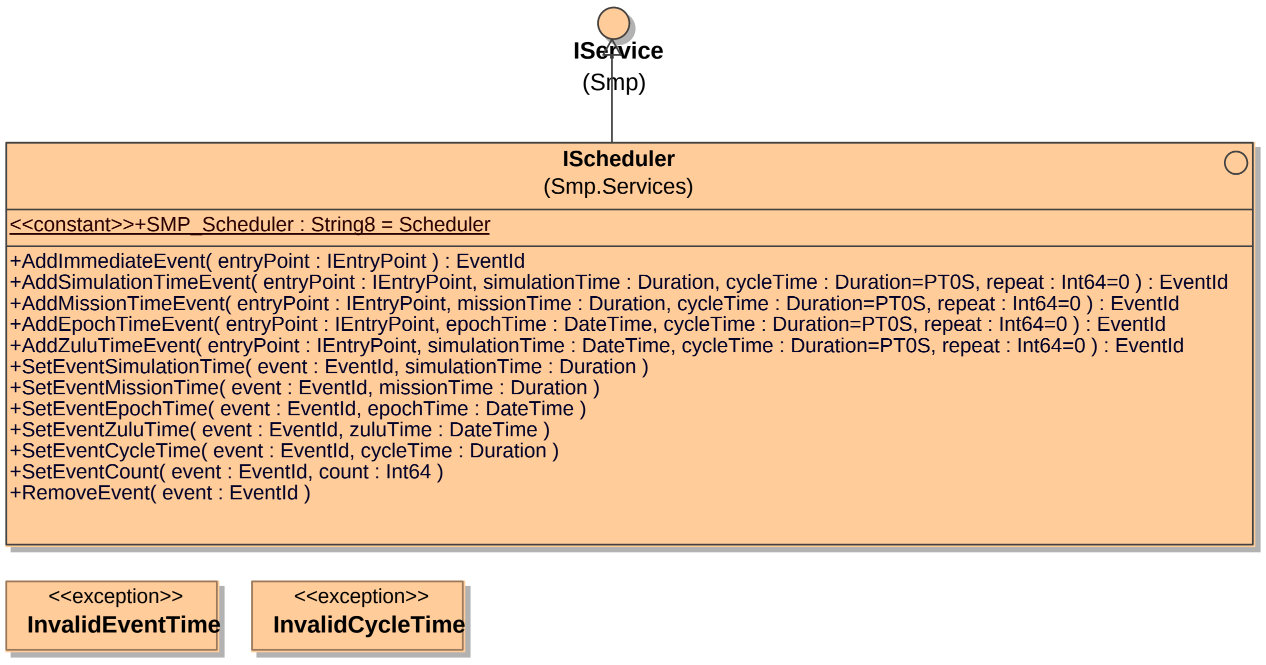

IService
(Smp)
IScheduler
(Smp.Services)
<<constant>>+SMP_Scheduler : String8 = Scheduler
+AddImmediateEvent(entryPoint : IEntryPoint) : EventId
+AddSimulationTimeEvent(entryPoint : IEntryPoint, simulationTime : Duration, cycleTime : Duration=PT0S, repeat : Int64=0) : EventId
+AddMissionTimeEvent(entryPoint : IEntryPoint, missionTime : Duration, cycleTime : Duration=PT0S, repeat : Int64=0) : EventId
+AddEpochTimeEvent(entryPoint : IEntryPoint, epochTime : DateTime, cycleTime : Duration=PT0S, repeat : Int64=0) : EventId
+AddZuluTimeEvent(entryPoint : IEntryPoint, simulationTime : DateTime, cycleTime : Duration=PT0S, repeat : Int64=0) : EventId
+SetEventSimulationTime(event : EventId, simulationTime : Duration)
+SetEventMissionTime(event : EventId, missionTime : Duration)
+SetEventEpochTime(event : EventId, epochTime : DateTime)
+SetEventZuluTime(event : EventId, zuluTime : DateTime)
+SetEventCycleTime(event : EventId, cycleTime : Duration)
+SetEventCount(event : EventId, count : Int64)
+RemoveEvent(event : EventId)
<<exception>>
InvalidEventTime
<<exception>>
InvalidCycleTime

Figure 4-4: IScheduler

Base Interfaces

Smp::IService

Table 4-17: Constants of IScheduler

Name	Type	Description	Value
SMP_Scheduler	String8	Name of the Scheduler service.	Scheduler

Table 4-18: Operations of IScheduler

Name	Description
AddEpochTimeEvent	Add event to scheduler that is called based on epoch time.
AddImmediateEvent	Add an immediate event to the scheduler.
AddMissionTimeEvent	Add event to scheduler that is called based on mission time.
AddSimulationTimeEvent	Add event to scheduler that is called based on simulation time.
AddZuluTimeEvent	Add event to scheduler that is called based on Zulu time.
RemoveEvent	Remove an event from the scheduler.
SetEventCount	Update the count of an existing event on the scheduler.

Name	Description
SetEventCycleTime	Update cycle time of an existing event on the scheduler.
SetEventEpochTime	Update when an existing epoch time event on the scheduler (an event that has been registered using AddEpochTimeEvent()) shall be triggered.
SetEventMissionTime	Update when an existing mission time event on the scheduler shall be triggered.
SetEventSimulationTime	Update when an existing simulation time event on the scheduler shall be triggered.
SetEventZuluTime	Update when an existing zulu time event on the scheduler shall be triggered.

4.3.1.1 Add Epoch Time Event

Add event to scheduler that is called based on epoch time.

An event with repeat=0 is not cyclic. It will be removed automatically after is has been triggered.

An event with repeat>0 is cyclic, and will be repeated repeat times. Therefore, it will be called repeat+1 times, and then it will be removed automatically.

An event with repeat=-1 is cyclic as well, but it will be triggered forever, unless it is removed from the scheduler using the RemoveEvent() method.

For a cyclic event, the cycleTime needs to be positive. For non-cyclic events, it is ignored.

The epochTime must not be before the current epoch time of the ITimeKeeper service. Otherwise, the event will never be executed, but immediately removed.

Prototype

public EventId AddEpochTimeEvent(in IEntryPoint entryPoint, in DateTime epochTime, in Duration cycleTime, in Int64 repeat) raises (InvalidCycleTime, InvalidEventTime);

Table 4-19: Parameters of AddEpochTimeEvent

Name	Dir.	Type	Description
	return	EventId	Event identifier that can be used to change or remove event.
entryPoint	in	IEntryPoint	Entry point to call from event.
epochTime	in	DateTime	Epoch time when to trigger the event for the first time.
cycleTime	in	Duration	Duration between two triggers of the event.
repeat	in	Int64	Number of times the event shall be repeated, or 0 for a single event, or -1 for no limit.

Exceptions

Smp::Services::IScheduler::InvalidCycleTime,
Smp::Services::IScheduler::InvalidEventTime

4.3.1.2 Add Immediate Event

Add an immediate event to the scheduler.

An immediate event is an event that will be added as a simulation time event (with simulation delta time of 0) at the front of the event queue. As such, it will be executed when the scheduler processes its simulation time events again, but not immediately in the call to AddImmediateEvent().

When the simulator is in Standby state, simulation time does not progress, and simulation time events (including immediate events) are not processed.

Remark: To execute an entry point immediately without going through the scheduler, its Execute() method can be called.

Prototype

public EventId AddImmediateEvent(in IEntryPoint entryPoint);

Table 4-20: Parameters of AddImmediateEvent

Name	Dir.	Type	Description
entryPoint	in	IEntryPoint	Entry point to call from event.
	return	EventId	Event identifier that can be used to change or remove event.

Exceptions

None

4.3.1.3 Add Mission Time Event

Add event to scheduler that is called based on mission time.

An event with repeat=0 is not cyclic. It will be removed automatically after is has been triggered.

An event with repeat>0 is cyclic, and will be repeated repeat times. Therefore, it will be called repeat+1 times, and then it will be removed automatically.

An event with repeat=-1 is cyclic as well, but it will be triggered forever, unless it is removed from the scheduler using the RemoveEvent() method.

For a cyclic event, the cycleTime needs to be positive. For non-cyclic events, it is ignored.

The missionTime must not be before the current mission time of the ITimeKeeper service. Otherwise, the event will never be executed, but immediately removed.

Prototype

public EventId AddMissionTimeEvent(in IEntryPoint entryPoint, in Duration missionTime, in Duration cycleTime, in Int64 repeat) raises (InvalidCycleTime, InvalidEventTime);

Table 4-21: Parameters of AddMissionTimeEvent

Name	Dir.	Type	Description
	return	EventId	Event identifier that can be used to change or remove event.
entryPoint	in	IEntryPoint	Entry point to call from event.
missionTime	in	Duration	Absolute mission time when to trigger the event for the first time.
cycleTime	in	Duration	Duration between two triggers of the event.
repeat	in	Int64	Number of times the event shall be repeated, or 0 for a single event, or -1 for no limit.

Exceptions

Smp::Services::IScheduler::InvalidCycleTime,
Smp::Services::IScheduler::InvalidEventTime

4.3.1.4 Add Simulation Time Event

Add event to scheduler that is called based on simulation time.

An event with repeat=0 is not cyclic. It will be removed automatically after is has been triggered.

An event with repeat>0 is cyclic, and will be repeated repeat times. Therefore, it will be called repeat+1 times, and then it will be removed automatically.

An event with repeat=-1 is cyclic as well, but it will be triggered forever, unless it is removed from the scheduler using the RemoveEvent() method.

For a cyclic event, the cycleTime needs to be positive. For non-cyclic events, it is ignored.

The simulationTime must not be negative. Otherwise, the event will never be executed, but immediately removed.

Prototype

public EventId AddSimulationTimeEvent(in IEntryPoint entryPoint, in Duration simulationTime, in Duration cycleTime, in Int64 repeat) raises (InvalidCycleTime, InvalidEventTime);

Table 4-22: Parameters of AddSimulationtimeEvent

Name	Dir.	Type	Description
	return	EventId	Event identifier that can be used to change or remove event.
entryPoint	in	IEntryPoint	Entry point to call from event.
simulationTime	in	Duration	Duration from now when to trigger the event for the first time.
cycleTime	in	Duration	Duration between two triggers of the event.
repeat	in	Int64	Number of times the event shall be repeated, or 0 for a single event, or -1 for no limit.

Exceptions

Smp::Services::IScheduler::InvalidCycleTime,
Smp::Services::IScheduler::InvalidEventTime

4.3.1.5 Add Zulu Time Event

Add event to scheduler that is called based on Zulu time.

An event with repeat=0 is not cyclic. It will be removed automatically after is has been triggered.

An event with repeat>0 is cyclic, and will be repeated repeat times. Therefore, it will be called repeat+1 times, and then it will be removed automatically.

An event with repeat=-1 is cyclic as well, but it will be triggered forever, unless it is removed from the scheduler using the RemoveEvent() method.

For a cyclic event, the cycleTime needs to be positive. For non-cyclic events, it is ignored.

The zuluTime must not be before the current Zulu time of the ITimeKeeper service. Otherwise, the event will never be executed, but immediately removed.

Prototype

public EventId AddZuluTimeEvent(in IEntryPoint entryPoint, in DateTime simulationTime, in Duration cycleTime, in Int64 repeat) raises (InvalidCycleTime, InvalidEventTime);

Table 4-23: Parameters of AddZuluTimeEvent

Name	Dir.	Type	Description
	return	EventId	Event identifier that can be used to change or remove event.
entryPoint	in	IEntryPoint	Entry point to call from event.
simulationTime	in	DateTime	Absolute (Zulu) time when to trigger the event for the first time.
cycleTime	in	Duration	Duration between two triggers of the event.
repeat	in	Int64	Number of times the event shall be repeated, or 0 for a single event, or -1 for no limit.

Exceptions

Smp::Services::IScheduler::InvalidCycleTime,
Smp::Services::IScheduler::InvalidEventTime

4.3.1.6 Remove Event

Remove an event from the scheduler.

When the given event is not a valid identifier of a scheduler event, the method throws an exception of type InvalidEventId.

An event with count=0 is removed automatically after it has been triggered.

Prototype

public void RemoveEvent(in EventId event) raises (InvalidEventId);

Table 4-24: Parameters of RemoveEvent

Name	Dir.	Type	Description
event	in	EventId	Event identifier of the event to remove.

Exceptions

Smp::Services::InvalidEventId

4.3.1.7 Set Event Count

Update the count of an existing event on the scheduler.

When the given event is not a valid identifier of a scheduler event, the method throws an exception of type InvalidEventId.

An event with count=0 is not cyclic. It will be removed automatically after is has been triggered.

An event with count>0 is cyclic, and will be repeated count times. Therefore, it will be called count+1times, and then it will be removed automatically.

An event with count=-1 is cyclic as well, but it will be triggered forever, unless it is removed from the scheduler using the RemoveEvent() method.

For a cyclic event, the cycleTime needs to be positive. For non-cyclic events, it is ignored.

Prototype

public void SetEventCount(in EventId event, in Int64 count) raises (InvalidEventId);

Table 4-25: Parameters of SetEventCount

Name	Dir.	Type	Description
event	in	EventId	Identifier of event to modify.
count	in	Int64	Number of times the event shall be repeated, or 0 for a single event, or -1 for no limit.

Exceptions

Smp::Services::InvalidEventId

4.3.1.8 Set Event Cycle Time

Update cycle time of an existing event on the scheduler.

When the given event is not a valid identifier of a scheduler event, the method throws an exception of type InvalidEventId.

For a cyclic event, the cycleTime needs to be positive. For non-cyclic events, it is ignored.

Prototype

public void SetEventCycleTime(in EventId event, in Duration cycleTime) raises (InvalidEventId);

Table 4-26: Parameters of SetEventCycleTime

Name	Dir.	Type	Description
event	in	EventId	Identifier of event to modify.
cycleTime	in	Duration	Duration between two triggers of the event.

Exceptions

Smp::Services::InvalidEventId

4.3.1.9 Set Event Epoch Time

Update when an existing epoch time event on the scheduler (an event that has been registered using AddEpochTimeEvent()) shall be triggered.

When the given event Id is not a valid identifier of a scheduler event, the method throws an exception of type InvalidEventId. In case an event is registered under the given event Id but it is not an epoch time event, the method throws an exception of type InvalidEventId as well.

The epochTime must not be before the current epoch time of the ITimeKeeper service. Otherwise, the event will never be executed, but immediately removed.

Prototype

public void SetEventEpochTime(in EventId event, in DateTime epochTime) raises (InvalidEventId);

Table 4-27: Parameters of SetEventEpochTime

Name	Dir.	Type	Description
event	in	EventId	Identifier of event to modify.
epochTime	in	DateTime	Epoch time when to trigger event.

Exceptions

Smp::Services::InvalidEventId

4.3.1.10 Set Event Mission Time

Update when an existing mission time event on the scheduler shall be triggered.

When the given event Id is not a valid identifier of a scheduler event, the method throws an exception of type InvalidEventId. In case an event is registered under the given event Id but it is not an mission time event, the method throws an exception of type InvalidEventId as well.

The missionTime must not be before the current mission time of the ITimeKeeper service. Otherwise, the event will never be executed, but immediately removed.

Prototype

public void SetEventMissionTime(in EventId event, in Duration missionTime) raises (InvalidEventId);

Table 4-28: Parameters of SetEventMissionTime

Name	Dir.	Type	Description
event	in	EventId	Identifier of event to modify.
missionTime	in	Duration	Absolute mission time when to trigger event.

Exceptions

Smp::Services::InvalidEventId

4.3.1.11 Set Event Simulation Time

Update when an existing simulation time event on the scheduler shall be triggered.

When the given event Id is not a valid identifier of a scheduler event, the method throws an exception of type InvalidEventId. In case an event is registered under the given event Id but it is not an simulation time event, the method throws an exception of type InvalidEventId as well.

The simulationTime must not be negative. Otherwise, the event will never be executed, but immediately removed.

Prototype

public void SetEventSimulationTime(in EventId event, in Duration simulationTime) raises (InvalidEventId);

Table 4-29: Parameters of SetEventSimulationTime

Name	Dir.	Type	Description
event	in	EventId	Identifier of event to modify.
simulationTime	in	Duration	Duration from now when to trigger event.

Exceptions

Smp::Services::InvalidEventId

4.3.1.12 Set Event Zulu Time

Update when an existing zulu time event on the scheduler shall be triggered.

When the given event Id is not a valid identifier of a scheduler event, the method throws an exception of type InvalidEventId. In case an event is registered under the given event Id but it is not an zulu time event, the method throws an exception of type InvalidEventId as well.

The zuluTime must not be before the current Zulu time of the ITimeKeeper service. Otherwise, the event will never be executed, but immediately removed.

Prototype

public void SetEventZuluTime(in EventId event, in DateTime zuluTime) raises (InvalidEventId);

Table 4-30: Parameters of SetEventZuluTime

Name	Dir.	Type	Description
event	in	EventId	Identifier of event to modify.
zuluTime	in	DateTime	Absolute (Zulu) time when to trigger event.

Exceptions

Smp::Services::InvalidEventId

4.3.1.13 Invalid Cycle Time

This exception is thrown by one of the AddEvent() methods of the scheduler when the event is a cyclic event (i.e. repeat is not 0), but the cycle time specified is not a positive duration.

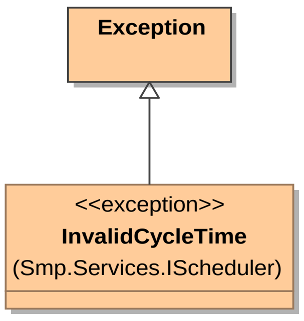


Figure 4-5: Invalid Cycle Time

Fields

None

4.3.1.14 Invalid Event Time

This exception is thrown by one of the AddEvent() methods of the scheduler when the time specified for the first execution of the event is in the past.

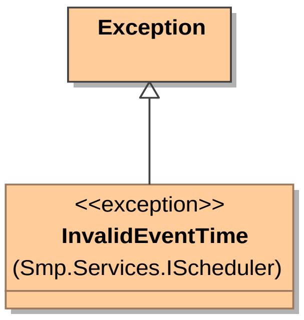


Figure 4-6: Invalid Event Time

Fields

None

4.4 Event Manager

The event manager service provides a global notification mechanism. Components can register entry points with a global event. Several pre-defined event types exist, but applications can define their own, specific global events as well.

Although it is possible that any component triggers one of the pre-defined events by calling the `Emit()` method, models shall not emit pre-defined events, but only user-defined events.

To prevent accidentally emitting pre-defined events, these have been put into a "namespace", i.e. a prefix string "Smp_" has been added. It is recommended that user events include a "namespace" as well, for example "`MyApp_MyEvent1`".

4.4.1 IEvent Manager

This interface gives access to the Event Manager Service.

Components can register entry points with events, and they can define and emit events.

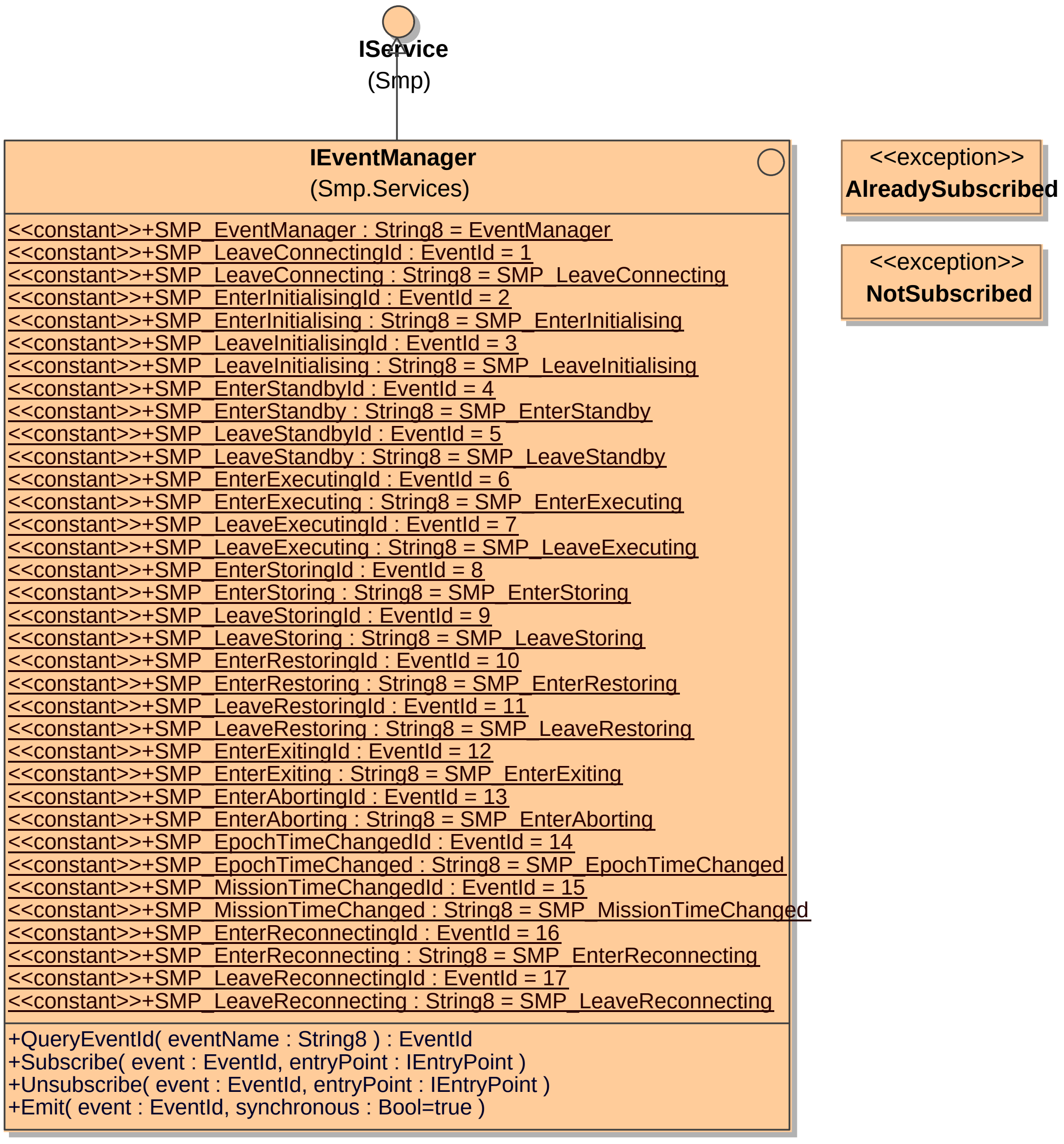

IService
(Smp)
IEventManager
(Smp.Services)
<<constant>>+SMP_EventManager : String8 = EventManager
<<constant>>+SMP_LeaveConnectingId : EventId = 1
<<constant>>+SMP_LeaveConnecting : String8 = SMP_LeaveConnecting
<<constant>>+SMP_EnterInitialisingId : EventId = 2
<<constant>>+SMP_EnterInitialising : String8 = SMP_EnterInitialising
<<constant>>+SMP_LeaveInitialisingId : EventId = 3
<<constant>>+SMP_LeaveInitialising : String8 = SMP_LeaveInitialising
<<constant>>+SMP_EnterStandbyId : EventId = 4
<<constant>>+SMP_EnterStandby : String8 = SMP_EnterStandby
<<constant>>+SMP_LeaveStandbyId : EventId = 5
<<constant>>+SMP_LeaveStandby : String8 = SMP_LeaveStandby
<<constant>>+SMP_EnterExecutingId : EventId = 6
<<constant>>+SMP_EnterExecuting : String8 = SMP_EnterExecuting
<<constant>>+SMP_LeaveExecutingId : EventId = 7
<<constant>>+SMP_LeaveExecuting : String8 = SMP_LeaveExecuting
<<constant>>+SMP_EnterStoringId : EventId = 8
<<constant>>+SMP_EnterStoring : String8 = SMP_EnterStoring
<<constant>>+SMP_LeaveStoringId : EventId = 9
<<constant>>+SMP_LeaveStoring : String8 = SMP_LeaveStoring
<<constant>>+SMP_EnterRestoringId : EventId = 10
<<constant>>+SMP_EnterRestoring : String8 = SMP_EnterRestoring
<<constant>>+SMP_LeaveRestoringId : EventId = 11
<<constant>>+SMP_LeaveRestoring : String8 = SMP_LeaveRestoring
<<constant>>+SMP_EnterExitingId : EventId = 12
<<constant>>+SMP_EnterExiting : String8 = SMP_EnterExiting
<<constant>>+SMP_EnterAbortingId : EventId = 13
<<constant>>+SMP_EnterAborting : String8 = SMP_EnterAborting
<<constant>>+SMP_EpochTimeChangedId : EventId = 14
<<constant>>+SMP_EpochTimeChanged : String8 = SMP_EpochTimeChanged
<<constant>>+SMP_MissionTimeChangedId : EventId = 15
<<constant>>+SMP_MissionTimeChanged : String8 = SMP_MissionTimeChanged
<<constant>>+SMP_EnterReconnectingId : EventId = 16
<<constant>>+SMP_EnterReconnecting : String8 = SMP_EnterReconnecting
<<constant>>+SMP_LeaveReconnectingId : EventId = 17
<<constant>>+SMP_LeaveReconnecting : String8 = SMP_LeaveReconnecting
+QueryEventId(eventName : String8) : EventId
+Subscribe(event : EventId, entryPoint : IEntryPoint)
+Unsubscribe(event : EventId, entryPoint : IEntryPoint)
+Emit(event : EventId, synchronous : Bool=true)
<<exception>>
AlreadySubscribed
<<exception>>
NotSubscribed

Figure 4-7: IEvent Manager

Base Interfaces

Smp::IService

Table 4-31: Constants of IEventManager

Name	Type	Description	Value
SMP_EnterAborting	String8	Enter Aborting state.	SMP_EnterAborting
SMP_EnterAbortingId	EventId	This event is raised when entering the Aborting state with the Abort() state transition command from any other state.	13
SMP_EnterExecuting	String8	Enter Executing state.	SMP_EnterExecuting
SMP_EnterExecutingId	EventId	This event is raised when entering the Executing state with the Run() state transition command from Standby state.	6
SMP_EnterExiting	String8	Enter Exiting state.	SMP_EnterExiting
SMP_EnterExitingId	EventId	This event is raised when entering the Exiting state with the Exit() state transition command from Standby state.	12
SMP_EnterInitialising	String8	Enter Initialising state.	SMP_EnterInitialising
SMP_EnterInitialisingId	EventId	This event is raised when entering the Initialising state with an automatic state transition from Connecting state, or with the Initialise() state transition.	2
SMP_EnterReconnecting	String8	Enter Reconnecting state.	SMP_EnterReconnecting
SMP_EnterReconnectingId	EventId	This event is raised when entering the Reconnecting state with the Reconnect() state transition from Standby state.	16
SMP_EnterRestoring	String8	Enter Restoring state.	SMP_EnterRestoring
SMP_EnterRestoringId	EventId	This event is raised when entering the Restoring state with the Restore() state transition command from Standby state.	10
SMP_EnterStandby	String8	Enter Standby state.	SMP_EnterStandby
SMP_EnterStandbyId	EventId	This event is raised when entering the Standby state with an automatic state transition from Initialising, Storing or Restoring state, or with the Hold() state transition command from Executing state.	4
SMP_EnterStoring	String8	Enter Storing state.	SMP_EnterStoring

Name	Type	Description	Value
SMP_EnterStoringId	EventId	This event is raised when entering the Storing state with the Store() state transition command from Standby state.	8
SMP_EpochTimeChanged	String8	Epoch Time has changed.	SMP_EpochTimeChanged
SMP_EpochTimeChangedId	EventId	This event is raised when changing the epoch time with the SetEpochTime() method of the time keeper service.	14
SMP_EventManager	String8	Name of the EventManager service.	EventManager
SMP_LeaveConnecting	String8	Leave Connecting state.	SMP_LeaveConnecting
SMP_LeaveConnectingId	EventId	This event is raised when leaving the Connecting state with an automatic state transition to Initializing state.	1
SMP_LeaveExecuting	String8	Leave Executing state.	SMP_LeaveExecuting
SMP_LeaveExecutingId	EventId	This event is raised when leaving the Executing state with the Hold() state transition command to Standby state.	7
SMP_LeaveInitialising	String8	Leave Initialising state.	SMP_LeaveInitialising
SMP_LeaveInitialisingId	EventId	This event is raised when leaving the Initialising state with an automatic state transition to Standby state.	3
SMP_LeaveReconnecting	String8	Leave Reconnecting state.	SMP_LeaveReconnecting
SMP_LeaveReconnectingId	EventId	This event is raised when leaving the Reconnecting state with an automatic state transition to Standby state.	17
SMP_LeaveRestoring	String8	Leave Restoring state.	SMP_LeaveRestoring
SMP_LeaveRestoringId	EventId	This event is raised when leaving the Restoring state with an automatic state transition to Standby state.	11
SMP_LeaveStandby	String8	Leave Standby state.	SMP_LeaveStandby

Name	Type	Description	Value
SMP_LeaveStandbyId	EventId	This event is raised when leaving the Standby state with the Run() state transition command to Executing state, with the Store() state transition command to Storing state, with the Restore() state transition command to Restoring state, or with the Initialise() state transition command to Initialising state.	5
SMP_LeaveStoring	String8	Leave Storing state.	SMP_LeaveStoring
SMP_LeaveStoringId	EventId	This event is raised when leaving the Storing state with an automatic state transition to Standby state.	9
SMP_MissionTimeChanged	String8	Mission time has changed.	SMP_MissionTimeChanged
SMP_MissionTimeChangedId	EventId	This event is raised when changing the mission time with one of the SetMissionTime() and SetMissionStart() methods of the time keeper service.	15

Table 4-32: Operations of IEventManager

Name	Description
Emit	Emit a global event.
QueryEventId	Get unique event identifier for an event name.
Subscribe	Subscribe entry point to a global event.
Unsubscribe	Unsubscribe entry point from a global event.

4.4.1.1 Emit

Emit a global event.

This will call all entry points that are subscribed to the global event with the given identifier at the time Emit() is called. Entry point subscription/unsubscription during the execution of Emit() is taken into account the next time Emit() is called. Entry points will be called in the order they have been subscribed to the global event.

Prototype

public void Emit(in EventId event, in Bool synchronous);

Table 4-33: Parameters of Emit

Name	Dir.	Type	Description
event	in	EventId	Event identifier of global event to emit.
synchronous	in	Bool	Flag whether to emit the given event synchronously (the default) or asynchronously.

Exceptions

None

4.4.1.2 Query Event Id

Get unique event identifier for an event name.

It is guaranteed that this method will always return the same value when called with the same event name. This holds for pre-defined event names as well as for user-defined events.

Prototype

public EventId QueryEventId(in String8 eventName);

Table 4-34: Parameters of QueryEventId

Name	Dir.	Type	Description
	return	EventId	Event identifier for global event with given name.
eventName	in	String8	Name of the global event.

Exceptions

None

4.4.1.3 Subscribe

Subscribe entry point to a global event.

This method raises an exception of type InvalidEventId when called with an invalid event identifier. When the entry point is already subscribed to the same event, an exception of type AlreadySubscribed is raised.

An entry point can only be subscribed once to an event.

Prototype

public void Subscribe(in EventId event, in IEntryPoint entryPoint) raises (InvalidEventId, AlreadySubscribed);

Table 4-35: Parameters of Subscribe

Name	Dir.	Type	Description
event	in	EventId	Event identifier of global event to subscribe to.
entryPoint	in	IEntryPoint	Entry point to subscribe to global event.

Exceptions

Smp::Services::InvalidEventId,
Smp::Services::IEventManager::AlreadySubscribed

4.4.1.4 Unsubscribe

Unsubscribe entry point from a global event.

This method raises an exception of type InvalidEventId when called with an invalid event identifier. When the entry point is not subscribed to the event, an exception of type NotSubscribed is raised.

An entry point can only be unsubscribed from an event when it has been subscribed earlier using Subscribe().

Prototype

public void Unsubscribe(in EventId event, in IEntryPoint entryPoint) raises (InvalidEventId, NotSubscribed);

Table 4-36: Parameters of Unsubscribe

Name	Dir.	Type	Description
event	in	EventId	Event identifier of global event to unsubscribe from.
entryPoint	in	IEntryPoint	Entry point to unsubscribe from global event.

Exceptions

Smp::Services::InvalidEventId, Smp::Services::IEventManager::NotSubscribed

4.4.1.5 Already Subscribed

This exception is raised when trying to subscribe an entry point to an event that is already subscribed.

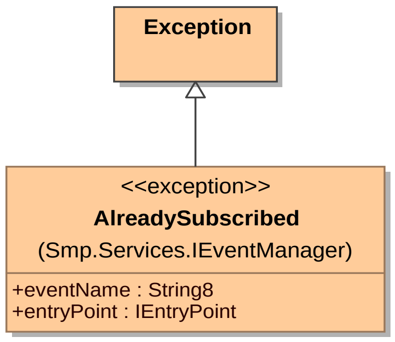


Figure 4-8: Already Subscribed

Table 4-37: Fields of AlreadySubscribed

Name	Type	Description
entryPoint	IEntryPoint	Entry point that is already subscribed.
eventName	String8	Name of event the entry point is already subscribed to.

4.4.1.6 Not Subscribed

This exception is raised when trying to unsubscribe an entry point from an event that is not subscribed to it.

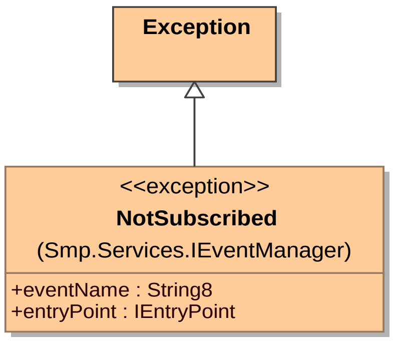
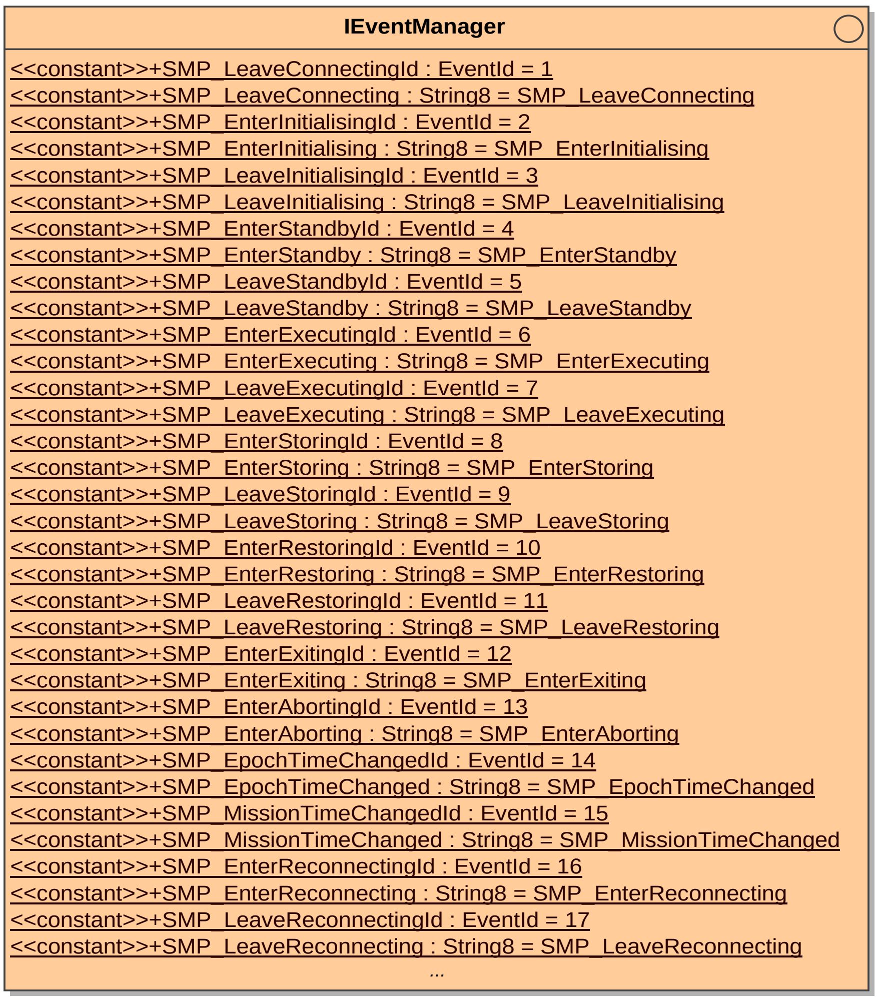


Figure 4-9: Not Subscribed

Table 4-38: Fields of NotSubscribed

Name	Type	Description
entryPoint	IEntryPoint	Entry point that is not subscribed.
eventName	String8	Name of event the entry point is not subscribed to.

4.4.2 Predefined Event Types



```
IEventManager
<<constant>>+SMP_LeaveConnectingId : EventId = 1
<<constant>>+SMP_LeaveConnecting : String8 = SMP_LeaveConnecting
<<constant>>+SMP_EnterInitialisingId : EventId = 2
<<constant>>+SMP_EnterInitialising : String8 = SMP_EnterInitialising
<<constant>>+SMP_LeaveInitialisingId : EventId = 3
<<constant>>+SMP_LeaveInitialising : String8 = SMP_LeaveInitialising
<<constant>>+SMP_EnterStandbyId : EventId = 4
<<constant>>+SMP_EnterStandby : String8 = SMP_EnterStandby
<<constant>>+SMP_LeaveStandbyId : EventId = 5
<<constant>>+SMP_LeaveStandby : String8 = SMP_LeaveStandby
<<constant>>+SMP_EnterExecutingId : EventId = 6
<<constant>>+SMP_EnterExecuting : String8 = SMP_EnterExecuting
<<constant>>+SMP_LeaveExecutingId : EventId = 7
<<constant>>+SMP_LeaveExecuting : String8 = SMP_LeaveExecuting
<<constant>>+SMP_EnterStoringId : EventId = 8
<<constant>>+SMP_EnterStoring : String8 = SMP_EnterStoring
<<constant>>+SMP_LeaveStoringId : EventId = 9
<<constant>>+SMP_LeaveStoring : String8 = SMP_LeaveStoring
<<constant>>+SMP_EnterRestoringId : EventId = 10
<<constant>>+SMP_EnterRestoring : String8 = SMP_EnterRestoring
<<constant>>+SMP_LeaveRestoringId : EventId = 11
<<constant>>+SMP_LeaveRestoring : String8 = SMP_LeaveRestoring
<<constant>>+SMP_EnterExitingId : EventId = 12
<<constant>>+SMP_EnterExiting : String8 = SMP_EnterExiting
<<constant>>+SMP_EnterAbortingId : EventId = 13
<<constant>>+SMP_EnterAborting : String8 = SMP_EnterAborting
<<constant>>+SMP_EpochTimeChangedId : EventId = 14
<<constant>>+SMP_EpochTimeChanged : String8 = SMP_EpochTimeChanged
<<constant>>+SMP_MissionTimeChangedId : EventId = 15
<<constant>>+SMP_MissionTimeChanged : String8 = SMP_MissionTimeChanged
<<constant>>+SMP_EnterReconnectingId : EventId = 16
<<constant>>+SMP_EnterReconnecting : String8 = SMP_EnterReconnecting
<<constant>>+SMP_LeaveReconnectingId : EventId = 17
<<constant>>+SMP_LeaveReconnecting : String8 = SMP_LeaveReconnecting
...
```

Figure 4-10: Predefined Event Types

The Event Manager supports some global event names and ids defined for state changes of the simulation environment, or for a modified epoch or mission time. The state transition events clearly indicate in their names whether they are emitted when entering the corresponding state, or when leaving it.

The events indicating changes in either mission or epoch time are raised after the corresponding time has been changed, so that an immediate call to the time keeper service will return the new epoch or mission time, respectively.

The names and ids of the predefined event kinds are as listed in the table below. As most of these events relate to state changes, the state diagram of the simulation environment is shown in Figure 3-71 (Simulation Environment State Diagram with State Transition Methods) on page 104.

Table 4-39: Predefined Event Types

Event Name	Id	Description
LeaveConnecting	1	This event is raised when leaving the Connecting state with an automatic state transition to Initializing state.
EnterInitialising	2	This event is raised when entering the Initialising state with an automatic state transition from Connecting state, or with the Initialise() state transition.
LeaveInitialising	3	This event is raised when leaving the Initialising state with an automatic state transition to Standby state.
EnterStandby	4	This event is raised when entering the Standby state with an automatic state transition from Initialising, Storing or Restoring state, or with the Hold() state transition command from Executing state.
LeaveStandby	5	This event is raised when leaving the Standby state with the Run()state transition command to Executing state, with the Store()state transition command to Storing state, with the Restore()state transition command to Restoring state, or with the Initialise() state transition command to Initialising state.
EnterExecuting	6	This event is raised when entering the Executing state with the Run() state transition command from Standby state.
LeaveExecuting	7	This event is raised when leaving the Executing state with the Hold() state transition command to Standby state.
EnterStoring	8	This event is raised when entering the Storing state with the Store() state transition command from Standby state.
LeaveStoring	9	This event is raised when leaving the Storing state with an automatic state transition to Standby state.
EnterRestoring	10	This event is raised when entering the Restoring state with the Restore() state transition command from Standby state.
LeaveRestoring	11	This event is raised when leaving the Restoring state with an automatic state transition to Standby state.
EnterExiting	12	This event is raised when entering the Exiting state with the Exit() state transition command from Standby state.
EnterAborting	13	This event is raised when entering the Aborting state with the Abort() state transition command from any other state.
EpochTimeChanged	14	This event is raised when changing the epoch time with the SetEpochTime() method of the time keeper service.
MissionTimeChanged	15	This event is raised when changing the mission time with one of the SetMissionTime() and SetMissionStart() methods of the time keeper service.
EnterReconnecting	16	This event is raised when entering the Reconnecting state with the Reconnect() state transition from Standby state.
LeaveReconnecting	17	This event is raised when leaving the Reconnecting state with an automatic state transition to Standby state.

User-defined event ids can be generated using the `QueryEventId()` method, which will return the same identifier every time it is called with the identical event name.

4.5 Resolver

Components can use the Resolver to resolve references to other components by name. References can either be specified using a fully qualified path, or using a path relative to some other component.

4.5.1 IResolver

This interface gives access to the Resolver Service.

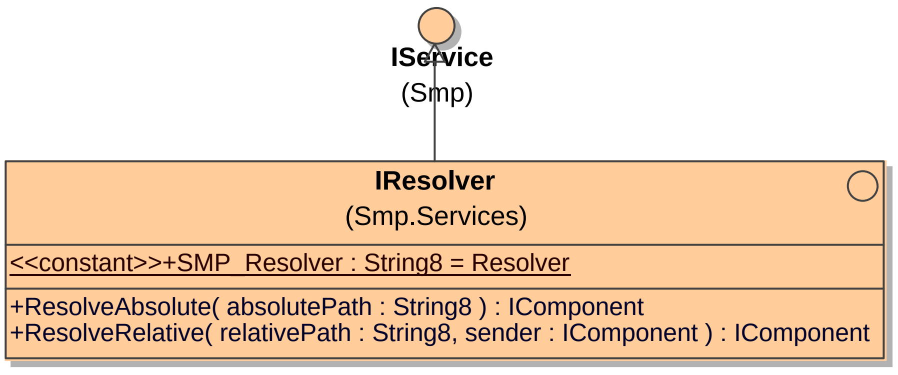

IService
(Smp)
IResolver
(Smp.Services)
<<constant>>+SMP_Resolver : String8 = Resolver
+ResolveAbsolute(absolutePath : String8) : IComponent
+ResolveRelative(relativePath : String8, sender : IComponent) : IComponent

Figure 4-11: IResolver

Base Interfaces

Smp::IService

Table 4-40: Constants of IResolver

Name	Type	Description	Value
SMP_Resolver	String8	Name of the Resolver service.	Resolver

Table 4-41: Operations of IResolver

Name	Description
ResolveAbsolute	Resolve reference to component via absolute path.
ResolveRelative	Resolve reference to component via relative path.

4.5.1.1 Resolve Absolute

Resolve reference to component via absolute path.

An absolute path contains the name of either the Models or the Services container, but not the name of the simulator, although the simulator itself is the top-level component. This allows keeping names as short as possible, and avoids a dependency on the name of the simulator itself.

Prototype

public IComponent ResolveAbsolute(in String8 absolutePath);

Table 4-42: Parameters of ResolveAbsolute

Name	Dir.	Type	Description
	return	IComponent	Component identified by path, or null if no component with the given path could be found.
absolutePath	in	String8	Absolute path to component in simulation.

Exceptions

None

4.5.1.2 Resolve Relative

Resolve reference to component via relative path.

Prototype

public IComponent ResolveRelative(in String8 relativePath, in IComponent sender);

Table 4-43: Parameters of ResolveRelative

Name	Dir.	Type	Description
	return	IComponent	Component identified by path, or null if no component with the given path could be found.
relativePath	in	String8	Relative path to component in simulation.
sender	in	IComponent	Component that asks for resolving the reference.

Exceptions

None

4.5.2 Component Paths

The Resolver can be used to resolve component references by name, either by a full path or by a relative path with respect to a given component. Therefore, it needs to be specified how path names have to be built. As all components in the tree of a simulation have a name, all that needs to be specified is how these names compose a path name, and how the parent component is identified.

Rule 1: Component names are assembled with one of the following characters: "\", "/", "!"

Rule 2: The parent component is specified by the following string: ".."

Examples:

```
Models/Satellite/Receivers/Receiver1
Services!Logger
..\..\Transmitters\Transmitter4
```

4.6 Link Registry

The Link Registry service maintains a list of all links between components, and can be used to find out whether a component can be safely removed from the simulation.

4.6.1 ILink Registry

This interface is implemented by the Link Registry Service.

The link registry maintains a global collection of links between components. Links can be added and removed, and can be queried for. Further, the link registry supports fetching and removing all links to a given target.

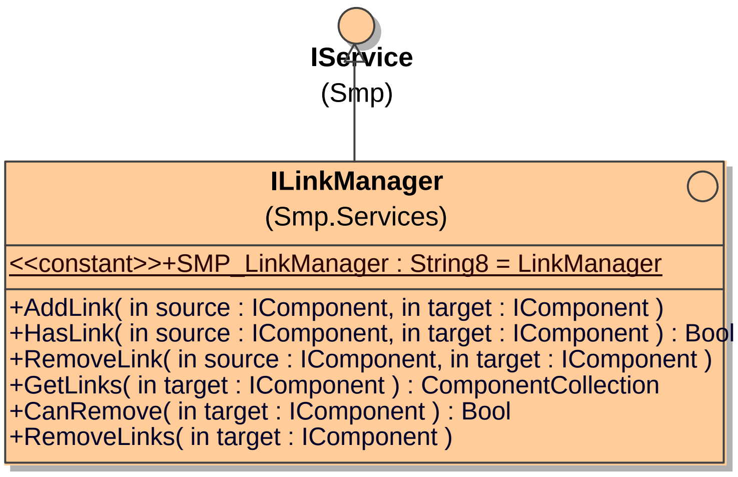

IService
(Smp)
ILinkManager
(Smp.Services)
<<constant>>+SMP_LinkManager : String8 = LinkManager
+AddLink(in source : IComponent, in target : IComponent)
+HasLink(in source : IComponent, in target : IComponent) : Bool
+RemoveLink(in source : IComponent, in target : IComponent)
+GetLinks(in target : IComponent) : ComponentCollection
+CanRemove(in target : IComponent) : Bool
+RemoveLinks(in target : IComponent)

Figure 4-12: ILink Registry

Base Interfaces

Smp::IService

Table 4-44: Constants of ILinkRegistry

Name	Type	Description	Value
SMP_LinkRegistry	String8	Name of the LinkRegistry service.	LinkRegistry

Table 4-45: Operations of ILinkRegistry

Name	Description
AddLink	Add a link from source component to target component.
CanRemove	Returns true if all sources linking to the given target can be asked to remove their link(s), false otherwise.
GetLinks	Returns a collection of all sources that have a link to the given target.
HasLink	Returns true if a link between source and target exists, false otherwise.
RemoveLink	Remove a link between source and target that has been added to the service using AddLink() before.
RemoveLinks	Removes all links to the given target.

4.6.1.1 Add Link

Add a link from source component to target component.

This method informs the link registry that a link between two components has been created. The link registry does not create this link, it only gets told about its existence.

This method can be called several times with the same arguments, when a source component has several links to the same target component.

Prototype

public void AddLink(in IComponent source, in IComponent target);

Table 4-46: Parameters of AddLink

Name	Dir.	Type	Description
source	in	IComponent	Source component of link (i.e. the component that links to another component).
target	in	IComponent	Target component of link (i.e. the component that is being linked to by another component).

Exceptions

None

4.6.1.2 Can Remove

Returns true if all sources linking to the given target can be asked to remove their link(s), false otherwise.

This method checks whether all sources that have a link to the given target implement the optional interface ILinkingComponent. If so, they can be asked to remove their links. The method returns false if at least one source exists that does not implement the ILinkingComponent interface.

Prototype

public Bool CanRemove(in IComponent target);

Table 4-47: Parameters of CanRemove

Name	Dir.	Type	Description
	return	Bool	True if all links to the given target can be removed, false otherwise.
target	in	IComponent	Target component to check for links.

Exceptions

None

4.6.1.3 Get Links

Returns a collection of all sources that have a link to the given target.

This method returns the collection of source components for which a link to the given target component has been added to the link registry.

Prototype

public ComponentCollection GetLinks(in IComponent target);

Table 4-48: Parameters of GetLinks

Name	Dir.	Type	Description
	return	ComponentCollection	Collection of source components which link to the given target.
target	in	IComponent	Target component to returns links for.

Exceptions

None

4.6.1.4 Has Link

Returns true if a link between source and target exists, false otherwise.

Prototype

public Bool HasLink(in IComponent source, in IComponent target);

Table 4-49: Parameters of HasLink

Name	Dir.	Type	Description
	return	Bool	True if such a link has been added before (and not been removed), false otherwise.
source	in	IComponent	Source component of link (i.e. the component that links to another component).
target	in	IComponent	Target component of link (i.e. the component that is being linked to by another component).

Exceptions

None

4.6.1.5 Remove Link

Remove a link between source and target that has been added to the service using AddLink() before.

This method informs the link registry that a link between two components has been deleted. The link registry does not delete this link, it only gets told about the fact that the link no longer exists.

This method can be called several times with the same arguments, when a source component had several links to the same target component.

Prototype

public void RemoveLink(in IComponent source, in IComponent target);

Table 4-50: Parameters of RemoveLink

Name	Dir.	Type	Description
source	in	IComponent	Source component of link (i.e. the component that links to another component).
target	in	IComponent	Target component of link (i.e. the component that is being linked to by another component).

Exceptions

None

4.6.1.6 Remove Links

Removes all links to the given target.

This method calls the RemoveLinks() method of all source components that implement the optional ILinkingComponent interface, so it asks all link sources to remove their links to the given target.

Prototype

public void RemoveLinks(in IComponent target);

Table 4-51: Parameters of RemoveLinks

Name	Dir.	Type	Description
target	in	IComponent	Target component of link (i.e. the component that is being linked to by other components).

Exceptions

No

Annex A (informative) Component model catalogue

This Annex A contains the Component Model in the format of an SMDL catalogue.

A.1 Smp Catalogue

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalogue:Catalogue
xmlns:Elements="http://www.esa.int/2008/02/Core/Elements"

xmlns:Types="http://www.esa.int/2008/02/Core/Types"

xmlns:Catalogue="http://www.esa.int/2008/02/Smdl/Catalogue"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:xlink="http://www.w3.org/1999/xlink"

xsi:schemaLocation="http://www.esa.int/2008/02/Smdl/Catalogue
../xsd/Smdl/Catalogue.xsd"
                     Id="Smp.cat"
                     Name="Smp"
                     Creator="ECSS E40-07"
                     Date="2008-04-14T09:36:29.421+01:00"
                     Title="SMP Component Model"
                     Version="1.0">
      <Description>Contains the SMP Component Model as SMDL
Catalogue</Description>
   <Namespace Id="Smp" Name="Smp">
      <Description>SMP standard types and
interfaces.</Description>
      <Namespace Id="Smp.Attributes" Name="Attributes">
         <Description>Namespace for standard
attributes</Description>
         <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.Fallible" Name="Fallible"
               Visibility="public"
```

```
            Uuid="8596d697-fb84-41ce-a685-6912006ed660">
         <Description>The Fallible attribute marks a model
as being fallible. A fallible model provides support for model
failure status, i.e. the model can be assigned one or more
failures that can be switched and queried during runtime. The
default value for this attribute is false, which corresponds
to no support for model failures.</Description>
         <Type xlink:title="Bool" xlink:href="#Smp.Bool"/>
         <Default xsi:type="Types:BoolValue" Value="true"/>
         <Usage>Model</Usage>
      </Type>
      <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.Initialisation"
            Name="Initialisation"
            Visibility="public"
            Uuid="8596d697-fb84-41ce-a685-6912006ed680">
         <Description>This attribute can be applied to a
Task to specify that it is an initialisation task.
   When a schedule file with one or more initialisation tasks
is loaded, these tasks are added to the scheduler as
initialisation entry points (via
IScheduler::AddInitEntryPoint()) and therefore executed in the
simulator's Initialising state.</Description>
         <Type xlink:title="Bool" xlink:href="#Smp.Bool"/>
         <Default xsi:type="Types:BoolValue" Value="true"/>
         <Usage>Task</Usage>
      </Type>
      <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.FieldUpdate"
            Name="FieldUpdate"
            Visibility="public"
            Uuid="8596d697-fb84-41ce-a685-6912006ed681">
         <Description>This attribute can either be applied
to a Trigger to specify the update behaviour for associated
fields when the entry point is executed, or to a Schedule in
order to define the global default behaviour within the
schedule.
   When the default behaviour ("None") is selected, field
values are &lt;i&gt;not&lt;/i&gt; updated automatically when
the entry point is executed. In this case, all field updates
must be explicitly scheduled via Transfer elements.
   When the "Pull" behaviour is selected, all input fields
associated with the entry point are updated from the linked
outputs &lt;i&gt;before&lt;/i&gt; the entry point is called.
   When the "Push" behaviour is selected, the values of all
output fields associated with the entry point are
automatically transferred to their linked input fields (in
other models) &lt;i&gt;after&lt;/i&gt; the entry point has
been called.</Description>
         <Type xlink:title="FieldUpdateKind"
xlink:href="#Smp.Attributes.FieldUpdateKind"/>
```

```
            <Default xsi:type="Types:Int32Value" Value="0"/>
            <Usage>Trigger</Usage>
            <Usage>Schedule</Usage>
         </Type>
         <Type xsi:type="Types:Enumeration"
Id="Smp.Attributes.FieldUpdateKind"
               Name="FieldUpdateKind"
               Visibility="public"
               Uuid="cdce9add-f196-11dc-a846-558902839034">
            <Description>This enumeration allows to specify
the behaviour when a Trigger is updated.</Description>
            <Literal Id="Smp.Attributes.FieldUpdateKind.None"
Name="None" Value="0">
               <Description>Field values are not updated
automatically when the entry point is executed. In this case,
all field updates must be explicitly scheduled via Transfer
elements.</Description>
            </Literal>
            <Literal Id="Smp.Attributes.FieldUpdateKind.Pull"
Name="Pull" Value="1">
               <Description>All input fields associated with
the entry point are updated from the linked outputs
&lt;i&gt;before&lt;/i&gt; the entry point is
called.</Description>
            </Literal>
            <Literal Id="Smp.Attributes.FieldUpdateKind.Push"
Name="Push" Value="2">
               <Description>The values of all output fields
associated with the entry point are automatically transferred
to their linked input fields (in other models)
&lt;i&gt;after&lt;/i&gt; the entry point has been
called.</Description>
            </Literal>
         </Type>
         <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.Forcible" Name="Forcible"
               Visibility="public"
               Uuid="8596d697-fb84-41ce-a685-6912006ed661">
            <Description>The Forcible attribute marks a field
as being forcible. A forcible field provides support for
forcing, i.e. the field value can be forced to a specified
value during runtime. The following semantics apply for the
different kinds of fields:

   &lt;ul&gt;
   &lt;li&gt;Input field: The forced value is provided to the
model by the simulation environment. The containing model must
internally operate on the field value obtained via the
IField.GetValue() method, which ensures that the forced value
is obtained when forcing is enabled; otherwise the real value
is obtained.&lt;/li&gt;
```

```
   &lt;li&gt;Ouput field or Input/Output field: The forced
value is provided to all connected input fields by the
simulation environment (using the IForcibleField interface)
according to the Field Links specified in the SMDL Assembly.
The containing model must internally operate on the real
(unforced) field value.&lt;/li&gt;
   &lt;/ul&gt;
   The default value for this attribute is false, which
corresponds to no support for forcing.</Description>
            <Type xlink:title="Bool" xlink:href="#Smp.Bool"/>
            <Default xsi:type="Types:BoolValue" Value="true"/>
            <Usage>Property</Usage>
         </Type>
         <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.Abstract" Name="Abstract"
               Visibility="public"
               Uuid="8596d697-fb84-41ce-a685-6912006ed6c1">
            <Description>The Abstract attribute specifies that
an operation or property is abstract, i.e. that it must be
overridden in a derived type. The default value for this
attribute is false.</Description>
            <Type xlink:title="Bool" xlink:href="#Smp.Bool"/>
            <Default xsi:type="Types:BoolValue" Value="true"/>
            <Usage>Operation</Usage>
            <Usage>Property</Usage>
         </Type>
         <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.BaseClass" Name="BaseClass"
               Visibility="public"
               Uuid="8596d697-fb84-41ce-a685-6912006ed6c2">
            <Description>The BaseClass attribute specifies a
name of a C++ class that shall be used as base class
(implementation inheritance) for the Class or Model
implementation. This can be used to inherit the implementation
from a non-SMP C++ class into an SMP Class or Model
implementation, for example to wrap an existing C++
implementation as SMP Class or Model. The default value for
this attribute is the empty string, which corresponds to no
base class.</Description>
            <Type xlink:title="Bool" xlink:href="#Smp.Bool"/>
            <Default xsi:type="Types:BoolValue" Value="true"/>
            <Usage>Class</Usage>
            <Usage>Exception</Usage>
            <Usage>Model</Usage>
            <Usage>Service</Usage>
         </Type>
         <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.ByReference"
               Name="ByReference"
               Visibility="public"
               Uuid="8596d697-fb84-41ce-a685-6912006ed6c3">
            <Description>The ByReference attribute specifies
that a parameter is passed by reference, i.e. as pointer in
C++. The default value for this attribute is
false.</Description>
            <Type xlink:title="Bool" xlink:href="#Smp.Bool"/>
            <Default xsi:type="Types:BoolValue" Value="true"/>
            <Usage>Parameter</Usage>
         </Type>
         <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.Const" Name="Const"
               Visibility="public"
               Uuid="8596d697-fb84-41ce-a685-6912006ed6c4">
            <Description>The Const attribute specifies that a
feature is "constant" in the following sense:
   &lt;ul&gt;
   &lt;li&gt;Association: The value of the referenced element
is constant, i.e. it cannot be changed during
runtime.&lt;/li&gt;
   &lt;li&gt;Operation: The state of the containing type (e.g.
Model) is constant, i.e. the operation must not change any
field values during execution.&lt;/li&gt;
   &lt;li&gt;Property: The state of the containing type (e.g.
Model) is constant, i.e. the property getter must not change
any field values during execution.&lt;/li&gt;
   &lt;li&gt;Parameter: The value of the parameter is
constant, i.e. the operation must not change it during
execution. When applied to reference parameters (pointers in
C++), the referenced element must not be changed.&lt;/li&gt;
   &lt;/ul&gt;
   The default value for this attribute is
false.</Description>
            <Type xlink:title="Bool" xlink:href="#Smp.Bool"/>
            <Default xsi:type="Types:BoolValue" Value="true"/>
            <Usage>Association</Usage>
            <Usage>Property</Usage>
            <Usage>Operation</Usage>
            <Usage>Parameter</Usage>
         </Type>
         <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.Constructor"
               Name="Constructor"
               Visibility="public"
               Uuid="8596d697-fb84-41ce-a685-6912006ed6c5">
            <Description>The Constructor attribute specifies
that the operation is mapped to a C++ constructor. The default
value for this attribute is false, which corresponds to not
mapping to a constructor.
   A constructor must not have a return parameter.
   The name of the constructor is ignored as the Class or
Model name is used in C++.</Description>
            <Type xlink:title="Bool" xlink:href="#Smp.Bool"/>
            <Default xsi:type="Types:BoolValue" Value="true"/>
            <Usage>Operation</Usage>
         </Type>
         <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.Operator" Name="Operator"
               Visibility="public"
               Uuid="8596d697-fb84-41ce-a685-6912006ed6c6">
            <Description>The Operator attribute defines an
operator kind for an operation. It can be used to specify that
the operation is mapped to a C++ operator. The default value
for this attribute is None, which corresponds to not mapping
to an operator.</Description>
            <Type xlink:title="OperatorKind"
xlink:href="#Smp.Attributes.OperatorKind"/>
            <Default xsi:type="Types:Int32Value" Value="0"/>
            <Usage>Operation</Usage>
         </Type>
         <Type xsi:type="Types:Enumeration"
Id="Smp.Attributes.OperatorKind"
               Name="OperatorKind"
               Visibility="public"
               Uuid="d5562bc8-e618-11dc-ab64-bf8df6d7b83a">
            <Description>Enumeration of available operator
kinds</Description>
            <Literal Id="Smp.Attributes.OperatorKind.None"
Name="None" Value="0">
               <Description>No operator</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Positive"
Name="Positive" Value="1">
               <Description>Positive value</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Negative"
Name="Negative" Value="2">
               <Description>Negative value</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Assign"
Name="Assign" Value="3">
               <Description>Assign new value</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Add"
Name="Add" Value="4">
               <Description>Add value</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Subtract"
Name="Subtract" Value="5">
               <Description>Subtract value</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Multiply"
Name="Multiply" Value="6">
               <Description>Multiply with value</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Divide"
Name="Divide" Value="7">
               <Description>Divide by value</Description>
            </Literal>
            <Literal
Id="Smp.Attributes.OperatorKind.Remainder" Name="Remainder"
Value="8">
               <Description>Remainder of
division</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Greater"
Name="Greater" Value="9">
               <Description>Is greater than</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Less"
Name="Less" Value="10">
               <Description>Is less than</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Equal"
Name="Equal" Value="11">
               <Description>Is equal</Description>
            </Literal>
            <Literal
Id="Smp.Attributes.OperatorKind.NotGreater" Name="NotGreater"
Value="12">
               <Description>Is not greater than</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.NotLess"
Name="NotLess" Value="13">
               <Description>Is not less than</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.NotEqual"
Name="NotEqual" Value="14">
               <Description>Is not equal</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Indexer"
Name="Indexer" Value="15">
               <Description>Index into array</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Sum"
Name="Sum" Value="16">
               <Description>Sum of instance and another
value</Description>
            </Literal>
            <Literal
Id="Smp.Attributes.OperatorKind.Difference" Name="Difference"
Value="17">
               <Description>Difference between instance and
another value</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Product"
Name="Product" Value="18">
               <Description>Product of instance and another
value</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Quotient"
Name="Quotient" Value="19">
               <Description>Quotient of instance and another
value</Description>
            </Literal>
            <Literal Id="Smp.Attributes.OperatorKind.Module"
Name="Module" Value="20">
               <Description>Remainder of instance divided by
another value</Description>
            </Literal>
         </Type>
         <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.Static" Name="Static"
               Visibility="public"
               Uuid="8596d697-fb84-41ce-a685-6912006ed6c7">
            <Description>The Static attribute specifies that a
feature is static, i.e. that it is defined on type/classifier
scope. The default value for this attribute is false, which
corresponds to instance scope.</Description>
            <Type xlink:title="Bool" xlink:href="#Smp.Bool"/>
            <Default xsi:type="Types:BoolValue" Value="true"/>
            <Usage>Operation</Usage>
            <Usage>Property</Usage>
            <Usage>Field</Usage>
            <Usage>Association</Usage>
         </Type>
         <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.Virtual" Name="Virtual"
               Visibility="public"
               Uuid="8596d697-fb84-41ce-a685-6912006ed6c8">
            <Description>The Virtual attribute specifies that
an operation or property is virtual, i.e. that it may be
overridden in a derived type (polymorphism). The default value
for this attribute is false.</Description>
            <Type xlink:title="Bool" xlink:href="#Smp.Bool"/>
            <Default xsi:type="Types:BoolValue" Value="true"/>
            <Usage>Operation</Usage>
            <Usage>Property</Usage>
         </Type>
         <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.Minimum" Name="Minimum"
               Visibility="public"
               Uuid="8596d697-fb84-41ce-a685-6912006ed662">
            <Description>The Minimum attribute type defines a
duration for an Entry Point specifying the minimum duration
between two consecutive calls of the Entry Point. The default
```

```
value for this attribute is 0, which corresponds to no minimum
value.</Description>
            <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
            <Default xsi:type="Types:DurationValue"
Value="PT0S"/>
            <Usage>EntryPoint</Usage>
         </Type>
         <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.Maximum" Name="Maximum"
               Visibility="public"
               Uuid="8596d697-fb84-41ce-a685-6912006ed663">
            <Description>The Maximum attribute type defines a
duration for an Entry Point specifying the maximum duration
between two consecutive calls of the Entry Point. The default
value for this attribute is -1, which corresponds to no
maximum value.</Description>
            <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
            <Default xsi:type="Types:DurationValue"
Value="PT0S"/>
            <Usage>EntryPoint</Usage>
         </Type>
         <Type xsi:type="Types:AttributeType"
Id="Smp.Attributes.View" Name="View"
               Visibility="public"
               Uuid="8596d697-fb84-41ce-a685-6912006ed664">
            <Description>The View attribute can be applied to
a Field, Property, Operation or Entry Point to specify the
element's visibility during publication.</Description>
            <Type xlink:title="ViewKind"
xlink:href="#_12_1_f810373_1204186837776_552448_5139"/>
            <Default xsi:type="Types:Int32Value" Value="0"/>
            <Usage>Field</Usage>
            <Usage>Property</Usage>
            <Usage>Operation</Usage>
            <Usage>EntrryPoint</Usage>
         </Type>
      </Namespace>
      <Namespace Id="Smp.Management" Name="Management">
         <Description>Namespace for managed component
extensions</Description>
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Management.IManagedObject"
               Name="IManagedObject"
               Visibility="public"
               Uuid="d539a37f-e618-11dc-ab64-bf8df6d7b83a">
            <Description>Interface of a managed
object.</Description>
            <NestedType xsi:type="Types:Exception"
Id="Smp.Management.IManagedObject.InvalidObjectName"
```

```
                        Name="InvalidObjectName"
                        Visibility="public"
                        Uuid="d539a38f-e618-11dc-ab64-
bf8df6d7b83a">
               <Description>This exception is raised when
trying to set an object's name to an invalid name. Names
&lt;ul&gt;
&lt;li&gt;must not be empty,&lt;/li&gt;
&lt;li&gt;must start with a letter, and&lt;/li&gt;
&lt;li&gt;must only contain letters, digits, the underscore
("_") and brackets ("[" and "]").&lt;/li&gt;
&lt;/ul&gt;</Description>
               <Field
Id="Smp.Management.IManagedObject.InvalidObjectName.objectName
"
                      Name="objectName"
                      Visibility="public">
                  <Description>Invalid object name passed to
SetName().</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Field>
               <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
            </NestedType>
            <Operation
Id="Smp.Management.IManagedObject.SetName" Name="SetName"
Visibility="public">
               <Description>Define the name of the managed
object ("property setter").</Description>
               <Parameter
Id="Smp.Management.IManagedObject.SetName.name" Name="name">
                  <Description>Name of object.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Parameter>
               <RaisedException
xlink:title="InvalidObjectName"

xlink:href="#Smp.Management.IManagedObject.InvalidObjectName"/
>
            </Operation>
            <Operation
Id="Smp.Management.IManagedObject.SetDescription"
Name="SetDescription"
                       Visibility="public">
               <Description>Define the description of the
managed object ("property setter").</Description>
               <Parameter
Id="Smp.Management.IManagedObject.SetDescription.description"
                          Name="description">
                  <Description>Description of
object.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Parameter>
            </Operation>
            <Base xlink:title="IObject"
xlink:href="#Smp.IObject"/>
         </Type>
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Management.IManagedSimulator"
               Name="IManagedSimulator"
               Visibility="public"
               Uuid="d540cf27-e618-11dc-ab64-bf8df6d7b83a">
            <Description>This interface gives access to a
managed simulator.</Description>
            <NestedType xsi:type="Types:Exception"
Id="Smp.Management.IManagedSimulator.DuplicateUuid"
                        Name="DuplicateUuid"
                        Visibility="public"
                        Uuid="d540cf46-e618-11dc-ab64-
bf8df6d7b83a">
               <Description>This exception is raised when
trying to register a factory under a Uuid that has already
been used to register another (or the same) factory.This would
lead to duplicate implementation Uuids.</Description>
               <Field
Id="Smp.Management.IManagedSimulator.DuplicateUuid.newName"
Name="newName"
                      Visibility="public">
                  <Description>Name of factory that tried to
register under this Uuid.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Field>
               <Field
Id="Smp.Management.IManagedSimulator.DuplicateUuid.oldName"
Name="oldName"
                      Visibility="public">
                  <Description>Name of factory already
registered under this Uuid.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Field>
               <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
            </NestedType>
            <Operation
Id="Smp.Management.IManagedSimulator.RegisterFactory"
Name="RegisterFactory"
                       Visibility="public">
               <Description>This method registers a component
factory with the managed simulator. The managed simulator can
use this factory to create component instances of the
component implementation in its CreateInstance()
method.</Description>
               <Parameter
Id="Smp.Management.IManagedSimulator.RegisterFactory.component
Factory"
                          Name="componentFactory">
                  <Description>Factory to create instance of
the component implementation.</Description>
                  <Type xlink:title="IFactory"
xlink:href="#Smp.IFactory"/>
               </Parameter>
               <RaisedException xlink:title="DuplicateUuid"

xlink:href="#Smp.Management.IManagedSimulator.DuplicateUuid"/>
            </Operation>
            <Operation
Id="Smp.Management.IManagedSimulator.CreateInstance"
Name="CreateInstance"
                       Visibility="public">
               <Description>This method creates an instance of
the component with the given implementation
identifier.</Description>
               <Parameter
Id="Smp.Management.IManagedSimulator.CreateInstance.return"
Name="return"
                          Direction="return">
                  <Description>New instance of the component
with the given implementation identifier or null in case no
factory for the given implementation identifier has been
registered.</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
               <Parameter
Id="Smp.Management.IManagedSimulator.CreateInstance.implUuid"
Name="implUuid">
                  <Description>Implementation identifier of
the component.</Description>
                  <Type xlink:title="Uuid"
xlink:href="#Smp.Uuid"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Management.IManagedSimulator.GetFactory"
Name="GetFactory"
                       Visibility="public">
               <Description>This method returns the factory of
the component with the given implementation
identifier.</Description>
               <Parameter
Id="Smp.Management.IManagedSimulator.GetFactory.return"
Name="return"
                          Direction="return">
                  <Description>Factory of the component with
the given implementation identifier or null in case no factory
for the given implementation identifier has been
registered.</Description>
                  <Type xlink:title="IFactory"
xlink:href="#Smp.IFactory"/>
               </Parameter>
               <Parameter
Id="Smp.Management.IManagedSimulator.GetFactory.implUuid"
Name="implUuid">
                  <Description>Implementation identifier of
the component.</Description>
                  <Type xlink:title="Uuid"
xlink:href="#Smp.Uuid"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Management.IManagedSimulator.GetFactories"
Name="GetFactories"
                       Visibility="public">
               <Description>This method returns all factories
of components with the given specification
identifier.</Description>
               <Parameter
Id="Smp.Management.IManagedSimulator.GetFactories.return"
Name="return"
                          Direction="return">
                  <Description>Collection of factories for the
given specification identifier.</Description>
                  <Type xlink:title="FactoryCollection"
xlink:href="#Smp.FactoryCollection"/>
               </Parameter>
               <Parameter
Id="Smp.Management.IManagedSimulator.GetFactories.specUuid"
Name="specUuid">
                  <Description>Specification identifier of the
component.</Description>
                  <Type xlink:title="Uuid"
xlink:href="#Smp.Uuid"/>
               </Parameter>
            </Operation>
            <Base xlink:title="ISimulator"
xlink:href="#Smp.ISimulator"/>
         </Type>
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Management.IManagedComponent"
               Name="IManagedComponent"
               Visibility="public"
               Uuid="d539a394-e618-11dc-ab64-bf8df6d7b83a">
            <Description>Interface of a managed
component.</Description>
            <Operation
Id="Smp.Management.IManagedComponent.SetParent"
Name="SetParent"
                       Visibility="public">
               <Description>Define the parent component
("property setter"). Components link to their parent to allow
traversing the tree of components upwards.</Description>
               <Parameter
Id="Smp.Management.IManagedComponent.SetParent.parent"
Name="parent">
                  <Description>Parent composite of
component.</Description>
                  <Type xlink:title="IComposite"
xlink:href="#Smp.IComposite"/>
               </Parameter>
            </Operation>
            <Base xlink:title="IManagedObject"
xlink:href="#Smp.Management.IManagedObject"/>
            <Base xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
         </Type>
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Management.IComponentCollection"
               Name="IComponentCollection"
               Visibility="public"
               Uuid="d540cf17-e618-11dc-ab64-bf8df6d7b83a">
            <Description>Base interface for managed
collections, which are either references or
containers.</Description>
            <Operation
Id="Smp.Management.IComponentCollection.GetCount"
Name="GetCount"
                       Visibility="public">
               <Description>Query for the number of components
in the collection.</Description>
               <Parameter
Id="Smp.Management.IComponentCollection.GetCount.return"
Name="return"
                          Direction="return">
                  <Description>Current number of components in
the collection.</Description>
                  <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Management.IComponentCollection.GetUpper"
Name="GetUpper"
                       Visibility="public">
               <Description>Query the maximum number of
components in the collection.</Description>
               <Parameter
Id="Smp.Management.IComponentCollection.GetUpper.return"
Name="return"
                          Direction="return">
                  <Description>Maximum number of components in
the collection. (-1 = unlimited).</Description>
                  <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Management.IComponentCollection.GetLower"
Name="GetLower"
                       Visibility="public">
               <Description>Query the minimum number of
components in the collection.</Description>
               <Parameter
Id="Smp.Management.IComponentCollection.GetLower.return"
Name="return"
                          Direction="return">
                  <Description>Minimum number of components in
the collection.</Description>
                  <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
               </Parameter>
            </Operation>
         </Type>
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Management.IManagedReference"
               Name="IManagedReference"
               Visibility="public"
               Uuid="d53c13cb-e618-11dc-ab64-bf8df6d7b83a">
            <Description>Interface of a managed
reference.</Description>
            <NestedType xsi:type="Types:Exception"
Id="Smp.Management.IManagedReference.ReferenceFull"
                        Name="ReferenceFull"
                        Visibility="public"
                        Uuid="d53c13dc-e618-11dc-ab64-
bf8df6d7b83a">
               <Description>This exception is raised when
trying to add a component to a reference that is full, i.e.
where the Count has reached the Upper limit.</Description>
```

```
               <Field
Id="Smp.Management.IManagedReference.ReferenceFull.referenceNa
me"
                      Name="referenceName"
                      Visibility="public">
                  <Description>Name of
reference.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Field>
               <Field
Id="Smp.Management.IManagedReference.ReferenceFull.referenceSi
ze"
                      Name="referenceSize"
                      Visibility="public">
                  <Description>Number of components in the
reference, which is its Upper limit when the reference is
full.</Description>
                  <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
               </Field>
               <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
            </NestedType>
            <NestedType xsi:type="Types:Exception"
Id="Smp.Management.IManagedReference.NotReferenced"
                        Name="NotReferenced"
                        Visibility="public"
                        Uuid="d53e5dd3-e618-11dc-ab64-
bf8df6d7b83a">
               <Description>This exception is thrown when
trying to remove a component from a reference which was not
referenced before.</Description>
               <Field
Id="Smp.Management.IManagedReference.NotReferenced.referenceNa
me"
                      Name="referenceName"
                      Visibility="public">
                  <Description>Name of
reference.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Field>
               <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
               <Association
Id="Smp.Management.IManagedReference.NotReferenced.component"
Name="component"
                            Visibility="public">
                  <Description>Component that is not
referenced.</Description>
```

```
                    <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
                </Association>
            </NestedType>
            <NestedType xsi:type="Types:Exception"
Id="Smp.Management.IManagedReference.CannotRemove"
                        Name="CannotRemove"
                        Visibility="public"
                        Uuid="d53e5dda-e618-11dc-ab64-
bf8df6d7b83a">
                <Description>This exception is thrown when
trying to remove a component from a reference when the number
of referenced components is lower than or equal to the Lower
limit.</Description>
                <Field
Id="Smp.Management.IManagedReference.CannotRemove.referenceNam
e"
                      Name="referenceName"
                      Visibility="public">
                    <Description>Name of
reference.</Description>
                    <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
                </Field>
                <Field
Id="Smp.Management.IManagedReference.CannotRemove.lowerLimit"
Name="lowerLimit"
                      Visibility="public">
                    <Description>Lower limit of the
reference.</Description>
                    <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
                </Field>
                <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
                <Association
Id="Smp.Management.IManagedReference.CannotRemove.component"
Name="component"
                            Visibility="public">
                    <Description>Component that could not be
removed.</Description>
                    <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
                </Association>
            </NestedType>
            <Operation
Id="Smp.Management.IManagedReference.AddComponent"
Name="AddComponent"
                       Visibility="public">
                <Description>Add a referenced
component.</Description>
```

```
               <Parameter
Id="Smp.Management.IManagedReference.AddComponent.component"
Name="component">
                  <Description>New referenced
component.</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
               <RaisedException xlink:title="ReferenceFull"

xlink:href="#Smp.Management.IManagedReference.ReferenceFull"/>
               <RaisedException
xlink:title="InvalidObjectType"
xlink:href="#Smp.InvalidObjectType"/>
            </Operation>
            <Operation
Id="Smp.Management.IManagedReference.RemoveComponent"
Name="RemoveComponent"
                       Visibility="public">
               <Description>Remove a referenced
component.</Description>
               <Parameter
Id="Smp.Management.IManagedReference.RemoveComponent.component
"
                          Name="component">
                  <Description>Referenced component to
remove.</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
               <RaisedException xlink:title="NotReferenced"

xlink:href="#Smp.Management.IManagedReference.NotReferenced"/>
               <RaisedException xlink:title="CannotRemove"

xlink:href="#Smp.Management.IManagedReference.CannotRemove"/>
            </Operation>
            <Base xlink:title="IComponentCollection"

xlink:href="#Smp.Management.IComponentCollection"/>
            <Base xlink:title="IReference"
xlink:href="#Smp.IReference"/>
         </Type>
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Management.IManagedContainer"
               Name="IManagedContainer"
               Visibility="public"
               Uuid="d53c13b9-e618-11dc-ab64-bf8df6d7b83a">
            <Description>Interface of a managed
container.</Description>
```

```
            <NestedType xsi:type="Types:Exception"
Id="Smp.Management.IManagedContainer.ContainerFull"
                        Name="ContainerFull"
                        Visibility="public"
                        Uuid="d53c13c4-e618-11dc-ab64-
bf8df6d7b83a">
               <Description>This exception is raised when
trying to add a component to a container that is full, i.e.
where the Count has reached the Upper limit.</Description>
               <Field
Id="Smp.Management.IManagedContainer.ContainerFull.containerNa
me"
                      Name="containerName"
                      Visibility="public">
                  <Description>Name of full
container.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Field>
               <Field
Id="Smp.Management.IManagedContainer.ContainerFull.containerSi
ze"
                      Name="containerSize"
                      Visibility="public">
                  <Description>Number of components in the
container, which is its Upper limit when the container is
full.</Description>
                  <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
               </Field>
               <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
            </NestedType>
            <Operation
Id="Smp.Management.IManagedContainer.AddComponent"
Name="AddComponent"
                       Visibility="public">
               <Description>Add a contained component to the
container.</Description>
               <Parameter
Id="Smp.Management.IManagedContainer.AddComponent.component"
Name="component">
                  <Description>New contained
component.</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
               <RaisedException xlink:title="ContainerFull"

xlink:href="#Smp.Management.IManagedContainer.ContainerFull"/>
```

```
               <RaisedException xlink:title="DuplicateName"
xlink:href="#Smp.DuplicateName"/>
               <RaisedException
xlink:title="InvalidObjectType"
xlink:href="#Smp.InvalidObjectType"/>
            </Operation>
            <Base xlink:title="IComponentCollection"

xlink:href="#Smp.Management.IComponentCollection"/>
            <Base xlink:title="IContainer"
xlink:href="#Smp.IContainer"/>
         </Type>
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Management.IEventConsumer"
               Name="IEventConsumer"
               Visibility="public"
               Uuid="d53e5df4-e618-11dc-ab64-bf8df6d7b83a">
            <Description>Interface of an event
consumer.</Description>
            <Operation
Id="Smp.Management.IEventConsumer.GetEventSinks"
Name="GetEventSinks"
                       Visibility="public">
               <Description>Query for the collection of all
event sinks of the component.</Description>
               <Parameter
Id="Smp.Management.IEventConsumer.GetEventSinks.return"
Name="return"
                          Direction="return">
                  <Description>Collection of event
sinks.</Description>
                  <Type xlink:title="EventSinkCollection"
xlink:href="#Smp.EventSinkCollection"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Management.IEventConsumer.GetEventSink"
Name="GetEventSink"
                       Visibility="public">
               <Description>Query for an event sink of this
component by its name.</Description>
               <Parameter
Id="Smp.Management.IEventConsumer.GetEventSink.return"
Name="return"
                          Direction="return">
                  <Description>Event sink with the given name,
or null if no event sink with the given name could be
found.</Description>
                  <Type xlink:title="IEventSink"
xlink:href="#Smp.IEventSink"/>
               </Parameter>
               <Parameter
Id="Smp.Management.IEventConsumer.GetEventSink.name"
Name="name">
                  <Description>Event sink name.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Parameter>
            </Operation>
            <Base xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
         </Type>
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Management.IEventProvider"
               Name="IEventProvider"
               Visibility="public"
               Uuid="d53e5de3-e618-11dc-ab64-bf8df6d7b83a">
            <Description>Interface of an event
provider.</Description>
            <Operation
Id="Smp.Management.IEventProvider.GetEventSource"
Name="GetEventSource"
                       Visibility="public">
               <Description>Query for an event source of this
component by its name.</Description>
               <Parameter
Id="Smp.Management.IEventProvider.GetEventSource.return"
Name="return"
                          Direction="return">
                  <Description>Event source with the given
name or null if no event source with the given name could be
found.</Description>
                  <Type xlink:title="IEventSource"
xlink:href="#Smp.IEventSource"/>
               </Parameter>
               <Parameter
Id="Smp.Management.IEventProvider.GetEventSource.name"
Name="name">
                  <Description>Event source
name.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Management.IEventProvider.GetEventSources"
Name="GetEventSources"
                       Visibility="public">
               <Description>Query for the collection of all
event sources of the component.</Description>
```

```
               <Parameter
Id="Smp.Management.IEventProvider.GetEventSources.return"
Name="return"
                          Direction="return">
                  <Description>Collection of event
sources.</Description>
                  <Type xlink:title="EventSourceCollection"
xlink:href="#Smp.EventSourceCollection"/>
               </Parameter>
            </Operation>
            <Base xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
         </Type>
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Management.IManagedModel"
               Name="IManagedModel"
               Visibility="public"
               Uuid="d539a39d-e618-11dc-ab64-bf8df6d7b83a">
            <Description>Interface of a managed
model.</Description>
            <NestedType xsi:type="Types:Exception"
Id="Smp.Management.IManagedModel.InvalidFieldName"
                        Name="InvalidFieldName"
                        Visibility="public"
                        Uuid="d53c13b4-e618-11dc-ab64-
bf8df6d7b83a">
               <Description>This exception is raised when an
invalid field name is specified.</Description>
               <Field
Id="Smp.Management.IManagedModel.InvalidFieldName.fieldName"
Name="fieldName"
                      Visibility="public">
                  <Description>Fully qualified field name that
is invalid.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Field>
               <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
            </NestedType>
            <Operation
Id="Smp.Management.IManagedModel.GetSimpleField"
Name="GetSimpleField"
                       Visibility="public">
               <Description>Get the field of given name that
is typed by a simple type.
</Description>
               <Parameter
Id="Smp.Management.IManagedModel.GetSimpleField.return"
Name="return"
                          Direction="return">
```

```
                            <Description>Simple field.</Description>
                            <Type xlink:title="ISimpleField"
xlink:href="#Smp.ISimpleField"/>
                         </Parameter>
                         <Parameter
Id="Smp.Management.IManagedModel.GetSimpleField.fullName"
Name="fullName">
                            <Description>Fully qualified field name
(relative to the model).</Description>
                            <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
                         </Parameter>
                         <RaisedException xlink:title="InvalidFieldName"

xlink:href="#Smp.Management.IManagedModel.InvalidFieldName"/>
                      </Operation>
                      <Operation
Id="Smp.Management.IManagedModel.GetArrayField"
Name="GetArrayField"
                                Visibility="public">
                         <Description>Get the field of given name that
is an array of a simple type.</Description>
                         <Parameter
Id="Smp.Management.IManagedModel.GetArrayField.fullName"
Name="fullName">
                            <Description>Fully qualified array field
name (relative to the model)</Description>
                            <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
                         </Parameter>
                         <Parameter
Id="Smp.Management.IManagedModel.GetArrayField.return"
Name="return"
                                   Direction="return">
                            <Description>Array field.</Description>
                            <Type xlink:title="IArrayField"
xlink:href="#Smp.IArrayField"/>
                         </Parameter>
                         <RaisedException xlink:title="InvalidFieldName"

xlink:href="#Smp.Management.IManagedModel.InvalidFieldName"/>
                      </Operation>
                      <Base xlink:title="IManagedComponent"
xlink:href="#Smp.Management.IManagedComponent"/>
                      <Base xlink:title="IModel"
xlink:href="#Smp.IModel"/>
                   </Type>
                   <Type xsi:type="Catalogue:Interface"
Id="Smp.Management.IEntryPointPublisher"
                         Name="IEntryPointPublisher"
                         Visibility="public"
```

```
                        Uuid="d53e5e05-e618-11dc-ab64-bf8df6d7b83a">
            <Description>Interface of an entry point
publisher.</Description>
            <Operation
Id="Smp.Management.IEntryPointPublisher.GetEntryPoints"
Name="GetEntryPoints"
                        Visibility="public">
                <Description>Query for the collection of all
entry points of the model.</Description>
                <Parameter
Id="Smp.Management.IEntryPointPublisher.GetEntryPoints.return"
Name="return"
                           Direction="return">
                    <Description>Collection of entry
points.</Description>
                    <Type xlink:title="EntryPointCollection"
xlink:href="#Smp.EntryPointCollection"/>
                </Parameter>
            </Operation>
            <Operation
Id="Smp.Management.IEntryPointPublisher.GetEntryPoint"
Name="GetEntryPoint"
                        Visibility="public">
                <Description>Query for an entry point of this
model by its name.</Description>
                <Parameter
Id="Smp.Management.IEntryPointPublisher.GetEntryPoint.return"
Name="return"
                           Direction="return">
                    <Description>Entry point with given name, or
null if no entry point with given name could be
found.</Description>
                    <Type xlink:title="IEntryPoint"
xlink:href="#Smp.IEntryPoint"/>
                </Parameter>
                <Parameter
Id="Smp.Management.IEntryPointPublisher.GetEntryPoint.name"
Name="name">
                    <Description>Entry point name.</Description>
                    <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
                </Parameter>
            </Operation>
            <Base xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
         </Type>
      </Namespace>
      <Namespace Id="Smp.Services" Name="Services">
         <Description>Namespace for simulation
services</Description>
```

```
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Services.ILinkRegistry"
               Name="ILinkRegistry"
               Visibility="public"
               Uuid="d54f26d7-e618-11dc-ab64-bf8df6d7b83a">
            <Description>This interface is implemented by the
Link Registry Service.</Description>
            <Constant
Id="Smp.Services.ILinkRegistry.SMP_LinkRegistry"
Name="SMP_LinkRegistry"
                      Visibility="public">
               <Description>Name of the LinkRegistry
service.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="LinkRegistry"/>
            </Constant>
            <Operation Id="Smp.Services.ILinkRegistry.AddLink"
Name="AddLink" Visibility="public">
               <Description>Add a link from source component
to target component.</Description>
               <Parameter
Id="Smp.Services.ILinkRegistry.AddLink.source" Name="source">
                  <Description>Source component of link (i.e.
the component that links to another component).</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
               <Parameter
Id="Smp.Services.ILinkRegistry.AddLink.target" Name="target">
                  <Description>Target component of link (i.e.
the component that is being linked to by another
component).</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
            </Operation>
            <Operation Id="Smp.Services.ILinkRegistry.HasLink"
Name="HasLink" Visibility="public">
               <Description>Returns true if a link between
source and target exists, false otherwise.</Description>
               <Parameter
Id="Smp.Services.ILinkRegistry.HasLink.return" Name="return"
Direction="return">
                  <Description>True if such a link has been
added before (and not been removed), false
otherwise.</Description>
                  <Type xlink:title="Bool"
xlink:href="#Smp.Bool"/>
               </Parameter>
               <Parameter
Id="Smp.Services.ILinkRegistry.HasLink.source" Name="source">
                  <Description>Source component of link  (i.e.
the component that links to another component).</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
               <Parameter
Id="Smp.Services.ILinkRegistry.HasLink.target" Name="target">
                  <Description>Target component of link  (i.e.
the component that is being linked to by another
component).</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Services.ILinkRegistry.RemoveLink" Name="RemoveLink"
Visibility="public">
               <Description>Remove a link between source and
target that has been added to the service using AddLink()
before.</Description>
               <Parameter
Id="Smp.Services.ILinkRegistry.RemoveLink.source"
Name="source">
                  <Description>Source component of link (i.e.
the component that links to another component).</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
               <Parameter
Id="Smp.Services.ILinkRegistry.RemoveLink.target"
Name="target">
                  <Description>Target component of link (i.e.
the component that is being linked to by another
component).</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Services.ILinkRegistry.GetLinks" Name="GetLinks"
Visibility="public">
               <Description>Returns a collection of all
sources that have a link to the given target.</Description>
               <Parameter
Id="Smp.Services.ILinkRegistry.GetLinks.return" Name="return"
Direction="return">
                  <Description>Collection of source components
which link to the given target.</Description>
```

```
                  <Type xlink:title="ComponentCollection"
xlink:href="#Smp.ComponentCollection"/>
               </Parameter>
               <Parameter
Id="Smp.Services.ILinkRegistry.GetLinks.target" Name="target">
                  <Description>Target component to returns
links for.</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Services.ILinkRegistry.CanRemove" Name="CanRemove"
Visibility="public">
               <Description>Returns true if all sources
linking to the given target can be asked to remove their
link(s), false otherwise.</Description>
               <Parameter
Id="Smp.Services.ILinkRegistry.CanRemove.return" Name="return"
                          Direction="return">
                  <Description>True if all links to the given
target can be removed, false otherwise.</Description>
                  <Type xlink:title="Bool"
xlink:href="#Smp.Bool"/>
               </Parameter>
               <Parameter
Id="Smp.Services.ILinkRegistry.CanRemove.target"
Name="target">
                  <Description>Target component to check for
links.</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Services.ILinkRegistry.RemoveLinks" Name="RemoveLinks"
                       Visibility="public">
               <Description>Removes all links to the given
target.</Description>
               <Parameter
Id="Smp.Services.ILinkRegistry.RemoveLinks.target"
Name="target">
                  <Description>Target component of link (i.e.
the component that is being linked to by other
components).</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
            </Operation>
            <Base xlink:title="IService"
xlink:href="#Smp.IService"/>
```

```
         </Type>
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Services.ILogger" Name="ILogger"
               Visibility="public"
               Uuid="d5434051-e618-11dc-ab64-bf8df6d7b83a">
            <Description>This interface gives access to the
Logger Service.</Description>
            <Constant
Id="Smp.Services.ILogger.LMK_Information"
Name="LMK_Information"
                      Visibility="public">
               <Description>The message contains general
information.</Description>
               <Type xlink:title="LogMessageKind"
xlink:href="#Smp.Services.LogMessageKind"/>
               <Value xsi:type="Types:Int64Value" Value="0"/>
            </Constant>
            <Constant Id="Smp.Services.ILogger.LMK_Event"
Name="LMK_Event" Visibility="public">
               <Description>The message has been sent from an
event, typically from a state transition.</Description>
               <Type xlink:title="LogMessageKind"
xlink:href="#Smp.Services.LogMessageKind"/>
               <Value xsi:type="Types:Int64Value" Value="1"/>
            </Constant>
            <Constant Id="Smp.Services.ILogger.LMK_Warning"
Name="LMK_Warning" Visibility="public">
               <Description>The message contains a
warning.</Description>
               <Type xlink:title="LogMessageKind"
xlink:href="#Smp.Services.LogMessageKind"/>
               <Value xsi:type="Types:Int64Value" Value="2"/>
            </Constant>
            <Constant Id="Smp.Services.ILogger.LMK_Error"
Name="LMK_Error" Visibility="public">
               <Description>The message has been raised
because of an error.</Description>
               <Type xlink:title="LogMessageKind"
xlink:href="#Smp.Services.LogMessageKind"/>
               <Value xsi:type="Types:Int64Value" Value="3"/>
            </Constant>
            <Constant Id="Smp.Services.ILogger.LMK_Debug"
Name="LMK_Debug" Visibility="public">
               <Description>The message contains debug
information.</Description>
               <Type xlink:title="LogMessageKind"
xlink:href="#Smp.Services.LogMessageKind"/>
               <Value xsi:type="Types:Int64Value" Value="4"/>
            </Constant>
            <Constant
Id="Smp.Services.ILogger.LMK_InformationName"
Name="LMK_InformationName"
                      Visibility="public">
               <Description>The message contains general
information.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="Information"/>
            </Constant>
            <Constant Id="Smp.Services.ILogger.LMK_DebugName"
Name="LMK_DebugName"
                      Visibility="public">
               <Description>The message contains debug
information.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="Debug"/>
            </Constant>
            <Constant Id="Smp.Services.ILogger.LMK_ErrorName"
Name="LMK_ErrorName"
                      Visibility="public">
               <Description>The message has been raised
because of an error.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="Error"/>
            </Constant>
            <Constant
Id="Smp.Services.ILogger.LMK_WarningName"
Name="LMK_WarningName"
                      Visibility="public">
               <Description>The message contains a
warning.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="Warning"/>
            </Constant>
            <Constant Id="Smp.Services.ILogger.LMK_EventName"
Name="LMK_EventName"
                      Visibility="public">
               <Description>The message has been sent from an
event, typically from a state transition.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="Event"/>
```

```
            </Constant>
            <Constant Id="Smp.Services.ILogger.SMP_Logger"
Name="SMP_Logger" Visibility="public">
               <Description>Name of the logger
service.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="Logger"/>
            </Constant>
            <Operation
Id="Smp.Services.ILogger.QueryLogMessageKind"
Name="QueryLogMessageKind"
                       Visibility="public">
               <Description>Return identifier of log message
kind by name.
</Description>
               <Parameter
Id="Smp.Services.ILogger.QueryLogMessageKind.return"
Name="return"
                          Direction="return">
                  <Description>Identifier of log message
kind.</Description>
                  <Type xlink:title="LogMessageKind"
xlink:href="#Smp.Services.LogMessageKind"/>
               </Parameter>
               <Parameter
Id="Smp.Services.ILogger.QueryLogMessageKind.messageKindName"
                          Name="messageKindName">
                  <Description>Name of log message
kind.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Parameter>
            </Operation>
            <Operation Id="Smp.Services.ILogger.Log"
Name="Log" Visibility="public">
               <Description>This function logs a message to
the simulation log.</Description>
               <Parameter Id="Smp.Services.ILogger.Log.sender"
Name="sender">
                  <Description>Object that sends the
message.</Description>
                  <Type xlink:title="IObject"
xlink:href="#Smp.IObject"/>
               </Parameter>
               <Parameter
Id="Smp.Services.ILogger.Log.message" Name="message">
                  <Description>The message to
log.</Description>
```

```
                    <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
                 </Parameter>
                 <Parameter Id="Smp.Services.ILogger.Log.kind"
Name="kind">
                    <Description>Kind of message.</Description>
                    <Type xlink:title="LogMessageKind"
xlink:href="#Smp.Services.LogMessageKind"/>
                    <Default xsi:type="Types:Int64Value"
Value="0"/>
                 </Parameter>
              </Operation>
              <Base xlink:title="IService"
xlink:href="#Smp.IService"/>
           </Type>
           <Type xsi:type="Types:Integer"
Id="Smp.Services.LogMessageKind" Name="LogMessageKind"
                 Visibility="public"
                 Uuid="d543404f-e618-11dc-ab64-bf8df6d7b83a"
                 Minimum="0"
                 Maximum="2147483647">
              <Description>This type is used as identifier of a
log message kind.</Description>
              <PrimitiveType xlink:title="Int32"
xlink:href="#Smp.Int32"/>
           </Type>
           <Type xsi:type="Catalogue:Interface"
Id="Smp.Services.ITimeKeeper" Name="ITimeKeeper"
                 Visibility="public"
                 Uuid="d5458977-e618-11dc-ab64-bf8df6d7b83a">
              <Description>This interface gives access to the
Time Keeper Service.</Description>
              <Constant
Id="Smp.Services.ITimeKeeper.SMP_TimeKeeper"
Name="SMP_TimeKeeper"
                        Visibility="public">
                 <Description>Name of the TimeKeeper
service.</Description>
                 <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
                 <Value xsi:type="Types:String8Value"
Value="TimeKeeper"/>
              </Constant>
              <Operation
Id="Smp.Services.ITimeKeeper.GetSimulationTime"
Name="GetSimulationTime"
                         Visibility="public">
                 <Description>Return Simulation
time.</Description>
```

```
                        <Parameter
Id="Smp.Services.ITimeKeeper.GetSimulationTime.return"
Name="return"
                                    Direction="return">
                             <Description>Current simulation
time.</Description>
                             <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
                        </Parameter>
                   </Operation>
                   <Operation
Id="Smp.Services.ITimeKeeper.GetEpochTime" Name="GetEpochTime"
                               Visibility="public">
                        <Description>Return Epoch time.</Description>
                        <Parameter
Id="Smp.Services.ITimeKeeper.GetEpochTime.return"
Name="return"
                                    Direction="return">
                             <Description>Current epoch
time.</Description>
                             <Type xlink:title="DateTime"
xlink:href="#Smp.DateTime"/>
                        </Parameter>
                   </Operation>
                   <Operation
Id="Smp.Services.ITimeKeeper.GetMissionTime"
Name="GetMissionTime"
                               Visibility="public">
                        <Description>Return Mission time.</Description>
                        <Parameter
Id="Smp.Services.ITimeKeeper.GetMissionTime.return"
Name="return"
                                    Direction="return">
                             <Description>Current mission
time.</Description>
                             <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
                        </Parameter>
                   </Operation>
                   <Operation
Id="Smp.Services.ITimeKeeper.GetZuluTime" Name="GetZuluTime"
                               Visibility="public">
                        <Description>Return Zulu time.</Description>
                        <Parameter
Id="Smp.Services.ITimeKeeper.GetZuluTime.return" Name="return"
                                    Direction="return">
                             <Description>Current Zulu
time.</Description>
                             <Type xlink:title="DateTime"
xlink:href="#Smp.DateTime"/>
                        </Parameter>
```

```
            </Operation>
            <Operation
Id="Smp.Services.ITimeKeeper.SetEpochTime" Name="SetEpochTime"
                       Visibility="public">
               <Description>Set Epoch time.
</Description>
               <Parameter
Id="Smp.Services.ITimeKeeper.SetEpochTime.epochTime"
Name="epochTime">
                  <Description>New epoch time.</Description>
                  <Type xlink:title="DateTime"
xlink:href="#Smp.DateTime"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Services.ITimeKeeper.SetMissionStart"
Name="SetMissionStart"
                       Visibility="public">
               <Description>Set Mission time by defining the
mission start time.</Description>
               <Parameter
Id="Smp.Services.ITimeKeeper.SetMissionStart.missionStart"
Name="missionStart">
                  <Description>New mission start date and
time.</Description>
                  <Type xlink:title="DateTime"
xlink:href="#Smp.DateTime"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Services.ITimeKeeper.SetMissionTime"
Name="SetMissionTime"
                       Visibility="public">
               <Description>Set Mission time.</Description>
               <Parameter
Id="Smp.Services.ITimeKeeper.SetMissionTime.missionTime"
Name="missionTime">
                  <Description>New mission time.</Description>
                  <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
               </Parameter>
            </Operation>
            <Base xlink:title="IService"
xlink:href="#Smp.IService"/>
         </Type>
         <Type xsi:type="Types:Enumeration"
Id="Smp.Services.TimeKind" Name="TimeKind"
               Visibility="public"
               Uuid="d54589a6-e618-11dc-ab64-bf8df6d7b83a">
            <Description>Enumeration of supported time
kinds.</Description>
```

```
            <Literal
Id="Smp.Services.TimeKind.TK_SimulationTime"
Name="TK_SimulationTime" Value="0">
               <Description>Simulation time.</Description>
            </Literal>
            <Literal Id="Smp.Services.TimeKind.TK_MissionTime"
Name="TK_MissionTime" Value="1">
               <Description>Mission time.</Description>
            </Literal>
            <Literal Id="Smp.Services.TimeKind.TK_EpochTime"
Name="TK_EpochTime" Value="2">
               <Description>Epoch time.</Description>
            </Literal>
            <Literal Id="Smp.Services.TimeKind.TK_ZuluTime"
Name="TK_ZuluTime" Value="3">
               <Description>Zulu time.</Description>
            </Literal>
         </Type>
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Services.IScheduler" Name="IScheduler"
               Visibility="public"
               Uuid="d54589b4-e618-11dc-ab64-bf8df6d7b83a">
            <Description>This interface gives access to the
Scheduler Service.</Description>
            <Constant
Id="Smp.Services.IScheduler.SMP_Scheduler"
Name="SMP_Scheduler"
                      Visibility="public">
               <Description>Name of the Scheduler
service.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="Scheduler"/>
            </Constant>
            <NestedType xsi:type="Types:Exception"
Id="Smp.Services.IScheduler.InvalidEventTime"
                        Name="InvalidEventTime"
                        Visibility="public"
                        Uuid="d54a451c-e618-11dc-ab64-
bf8df6d7b83a">
               <Description>This exception is thrown by one of
the AddEvent() methods of the scheduler when the time
specified for the first execution of the event is in the
past.</Description>
               <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
            </NestedType>
            <NestedType xsi:type="Types:Exception"
Id="Smp.Services.IScheduler.InvalidCycleTime"
                        Name="InvalidCycleTime"
                        Visibility="public"
                        Uuid="d54a451e-e618-11dc-ab64-
bf8df6d7b83a">
               <Description>This exception is thrown by one of
the AddEvent() methods of the scheduler when the event is a
cyclic event (i.e. repeat is not 0), but the cycle time
specified is not a positive duration.</Description>
               <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
            </NestedType>
            <Operation
Id="Smp.Services.IScheduler.AddImmediateEvent"
Name="AddImmediateEvent"
                       Visibility="public">
               <Description>Add an immediate event to the
scheduler.</Description>
               <Parameter
Id="Smp.Services.IScheduler.AddImmediateEvent.entryPoint"
Name="entryPoint">
                  <Description>Entry point to call from
event.</Description>
                  <Type xlink:title="IEntryPoint"
xlink:href="#Smp.IEntryPoint"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddImmediateEvent.return"
Name="return"
                          Direction="return">
                  <Description>Event identifier that can be
used to change or remove event.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Services.IScheduler.AddSimulationTimeEvent"
                       Name="AddSimulationTimeEvent"
                       Visibility="public">
               <Description>Add event to scheduler that is
called based on simulation time.</Description>
               <Parameter
Id="Smp.Services.IScheduler.AddSimulationTimeEvent.return"
Name="return"
                          Direction="return">
                  <Description>Event identifier that can be
used to change or remove event.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddSimulationTimeEvent.entryPoint"
```

```
                          Name="entryPoint">
                  <Description>Entry point to call from
event.</Description>
                  <Type xlink:title="IEntryPoint"
xlink:href="#Smp.IEntryPoint"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddSimulationTimeEvent.simulationT
ime"
                          Name="simulationTime">
                  <Description>Duration from now when to
trigger the event for the first time.</Description>
                  <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddSimulationTimeEvent.cycleTime"
Name="cycleTime">
                  <Description>Duration between two triggers
of the event.</Description>
                  <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
                  <Default xsi:type="Types:DurationValue"
Value="PT0S"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddSimulationTimeEvent.repeat"
Name="repeat">
                  <Description>Number of times the event shall
be repeated, or 0 for a single event, or -1 for no
limit.</Description>
                  <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
                  <Default xsi:type="Types:Int64Value"
Value="0"/>
               </Parameter>
               <RaisedException xlink:title="InvalidCycleTime"

xlink:href="#Smp.Services.IScheduler.InvalidCycleTime"/>
               <RaisedException xlink:title="InvalidEventTime"

xlink:href="#Smp.Services.IScheduler.InvalidEventTime"/>
            </Operation>
            <Operation
Id="Smp.Services.IScheduler.AddMissionTimeEvent"
Name="AddMissionTimeEvent"
                        Visibility="public">
               <Description>Add event to scheduler that is
called based on mission time.</Description>
```

```
               <Parameter
Id="Smp.Services.IScheduler.AddMissionTimeEvent.return"
Name="return"
                          Direction="return">
                  <Description>Event identifier that can be
used to change or remove event.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddMissionTimeEvent.entryPoint"
Name="entryPoint">
                  <Description>Entry point to call from
event.</Description>
                  <Type xlink:title="IEntryPoint"
xlink:href="#Smp.IEntryPoint"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddMissionTimeEvent.missionTime"
Name="missionTime">
                  <Description>Absolute mission time when to
trigger the event for the first time.</Description>
                  <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddMissionTimeEvent.cycleTime"
Name="cycleTime">
                  <Description>Duration between two triggers
of the event.</Description>
                  <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
                  <Default xsi:type="Types:DurationValue"
Value="PT0S"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddMissionTimeEvent.repeat"
Name="repeat">
                  <Description>Number of times the event shall
be repeated, or 0 for a single event, or -1 for no
limit.</Description>
                  <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
                  <Default xsi:type="Types:Int64Value"
Value="0"/>
               </Parameter>
               <RaisedException xlink:title="InvalidCycleTime"

xlink:href="#Smp.Services.IScheduler.InvalidCycleTime"/>
               <RaisedException xlink:title="InvalidEventTime"
```

```
xlink:href="#Smp.Services.IScheduler.InvalidEventTime"/>
            </Operation>
            <Operation
Id="Smp.Services.IScheduler.AddEpochTimeEvent"
Name="AddEpochTimeEvent"
                       Visibility="public">
               <Description>Add event to scheduler that is
called based on epoch time.</Description>
               <Parameter
Id="Smp.Services.IScheduler.AddEpochTimeEvent.return"
Name="return"
                          Direction="return">
                  <Description>Event identifier that can be
used to change or remove event.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddEpochTimeEvent.entryPoint"
Name="entryPoint">
                  <Description>Entry point to call from
event.</Description>
                  <Type xlink:title="IEntryPoint"
xlink:href="#Smp.IEntryPoint"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddEpochTimeEvent.epochTime"
Name="epochTime">
                  <Description>Epoch time when to trigger the
event for the first time.</Description>
                  <Type xlink:title="DateTime"
xlink:href="#Smp.DateTime"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddEpochTimeEvent.cycleTime"
Name="cycleTime">
                  <Description>Duration between two triggers
of the event.</Description>
                  <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
                  <Default xsi:type="Types:DurationValue"
Value="PT0S"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddEpochTimeEvent.repeat"
Name="repeat">
                  <Description>Number of times the event shall
be repeated, or 0 for a single event, or -1 for no
limit.</Description>
```

```
                  <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
                  <Default xsi:type="Types:Int64Value"
Value="0"/>
               </Parameter>
               <RaisedException xlink:title="InvalidCycleTime"

xlink:href="#Smp.Services.IScheduler.InvalidCycleTime"/>
               <RaisedException xlink:title="InvalidEventTime"

xlink:href="#Smp.Services.IScheduler.InvalidEventTime"/>
            </Operation>
            <Operation
Id="Smp.Services.IScheduler.AddZuluTimeEvent"
Name="AddZuluTimeEvent"
                       Visibility="public">
               <Description>Add event to scheduler that is
called based on Zulu time.</Description>
               <Parameter
Id="Smp.Services.IScheduler.AddZuluTimeEvent.return"
Name="return"
                          Direction="return">
                  <Description>Event identifier that can be
used to change or remove event.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddZuluTimeEvent.entryPoint"
Name="entryPoint">
                  <Description>Entry point to call from
event.</Description>
                  <Type xlink:title="IEntryPoint"
xlink:href="#Smp.IEntryPoint"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddZuluTimeEvent.simulationTime"
                          Name="simulationTime">
                  <Description>Absolute (Zulu) time when to
trigger the event for the first time.</Description>
                  <Type xlink:title="DateTime"
xlink:href="#Smp.DateTime"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddZuluTimeEvent.cycleTime"
Name="cycleTime">
                  <Description>Duration between two triggers
of the event.</Description>
                  <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
```

```
                  <Default xsi:type="Types:DurationValue"
Value="PT0S"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.AddZuluTimeEvent.repeat"
Name="repeat">
                  <Description>Number of times the event shall
be repeated, or 0 for a single event, or -1 for no
limit.</Description>
                  <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
                  <Default xsi:type="Types:Int64Value"
Value="0"/>
               </Parameter>
               <RaisedException xlink:title="InvalidCycleTime"

xlink:href="#Smp.Services.IScheduler.InvalidCycleTime"/>
               <RaisedException xlink:title="InvalidEventTime"

xlink:href="#Smp.Services.IScheduler.InvalidEventTime"/>
            </Operation>
            <Operation
Id="Smp.Services.IScheduler.SetEventSimulationTime"
                       Name="SetEventSimulationTime"
                       Visibility="public">
               <Description>Update when an existing simulation
time event on the scheduler shall be triggered.</Description>
               <Parameter
Id="Smp.Services.IScheduler.SetEventSimulationTime.event"
Name="event">
                  <Description>Identifier of event to
modify.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.SetEventSimulationTime.simulationT
ime"
                          Name="simulationTime">
                  <Description>Duration from now when to
trigger event.</Description>
                  <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
               </Parameter>
               <RaisedException xlink:title="InvalidEventId"
xlink:href="#Smp.Services.InvalidEventId"/>
            </Operation>
            <Operation
Id="Smp.Services.IScheduler.SetEventMissionTime"
Name="SetEventMissionTime"
                       Visibility="public">
               <Description>Update when an existing mission
time event on the scheduler shall be triggered.</Description>
               <Parameter
Id="Smp.Services.IScheduler.SetEventMissionTime.event"
Name="event">
                  <Description>Identifier of event to
modify.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.SetEventMissionTime.missionTime"
Name="missionTime">
                  <Description>Absolute mission time when to
trigger event.</Description>
                  <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
               </Parameter>
               <RaisedException xlink:title="InvalidEventId"
xlink:href="#Smp.Services.InvalidEventId"/>
            </Operation>
            <Operation
Id="Smp.Services.IScheduler.SetEventEpochTime"
Name="SetEventEpochTime"
                       Visibility="public">
               <Description>Update when an existing epoch time
event on the scheduler (an event that has been registered
using AddEpochTimeEvent()) shall be triggered.</Description>
               <Parameter
Id="Smp.Services.IScheduler.SetEventEpochTime.event"
Name="event">
                  <Description>Identifier of event to
modify.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.SetEventEpochTime.epochTime"
Name="epochTime">
                  <Description>Epoch time when to trigger
event.</Description>
                  <Type xlink:title="DateTime"
xlink:href="#Smp.DateTime"/>
               </Parameter>
               <RaisedException xlink:title="InvalidEventId"
xlink:href="#Smp.Services.InvalidEventId"/>
            </Operation>
            <Operation
Id="Smp.Services.IScheduler.SetEventZuluTime"
Name="SetEventZuluTime"
                       Visibility="public">
               <Description>Update when an existing zulu time
event on the scheduler shall be triggered.</Description>
               <Parameter
Id="Smp.Services.IScheduler.SetEventZuluTime.event"
Name="event">
                  <Description>Identifier of event to
modify.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.SetEventZuluTime.zuluTime"
Name="zuluTime">
                  <Description>Absolute (Zulu) time when to
trigger event.</Description>
                  <Type xlink:title="DateTime"
xlink:href="#Smp.DateTime"/>
               </Parameter>
               <RaisedException xlink:title="InvalidEventId"
xlink:href="#Smp.Services.InvalidEventId"/>
            </Operation>
            <Operation
Id="Smp.Services.IScheduler.SetEventCycleTime"
Name="SetEventCycleTime"
                       Visibility="public">
               <Description>Update cycle time of an existing
event on the scheduler.</Description>
               <Parameter
Id="Smp.Services.IScheduler.SetEventCycleTime.event"
Name="event">
                  <Description>Identifier of event to
modify.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.SetEventCycleTime.cycleTime"
Name="cycleTime">
                  <Description>Duration between two triggers
of the event.</Description>
                  <Type xlink:title="Duration"
xlink:href="#Smp.Duration"/>
               </Parameter>
               <RaisedException xlink:title="InvalidEventId"
xlink:href="#Smp.Services.InvalidEventId"/>
            </Operation>
            <Operation
Id="Smp.Services.IScheduler.SetEventCount"
Name="SetEventCount"
                       Visibility="public">
               <Description>Update the count of an existing
event on the scheduler.</Description>
               <Parameter
Id="Smp.Services.IScheduler.SetEventCount.event" Name="event">
                  <Description>Identifier of event to
modify.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IScheduler.SetEventCount.count" Name="count">
                  <Description>Number of times the event shall
be repeated, or 0 for a single event, or -1 for no
limit.</Description>
                  <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
               </Parameter>
               <RaisedException xlink:title="InvalidEventId"
xlink:href="#Smp.Services.InvalidEventId"/>
            </Operation>
            <Operation
Id="Smp.Services.IScheduler.RemoveEvent" Name="RemoveEvent"
Visibility="public">
               <Description>Remove an event from the
scheduler.</Description>
               <Parameter
Id="Smp.Services.IScheduler.RemoveEvent.event" Name="event">
                  <Description>Event identifier of the event
to remove.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <RaisedException xlink:title="InvalidEventId"
xlink:href="#Smp.Services.InvalidEventId"/>
            </Operation>
            <Base xlink:title="IService"
xlink:href="#Smp.IService"/>
         </Type>
         <Type xsi:type="Types:Integer"
Id="Smp.Services.EventId" Name="EventId"
               Visibility="public"
               Uuid="d54589a4-e618-11dc-ab64-bf8df6d7b83a">
            <Description>Identifier of global event of
scheduler or event manager service.</Description>
            <PrimitiveType xlink:title="Int64"
xlink:href="#Smp.Int64"/>
         </Type>
         <Type xsi:type="Types:Exception"
Id="Smp.Services.InvalidEventId"
               Name="InvalidEventId"
               Visibility="public"
```

```
                Uuid="d54f26d2-e618-11dc-ab64-bf8df6d7b83a">
            <Description>This exception is raised when an
invalid event id is provided, e.g. when calling Subscribe(),
Unsubscribe() or Emit() of the Event Manager (using an invalid
global event id), or when calling SetEventSimulationTime(),
SetEventMissionTime(), SetEventEpochTime(),
SetEventZuluTime(), SetEventCycleTime(), SetEventCount() or
RemoveEvent() of the Scheduler (using an invalid scheduler
event id).</Description>
            <Field Id="Smp.Services.InvalidEventId.eventId"
Name="eventId" Visibility="public">
                <Description>Invalid event
identifier.</Description>
                <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
            </Field>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </Type>
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Services.IEventManager"
                Name="IEventManager"
                Visibility="public"
                Uuid="d54a4520-e618-11dc-ab64-bf8df6d7b83a">
            <Description>This interface gives access to the
Event Manager Service.</Description>
            <Constant
Id="Smp.Services.IEventManager.SMP_EventManager"
Name="SMP_EventManager"
                       Visibility="public">
                <Description>Name of the EventManager
service.</Description>
                <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
                <Value xsi:type="Types:String8Value"
Value="EventManager"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_LeaveConnectingId"
                       Name="SMP_LeaveConnectingId"
                       Visibility="public">
                <Description>This event is raised when leaving
the Connecting state with an automatic state transition to
Initializing state.</Description>
                <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
                <Value xsi:type="Types:Int64Value" Value="1"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_LeaveConnecting"
Name="SMP_LeaveConnecting"
```

```
                      Visibility="public">
               <Description>Leave Connecting
state.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_LeaveConnecting"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EnterInitialisingId"
                      Name="SMP_EnterInitialisingId"
                      Visibility="public">
               <Description>This event is raised when entering
the Initialising state with an automatic state transition from
Connecting state, or with the Initialise() state transition.
</Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               <Value xsi:type="Types:Int64Value" Value="2"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EnterInitialising"
                      Name="SMP_EnterInitialising"
                      Visibility="public">
               <Description>Enter Initialising
state.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_EnterInitialising"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_LeaveInitialisingId"
                      Name="SMP_LeaveInitialisingId"
                      Visibility="public">
               <Description>This event is raised when leaving
the Initialising state with an automatic state transition to
Standby state.</Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               <Value xsi:type="Types:Int64Value" Value="3"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_LeaveInitialising"
                      Name="SMP_LeaveInitialising"
                      Visibility="public">
               <Description>Leave Initialising
state.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
```

```
               <Value xsi:type="Types:String8Value"
Value="SMP_LeaveInitialising"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EnterStandbyId"
Name="SMP_EnterStandbyId"
                      Visibility="public">
               <Description>This event is raised when entering
the Standby state with an automatic state transition from
Initialising, Storing or Restoring state, or with the Hold()
state transition command from Executing state. </Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               <Value xsi:type="Types:Int64Value" Value="4"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EnterStandby"
Name="SMP_EnterStandby"
                      Visibility="public">
               <Description>Enter Standby state.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_EnterStandby"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_LeaveStandbyId"
Name="SMP_LeaveStandbyId"
                      Visibility="public">
               <Description>This event is raised when leaving
the Standby state with the Run() state transition command to
Executing state, with the Store() state transition command to
Storing state, with the Restore() state transition command to
Restoring state, or with the Initialise() state transition
command to Initialising state. </Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               <Value xsi:type="Types:Int64Value" Value="5"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_LeaveStandby"
Name="SMP_LeaveStandby"
                      Visibility="public">
               <Description>Leave Standby state.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_LeaveStandby"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EnterExecutingId"
```

```
                      Name="SMP_EnterExecutingId"
                      Visibility="public">
               <Description>This event is raised when entering
the Executing state with the Run() state transition command
from Standby state.</Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               <Value xsi:type="Types:Int64Value" Value="6"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EnterExecuting"
Name="SMP_EnterExecuting"
                      Visibility="public">
               <Description>Enter Executing
state.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_EnterExecuting"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_LeaveExecutingId"
                      Name="SMP_LeaveExecutingId"
                      Visibility="public">
               <Description>This event is raised when leaving
the Executing state with the Hold() state transition command
to Standby state.</Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               <Value xsi:type="Types:Int64Value" Value="7"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_LeaveExecuting"
Name="SMP_LeaveExecuting"
                      Visibility="public">
               <Description>Leave Executing
state.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_LeaveExecuting"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EnterStoringId"
Name="SMP_EnterStoringId"
                      Visibility="public">
               <Description>This event is raised when entering
the Storing state with the Store() state transition command
from Standby state.</Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
```

```
            <Value xsi:type="Types:Int64Value" Value="8"/>
         </Constant>
         <Constant
Id="Smp.Services.IEventManager.SMP_EnterStoring"
Name="SMP_EnterStoring"
                      Visibility="public">
            <Description>Enter Storing state.</Description>
            <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            <Value xsi:type="Types:String8Value"
Value="SMP_EnterStoring"/>
         </Constant>
         <Constant
Id="Smp.Services.IEventManager.SMP_LeaveStoringId"
Name="SMP_LeaveStoringId"
                      Visibility="public">
            <Description>This event is raised when leaving
the Storing state with an automatic state transition to
Standby state.</Description>
            <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
            <Value xsi:type="Types:Int64Value" Value="9"/>
         </Constant>
         <Constant
Id="Smp.Services.IEventManager.SMP_LeaveStoring"
Name="SMP_LeaveStoring"
                      Visibility="public">
            <Description>Leave Storing state.</Description>
            <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            <Value xsi:type="Types:String8Value"
Value="SMP_LeaveStoring"/>
         </Constant>
         <Constant
Id="Smp.Services.IEventManager.SMP_EnterRestoringId"
                      Name="SMP_EnterRestoringId"
                      Visibility="public">
            <Description>This event is raised when entering
the Restoring state with the Restore() state transition
command from Standby state.</Description>
            <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
            <Value xsi:type="Types:Int64Value" Value="10"/>
         </Constant>
         <Constant
Id="Smp.Services.IEventManager.SMP_EnterRestoring"
Name="SMP_EnterRestoring"
                      Visibility="public">
            <Description>Enter Restoring
state.</Description>
```

```
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_EnterRestoring"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_LeaveRestoringId"
                      Name="SMP_LeaveRestoringId"
                      Visibility="public">
               <Description>This event is raised when leaving
the Restoring state with an automatic state transition to
Standby state.</Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               <Value xsi:type="Types:Int64Value" Value="11"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_LeaveRestoring"
Name="SMP_LeaveRestoring"
                      Visibility="public">
               <Description>Leave Restoring
state.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_LeaveRestoring"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EnterExitingId"
Name="SMP_EnterExitingId"
                      Visibility="public">
               <Description>This event is raised when entering
the Exiting state with the Exit() state transition command
from Standby state.</Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               <Value xsi:type="Types:Int64Value" Value="12"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EnterExiting"
Name="SMP_EnterExiting"
                      Visibility="public">
               <Description>Enter Exiting state.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_EnterExiting"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EnterAbortingId"
Name="SMP_EnterAbortingId"
```

```
                      Visibility="public">
               <Description>This event is raised when entering
the Aborting state with the Abort() state transition command
from any other state.</Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               <Value xsi:type="Types:Int64Value" Value="13"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EnterAborting"
Name="SMP_EnterAborting"
                      Visibility="public">
               <Description>Enter Aborting
state.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_EnterAborting"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EpochTimeChangedId"
                      Name="SMP_EpochTimeChangedId"
                      Visibility="public">
               <Description>This event is raised when changing
the epoch time with the SetEpochTime() method of the time
keeper service.</Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               <Value xsi:type="Types:Int64Value" Value="14"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EpochTimeChanged"
                      Name="SMP_EpochTimeChanged"
                      Visibility="public">
               <Description>Epoch Time has
changed.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_EpochTimeChanged"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_MissionTimeChangedId"
                      Name="SMP_MissionTimeChangedId"
                      Visibility="public">
               <Description>This event is raised when changing
the mission time with one of the SetMissionTime() and
SetMissionStart() methods of the time keeper service.
</Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
```

```
               <Value xsi:type="Types:Int64Value" Value="15"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_MissionTimeChanged"
                      Name="SMP_MissionTimeChanged"
                      Visibility="public">
               <Description>Mission time has
changed.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_MissionTimeChanged"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EnterReconnectingId"
                      Name="SMP_EnterReconnectingId"
                      Visibility="public">
               <Description>This event is raised when entering
the Reconnecting state with the Reconnect() state transition
from Standby state.</Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               <Value xsi:type="Types:Int64Value" Value="16"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_EnterReconnecting"
                      Name="SMP_EnterReconnecting"
                      Visibility="public">
               <Description>Enter Reconnecting
state.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_EnterReconnecting"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_LeaveReconnectingId"
                      Name="SMP_LeaveReconnectingId"
                      Visibility="public">
               <Description>This event is raised when leaving
the Reconnecting state with an automatic state transition to
Standby state.</Description>
               <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               <Value xsi:type="Types:Int64Value" Value="17"/>
            </Constant>
            <Constant
Id="Smp.Services.IEventManager.SMP_LeaveReconnecting"
                      Name="SMP_LeaveReconnecting"
                      Visibility="public">
               <Description>Leave Reconnecting
state.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="SMP_LeaveReconnecting"/>
            </Constant>
            <NestedType xsi:type="Types:Exception"
Id="Smp.Services.IEventManager.AlreadySubscribed"
                        Name="AlreadySubscribed"
                        Visibility="public"
                        Uuid="d54f26af-e618-11dc-ab64-
bf8df6d7b83a">
               <Description>This exception is raised when
trying to subscribe an entry point to an event that is already
subscribed.</Description>
               <Field
Id="Smp.Services.IEventManager.AlreadySubscribed.eventName"
Name="eventName"
                      Visibility="public">
                  <Description>Name of event the entry point
is already subscribed to.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Field>
               <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
               <Association
Id="Smp.Services.IEventManager.AlreadySubscribed.entryPoint"
Name="entryPoint"
                            Visibility="public">
                  <Description>Entry point that is already
subscribed.</Description>
                  <Type xlink:title="IEntryPoint"
xlink:href="#Smp.IEntryPoint"/>
               </Association>
            </NestedType>
            <NestedType xsi:type="Types:Exception"
Id="Smp.Services.IEventManager.NotSubscribed"
                        Name="NotSubscribed"
                        Visibility="public"
                        Uuid="d54f26b6-e618-11dc-ab64-
bf8df6d7b83a">
               <Description>This exception is raised when
trying to unsubscribe an entry point from an event that is not
subscribed to it.</Description>
               <Field
Id="Smp.Services.IEventManager.NotSubscribed.eventName"
Name="eventName"
                      Visibility="public">
                  <Description>Name of event the entry point
is not subscribed to.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Field>
               <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
               <Association
Id="Smp.Services.IEventManager.NotSubscribed.entryPoint"
Name="entryPoint"
                            Visibility="public">
                  <Description>Entry point that is not
subscribed.</Description>
                  <Type xlink:title="IEntryPoint"
xlink:href="#Smp.IEntryPoint"/>
               </Association>
            </NestedType>
            <Operation
Id="Smp.Services.IEventManager.QueryEventId"
Name="QueryEventId"
                       Visibility="public">
               <Description>Get unique event identifier for an
event name.</Description>
               <Parameter
Id="Smp.Services.IEventManager.QueryEventId.return"
Name="return"
                          Direction="return">
                  <Description>Event identifier for global
event with given name.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IEventManager.QueryEventId.eventName"
Name="eventName">
                  <Description>Name of the global
event.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Services.IEventManager.Subscribe" Name="Subscribe"
Visibility="public">
               <Description>Subscribe entry point to a global
event.</Description>
               <Parameter
Id="Smp.Services.IEventManager.Subscribe.event" Name="event">
                  <Description>Event identifier of global
event to subscribe to.</Description>
```

```
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IEventManager.Subscribe.entryPoint"
Name="entryPoint">
                  <Description>Entry point to subscribe to
global event.</Description>
                  <Type xlink:title="IEntryPoint"
xlink:href="#Smp.IEntryPoint"/>
               </Parameter>
               <RaisedException xlink:title="InvalidEventId"
xlink:href="#Smp.Services.InvalidEventId"/>
               <RaisedException
xlink:title="AlreadySubscribed"

xlink:href="#Smp.Services.IEventManager.AlreadySubscribed"/>
            </Operation>
            <Operation
Id="Smp.Services.IEventManager.Unsubscribe" Name="Unsubscribe"
                       Visibility="public">
               <Description>Unsubscribe entry point from a
global event.</Description>
               <Parameter
Id="Smp.Services.IEventManager.Unsubscribe.event"
Name="event">
                  <Description>Event identifier of global
event to unsubscribe from.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IEventManager.Unsubscribe.entryPoint"
Name="entryPoint">
                  <Description>Entry point to unsubscribe from
global event.</Description>
                  <Type xlink:title="IEntryPoint"
xlink:href="#Smp.IEntryPoint"/>
               </Parameter>
               <RaisedException xlink:title="InvalidEventId"
xlink:href="#Smp.Services.InvalidEventId"/>
               <RaisedException xlink:title="NotSubscribed"

xlink:href="#Smp.Services.IEventManager.NotSubscribed"/>
            </Operation>
            <Operation Id="Smp.Services.IEventManager.Emit"
Name="Emit" Visibility="public">
               <Description>Emit a global event.</Description>
               <Parameter
Id="Smp.Services.IEventManager.Emit.event" Name="event">
```

```
                  <Description>Event identifier of global
event to emit.</Description>
                  <Type xlink:title="EventId"
xlink:href="#Smp.Services.EventId"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IEventManager.Emit.synchronous"
Name="synchronous">
                  <Description>Flag whether to emit the given
event synchronously (the default) or
asynchronously.</Description>
                  <Type xlink:title="Bool"
xlink:href="#Smp.Bool"/>
                  <Default xsi:type="Types:BoolValue"
Value="true"/>
               </Parameter>
            </Operation>
            <Base xlink:title="IService"
xlink:href="#Smp.IService"/>
         </Type>
         <Type xsi:type="Catalogue:Interface"
Id="Smp.Services.IResolver" Name="IResolver"
               Visibility="public"
               Uuid="d54f26bd-e618-11dc-ab64-bf8df6d7b83a">
            <Description>This interface gives access to the
Resolver Service.</Description>
            <Constant Id="Smp.Services.IResolver.SMP_Resolver"
Name="SMP_Resolver"
                      Visibility="public">
               <Description>Name of the Resolver
service.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               <Value xsi:type="Types:String8Value"
Value="Resolver"/>
            </Constant>
            <Operation
Id="Smp.Services.IResolver.ResolveAbsolute"
Name="ResolveAbsolute"
                       Visibility="public">
               <Description>Resolve reference to component via
absolute path.
</Description>
               <Parameter
Id="Smp.Services.IResolver.ResolveAbsolute.return"
Name="return"
                          Direction="return">
                  <Description>Component identified by path,
or null if no component with the given path could be
found.</Description>
```

```
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IResolver.ResolveAbsolute.absolutePath"
Name="absolutePath">
                  <Description>Absolute path to component in
simulation.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Parameter>
            </Operation>
            <Operation
Id="Smp.Services.IResolver.ResolveRelative"
Name="ResolveRelative"
                       Visibility="public">
               <Description>Resolve reference to component via
relative path.</Description>
               <Parameter
Id="Smp.Services.IResolver.ResolveRelative.return"
Name="return"
                          Direction="return">
                  <Description>Component identified by path,
or null if no component with the given path could be
found.</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IResolver.ResolveRelative.relativePath"
Name="relativePath">
                  <Description>Relative path to component in
simulation.</Description>
                  <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
               </Parameter>
               <Parameter
Id="Smp.Services.IResolver.ResolveRelative.sender"
Name="sender">
                  <Description>Component that asks for
resolving the reference.</Description>
                  <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
               </Parameter>
            </Operation>
            <Base xlink:title="IService"
xlink:href="#Smp.IService"/>
         </Type>
      </Namespace>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.Char8"
Name="Char8" Visibility="public"
```

```
            Uuid="11a891e1-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>8 bit character</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.String8"
Name="String8"
            Visibility="public"
            Uuid="11a891e5-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>8 bit character string</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.Float32"
Name="Float32"
            Visibility="public"
            Uuid="11a891e9-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>32 bit single-precision
float</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.Float64"
Name="Float64"
            Visibility="public"
            Uuid="11a891eb-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>64 bit double-precision
float</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.Int8"
Name="Int8" Visibility="public"
            Uuid="11a891ed-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>8 bit   signed integer</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.UInt8"
Name="UInt8" Visibility="public"
            Uuid="11a891ef-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>8 bit unsigned integer</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.Int16"
Name="Int16" Visibility="public"
            Uuid="11a891f1-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>16 bit   signed integer</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.UInt16"
Name="UInt16"
            Visibility="public"
            Uuid="11a891f3-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>16 bit unsigned integer</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.Int32"
Name="Int32" Visibility="public"
            Uuid="11a891f5-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>32 bit   signed integer</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.UInt32"
Name="UInt32"
```

```
            Visibility="public"
            Uuid="11a891f7-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>32 bit unsigned integer</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.Int64"
Name="Int64" Visibility="public"
            Uuid="11a891f9-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>64 bit   signed integer</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.UInt64"
Name="UInt64"
            Visibility="public"
            Uuid="11a891fb-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>64 bit unsigned integer</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.Bool"
Name="Bool" Visibility="public"
            Uuid="11a891fd-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>boolean with true or false</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.DateTime"
Name="DateTime"
            Visibility="public"
            Uuid="11a891ff-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>point in time in nanoseconds
relative to MJD 2000+0.5</Description>
      </Type>
      <Type xsi:type="Types:PrimitiveType" Id="Smp.Duration"
Name="Duration"
            Visibility="public"
            Uuid="11a89201-1a78-11dc-9f3d-2d5be8a8137f">
         <Description>duration in nanoseconds
</Description>
      </Type>
      <Type xsi:type="Types:Array" Id="Smp.UuidBytes"
Name="UuidBytes" Visibility="public"
            Uuid="d55b0e57-e618-11dc-ab64-bf8df6d7b83a"
            Size="8">
         <Description>Final 8 bytes of Uuid.</Description>
         <ItemType xlink:title="UInt8"
xlink:href="#Smp.UInt8"/>
      </Type>
      <Type xsi:type="Types:Structure" Id="Smp.Uuid"
Name="Uuid" Visibility="public"
            Uuid="d55b0e59-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Universally Unique
Identifier.</Description>
         <Field Id="Smp.Uuid.Data1" Name="Data1"
Visibility="public">
            <Description>8 hex nibbles.</Description>
            <Type xlink:title="UInt32"
xlink:href="#Smp.UInt32"/>
            <Default xsi:type="Types:UInt32Value" Value="0"/>
         </Field>
         <Field Id="Smp.Uuid.Data2" Name="Data2"
Visibility="public">
            <Description>4 hex nibbles.</Description>
            <Type xlink:title="UInt16"
xlink:href="#Smp.UInt16"/>
            <Default xsi:type="Types:UInt16Value" Value="0"/>
         </Field>
         <Field Id="Smp.Uuid.Data3" Name="Data3"
Visibility="public">
            <Description>4 hex nibbles.</Description>
            <Type xlink:title="UInt16"
xlink:href="#Smp.UInt16"/>
            <Default xsi:type="Types:UInt16Value" Value="0"/>
         </Field>
         <Field Id="Smp.Uuid.Data4" Name="Data4"
Visibility="public">
            <Description>4+12 hex nibbles.</Description>
            <Type xlink:title="UuidBytes"
xlink:href="#Smp.UuidBytes"/>
         </Field>
      </Type>
      <Type xsi:type="Catalogue:Interface"
Id="Smp.IFallibleModel" Name="IFallibleModel"
            Visibility="public"
            Uuid="d572db1a-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface for a fallible model that
exposes its failure state and a collection of
failures.</Description>
         <Operation Id="Smp.IFallibleModel.IsFailed"
Name="IsFailed" Visibility="public">
            <Description>Query for whether the model is
failed. A model is failed when at least one of its failures is
failed.</Description>
            <Parameter Id="Smp.IFallibleModel.IsFailed.return"
Name="return" Direction="return">
               <Description>Whether the model is failed or
not.</Description>
               <Type xlink:title="Bool"
xlink:href="#Smp.Bool"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IFallibleModel.GetFailure"
Name="GetFailure" Visibility="public">
            <Description>Get a failure by name.</Description>
            <Parameter Id="Smp.IFallibleModel.GetFailure.name"
Name="name">
```

```
                    <Description>Name of the failure to
return.</Description>
                    <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
                 </Parameter>
                 <Parameter
Id="Smp.IFallibleModel.GetFailure.return" Name="return"
Direction="return">
                    <Description>Failure queried for by name, or
null if no failure with this name exists.</Description>
                    <Type xlink:title="IFailure"
xlink:href="#Smp.IFailure"/>
                 </Parameter>
              </Operation>
              <Operation Id="Smp.IFallibleModel.GetFailures"
Name="GetFailures" Visibility="public">
                 <Description>Query for the collection of all
failures.</Description>
                 <Parameter
Id="Smp.IFallibleModel.GetFailures.return" Name="return"
Direction="return">
                    <Description>Failure collection of the
model.</Description>
                    <Type xlink:title="FailureCollection"
xlink:href="#Smp.FailureCollection"/>
                 </Parameter>
              </Operation>
              <Base xlink:title="IModel" xlink:href="#Smp.IModel"/>
           </Type>
           <Type xsi:type="Catalogue:Interface" Id="Smp.IFailure"
Name="IFailure"
                 Visibility="public"
                 Uuid="d572db2e-e618-11dc-ab64-bf8df6d7b83a">
              <Description>Interface for a failure.</Description>
              <Operation Id="Smp.IFailure.Fail" Name="Fail"
Visibility="public">
                 <Description>Sets the state of the failure to
failed.</Description>
              </Operation>
              <Operation Id="Smp.IFailure.Unfail" Name="Unfail"
Visibility="public">
                 <Description>Sets the state of the failure to
unfailed.</Description>
              </Operation>
              <Operation Id="Smp.IFailure.IsFailed" Name="IsFailed"
Visibility="public">
                 <Description>Returns whether the failure's state
is set to failed.</Description>
                 <Parameter Id="Smp.IFailure.IsFailed.return"
Name="return" Direction="return">
```

```
               <Description>Returns true if the failure state
is Failed, false otherwise.</Description>
               <Type xlink:title="Bool"
xlink:href="#Smp.Bool"/>
            </Parameter>
         </Operation>
         <Base xlink:title="IObject"
xlink:href="#Smp.IObject"/>
      </Type>
      <Type xsi:type="Catalogue:Interface"
Id="Smp.ILinkingComponent"
            Name="ILinkingComponent"
            Visibility="public"
            Uuid="d5752575-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface for a component which can hold
links to other components.</Description>
         <Operation Id="Smp.ILinkingComponent.RemoveLinks"
Name="RemoveLinks" Visibility="public">
            <Description>Asks a component to remove all its
links to the given target component.
After this method has been called, the component must not try
to access the given target component anymore.</Description>
            <Parameter
Id="Smp.ILinkingComponent.RemoveLinks.target" Name="target">
               <Description>Target component to which all
links shall be removed.</Description>
               <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
            </Parameter>
         </Operation>
         <Base xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
      </Type>
      <Type xsi:type="Types:NativeType" Id="Smp.AnySimple"
Name="AnySimple"
            Visibility="public"
            Uuid="d575259e-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Variant that can store a value of any of
the simple types. The type attribute defines the type used to
represent the value, while the value attribute contains the
actual value.</Description>
         <Platform Name="cpp" Type="NT_AnySimple"
Location="Platform.h"/>
      </Type>
      <Type xsi:type="Types:Enumeration" Id="Smp.ViewKind"
Name="ViewKind"
            Visibility="public"
            Uuid="d579e033-e618-11dc-ab64-bf8df6d7b83a">
         <Description>This enumeration defines possible
options for the View attribute, which can be used to control
```

```
if and how an element is made visible when published to the
simulation infrastructure.
The simulation infrastructure must at least support the "None"
and the "EndUser" roles (i.e. hidden or always visible).
The simulation infrastructure may support the selection of
different user roles ("Debug", "Expert", "End User"), in which
case the "Debug" and the "Expert" role must also be supported
as described.</Description>
         <Literal Id="Smp.ViewKind.VK_None" Name="VK_None"
Value="0">
            <Description>The element is not made visible to
the user (this is the default).</Description>
         </Literal>
         <Literal Id="Smp.ViewKind.VK_Debug" Name="VK_Debug"
Value="1">
            <Description>The element is made visible for
debugging purposes.
The element is not visible to end users. If the simulation
infrastructure supports the selection of different user roles,
then the element shall be visible to "Debug" users
only.</Description>
         </Literal>
         <Literal Id="Smp.ViewKind.VK_Expert" Name="VK_Expert"
Value="2">
            <Description>The element is made visible for
expert users.
The element is not visible to end users. If the simulation
infrastructure supports the selection of different user roles,
then the element shall be visible to "Debug" and "Expert"
users.</Description>
         </Literal>
         <Literal Id="Smp.ViewKind.VK_All" Name="VK_All"
Value="3">
            <Description>The element is made visible to all
users.</Description>
         </Literal>
      </Type>
      <Type xsi:type="Types:Enumeration" Id="Smp.AccessKind"
Name="AccessKind"
            Visibility="public"
            Uuid="e7d5e125-eb8a-11dc-8642-c38618fe0a20">
         <Description>The Access Kind of a property defines
whether it has getter and setter.</Description>
         <Literal Id="Smp.AccessKind.AK_ReadWrite"
Name="AK_ReadWrite" Value="0">
            <Description>Read/Write access, i.e. getter and
setter.</Description>
         </Literal>
         <Literal Id="Smp.AccessKind.AK_ReadOnly"
Name="AK_ReadOnly" Value="1">
```

```
            <Description>Read only access, i.e. only getter
method.</Description>
         </Literal>
         <Literal Id="Smp.AccessKind.AK_WriteOnly"
Name="AK_WriteOnly" Value="2">
            <Description>Write only access, i.e. only setter
method.</Description>
         </Literal>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.IObject"
Name="IObject"
            Visibility="public"
            Uuid="d55b0e67-e618-11dc-ab64-bf8df6d7b83a">
         <Description>This interface is the base interface for
almost all other SMP interfaces. While most interfaces derive
from IComponent, which itself is derived from IObject, some
objects (including IField, IFailure, IEntryPoint, IEventSink,
IEventSource, IContainer and IReference) are directly derived
from IObject.</Description>
         <Operation Id="Smp.IObject.GetName" Name="GetName"
Visibility="public">
            <Description>Return the name of the object
("property getter").</Description>
            <Parameter Id="Smp.IObject.GetName.return"
Name="return" Direction="return">
               <Description>Name of object.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IObject.GetDescription"
Name="GetDescription" Visibility="public">
            <Description>Return the description of the object
("property getter").</Description>
            <Parameter Id="Smp.IObject.GetDescription.return"
Name="return" Direction="return">
               <Description>Description of
object.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
      </Type>
      <Type xsi:type="Types:Exception" Id="Smp.Exception"
Name="Exception"
            Visibility="public"
            Uuid="d5706b05-e618-11dc-ab64-bf8df6d7b83a">
         <Description>This is the base class for all SMP
exceptions.</Description>
         <Field Id="Smp.Exception.name" Name="name"
Visibility="protected">
```

```
            <Description>Name of the exception that is
returned by GetName.</Description>
            <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
         </Field>
         <Field Id="Smp.Exception.description"
Name="description" Visibility="protected">
            <Description>Description of the exception that is
returned by GetDescription.</Description>
            <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
         </Field>
         <Field Id="Smp.Exception.message" Name="message"
Visibility="protected">
            <Description>Description of the problem that is
returned by GetMessage.</Description>
            <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
         </Field>
         <Operation Id="Smp.Exception.GetName" Name="GetName"
Visibility="public">
            <Description>Returns the name of the exception
class. This name can be used e.g. for debugging
purposes.</Description>
            <Parameter Id="Smp.Exception.GetName.return"
Name="return" Direction="return">
               <Description>Name of the exception
class.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.Exception.GetDescription"
Name="GetDescription" Visibility="public">
            <Description>Returns a textual description of the
exception class. This description can be used e.g. for
debugging purposes.</Description>
            <Parameter
Id="Smp.Exception.GetDescription.return" Name="return"
Direction="return">
               <Description>Textual description of the
exception class.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.Exception.GetMessage"
Name="GetMessage" Visibility="public">
            <Description>Returns the description of the
problem encountered. This message can be used e.g. for
debugging purposes.
```

```
</Description>
            <Parameter Id="Smp.Exception.GetMessage.return"
Name="return" Direction="return">
               <Description>Textual description of the problem
encountered.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
      </Type>
      <Type xsi:type="Catalogue:Interface"
Id="Smp.IEventSource" Name="IEventSource"
            Visibility="public"
            Uuid="d55d579c-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface of an event source that event
sinks (IEventSink) can subscribe to.</Description>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IEventSource.AlreadySubscribed"
                     Name="AlreadySubscribed"
                     Visibility="public"
                     Uuid="d55d57ad-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised when trying
to subscribe an event sink to an event source that is already
subscribed.</Description>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
            <Association
Id="Smp.IEventSource.AlreadySubscribed.eventSource"
Name="eventSource"
                         Visibility="public">
               <Description>Event source the event sink is
subscribed to.</Description>
               <Type xlink:title="IEventSource"
xlink:href="#Smp.IEventSource"/>
            </Association>
            <Association
Id="Smp.IEventSource.AlreadySubscribed.eventSink"
Name="eventSink"
                         Visibility="public">
               <Description>Event sink that is already
subscribed.</Description>
               <Type xlink:title="IEventSink"
xlink:href="#Smp.IEventSink"/>
            </Association>
         </NestedType>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IEventSource.NotSubscribed"
                     Name="NotSubscribed"
                     Visibility="public"
                     Uuid="d55d57b4-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised when trying
to unsubscribe an event sink from an event source that is not
subscribed to it.</Description>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
            <Association
Id="Smp.IEventSource.NotSubscribed.eventSource"
Name="eventSource"
                         Visibility="public">
               <Description>Event source the event sink is not
subscribed to.</Description>
               <Type xlink:title="IEventSource"
xlink:href="#Smp.IEventSource"/>
            </Association>
            <Association
Id="Smp.IEventSource.NotSubscribed.eventSink" Name="eventSink"
                         Visibility="public">
               <Description>Event sink that is not
subscribed.</Description>
               <Type xlink:title="IEventSink"
xlink:href="#Smp.IEventSink"/>
            </Association>
         </NestedType>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IEventSource.InvalidEventSink"
                     Name="InvalidEventSink"
                     Visibility="public"
                     Uuid="d55d57bb-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised when trying
to subscribe an event sink to an event source that has a
different event type.</Description>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
            <Association
Id="Smp.IEventSource.InvalidEventSink.eventSource"
Name="eventSource"
                         Visibility="public">
               <Description>Event source the event sink is
subscribed to.</Description>
               <Type xlink:title="IEventSource"
xlink:href="#Smp.IEventSource"/>
            </Association>
            <Association
Id="Smp.IEventSource.InvalidEventSink.eventSink"
Name="eventSink"
                         Visibility="public">
               <Description>Event sink that is not of valid
type.</Description>
```

```
                <Type xlink:title="IEventSink"
xlink:href="#Smp.IEventSink"/>
            </Association>
        </NestedType>
        <Operation Id="Smp.IEventSource.Subscribe"
Name="Subscribe" Visibility="public">
            <Description>Subscribe to the event source, i.e.
request notifications.</Description>
            <Parameter
Id="Smp.IEventSource.Subscribe.eventSink" Name="eventSink">
                <Description>Event sink to subscribe to event
source.</Description>
                <Type xlink:title="IEventSink"
xlink:href="#Smp.IEventSink"/>
            </Parameter>
            <RaisedException xlink:title="AlreadySubscribed"

xlink:href="#Smp.IEventSource.AlreadySubscribed"/>
            <RaisedException xlink:title="InvalidEventSink"
xlink:href="#Smp.IEventSource.InvalidEventSink"/>
        </Operation>
        <Operation Id="Smp.IEventSource.Unsubscribe"
Name="Unsubscribe" Visibility="public">
            <Description>Unsubscribe from the event source,
i.e. cancel notifications.</Description>
            <Parameter
Id="Smp.IEventSource.Unsubscribe.eventSink" Name="eventSink">
                <Description>Event sink to unsubscribe from
event source.</Description>
                <Type xlink:title="IEventSink"
xlink:href="#Smp.IEventSink"/>
            </Parameter>
            <RaisedException xlink:title="NotSubscribed"
xlink:href="#Smp.IEventSource.NotSubscribed"/>
        </Operation>
        <Base xlink:title="IObject"
xlink:href="#Smp.IObject"/>
    </Type>
    <Type xsi:type="Catalogue:Interface" Id="Smp.IField"
Name="IField"
            Visibility="public"
            Uuid="d57796d2-e618-11dc-ab64-bf8df6d7b83a">
        <Description>Interface of a field.</Description>
        <NestedType xsi:type="Types:Exception"
Id="Smp.IField.InvalidFieldValue"
                     Name="InvalidFieldValue"
                     Visibility="public"
                     Uuid="d579e02c-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised when trying
to assign an illegal value to a field.</Description>
```

```
            <Field Id="Smp.IField.InvalidFieldValue.fieldName"
Name="fieldName"
                   Visibility="public">
               <Description>Fully qualified field name the
value was assigned to.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Field>
            <Field
Id="Smp.IField.InvalidFieldValue.invalidFieldValue"
Name="invalidFieldValue"
                   Visibility="public">
               <Description>Value that was passed as new field
value.</Description>
               <Type xlink:title="AnySimple"
xlink:href="#Smp.AnySimple"/>
            </Field>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </NestedType>
         <Operation Id="Smp.IField.GetView" Name="GetView"
Visibility="public">
            <Description>Return View kind of the
field.</Description>
            <Parameter Id="Smp.IField.GetView.return"
Name="return" Direction="return">
               <Description>The View kind of the field.
The view kind indicates which user roles have visibility of
the field.</Description>
               <Type xlink:title="ViewKind"
xlink:href="#Smp.ViewKind"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IField.IsState" Name="IsState"
Visibility="public">
            <Description>Return State flag of the
field.</Description>
            <Parameter Id="Smp.IField.IsState.return"
Name="return" Direction="return">
               <Description>The State flag of the field.
When true, the state of the field shall be stored by the
simulation infrastructure persistence mechanism on Store(),
and restored on Restore().</Description>
               <Type xlink:title="Bool"
xlink:href="#Smp.Bool"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IField.IsInput" Name="IsInput"
Visibility="public">
            <Description>Return Input flag of the
field.</Description>
```

```
            <Parameter Id="Smp.IField.IsInput.return"
Name="return" Direction="return">
               <Description>The Input flag of the field.
When true, the field is considered an input into the model and
can be used as target of a field link in data flow based
design.</Description>
               <Type xlink:title="Bool"
xlink:href="#Smp.Bool"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IField.IsOutput" Name="IsOutput"
Visibility="public">
            <Description>Return Output flag of the
field.</Description>
            <Parameter Id="Smp.IField.IsOutput.return"
Name="return" Direction="return">
               <Description>The Output flag of the field.
When true, the field is considered an output of the model and
can be used as source of a field link in data flow based
design.</Description>
               <Type xlink:title="Bool"
xlink:href="#Smp.Bool"/>
            </Parameter>
         </Operation>
         <Base xlink:title="IObject"
xlink:href="#Smp.IObject"/>
      </Type>
      <Type xsi:type="Types:NativeType"
Id="Smp.EventSourceCollection"
            Name="EventSourceCollection"
            Visibility="public"
            Uuid="d55d57c2-e618-11dc-ab64-bf8df6d7b83a">
         <Description>An event source collection is an ordered
collection of event sources, which allows iterating all
members.
</Description>
         <Platform Name="cpp"
Type="vector&lt;IEventSource*&gt;" Namespace="std"
Location="vector"/>
      </Type>
      <Type xsi:type="Catalogue:Interface"
Id="Smp.IDynamicInvocation"
            Name="IDynamicInvocation"
            Visibility="public"
            Uuid="d56212a9-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface for a component that supports
dynamic invocation of operations.</Description>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IDynamicInvocation.InvalidParameterCount"
                     Name="InvalidParameterCount"
                     Visibility="public"
                     Uuid="d56212c0-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised by the
Invoke() method when trying to invoke a method with a wrong
number of parameters.</Description>
            <Field
Id="Smp.IDynamicInvocation.InvalidParameterCount.operationName
"
                   Name="operationName"
                   Visibility="public">
               <Description>Operation name of request passed
to the Invoke() method.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Field>
            <Field
Id="Smp.IDynamicInvocation.InvalidParameterCount.operationPara
meters"
                   Name="operationParameters"
                   Visibility="public">
               <Description>Correct number of parameters of
operation.</Description>
               <Type xlink:title="Int32"
xlink:href="#Smp.Int32"/>
            </Field>
            <Field
Id="Smp.IDynamicInvocation.InvalidParameterCount.requestParame
ters"
                   Name="requestParameters"
                   Visibility="public">
               <Description>Wrong number of parameters of
operation.</Description>
               <Type xlink:title="Int32"
xlink:href="#Smp.Int32"/>
            </Field>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </NestedType>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IDynamicInvocation.InvalidParameterType"
                     Name="InvalidParameterType"
                     Visibility="public"
                     Uuid="d56212ca-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised by the
Invoke() method when trying to invoke a method passing a
parameter of wrong type.</Description>
            <Field
Id="Smp.IDynamicInvocation.InvalidParameterType.operationName"
                   Name="operationName"
                   Visibility="public">
```

```
                <Description>Operation name of request passed
to the Invoke() method.</Description>
                <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Field>
            <Field
Id="Smp.IDynamicInvocation.InvalidParameterType.parameterName"
                   Name="parameterName"
                   Visibility="public">
                <Description>Name of parameter of wrong
type.</Description>
                <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Field>
            <Field
Id="Smp.IDynamicInvocation.InvalidParameterType.invalidType"
Name="invalidType"
                   Visibility="public">
                <Description>Type that is not
valid.</Description>
                <Type xlink:title="PrimitiveTypeKind"
xlink:href="#Smp.PrimitiveTypeKind"/>
            </Field>
            <Field
Id="Smp.IDynamicInvocation.InvalidParameterType.expectedType"
                   Name="expectedType"
                   Visibility="public">
                <Description>Type that was
expected.</Description>
                <Type xlink:title="PrimitiveTypeKind"
xlink:href="#Smp.PrimitiveTypeKind"/>
            </Field>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
        </NestedType>
        <NestedType xsi:type="Types:Exception"
Id="Smp.IDynamicInvocation.InvalidOperationName"
                     Name="InvalidOperationName"
                     Visibility="public"
                     Uuid="d56483d6-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised by the
Invoke() method when trying to invoke a method that does not
exist, or that does not support dynamic
invocation.</Description>
            <Field
Id="Smp.IDynamicInvocation.InvalidOperationName.operationName"
                   Name="operationName"
                   Visibility="public">
                <Description>Operation name of request passed
to the Invoke() method.</Description>
```

```
                <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Field>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </NestedType>
         <Operation Id="Smp.IDynamicInvocation.CreateRequest"
Name="CreateRequest"
                    Visibility="public">
            <Description>Return a request object for the given
operation that describes the parameters and the return
value.</Description>
            <Parameter
Id="Smp.IDynamicInvocation.CreateRequest.return" Name="return"
                       Direction="return">
               <Description>Request object for operation, or
null if either no operation with the given name could be
found, or the operation with the given name does not support
dynamic invocation.</Description>
               <Type xlink:title="IRequest"
xlink:href="#Smp.IRequest"/>
            </Parameter>
            <Parameter
Id="Smp.IDynamicInvocation.CreateRequest.operationName"
Name="operationName">
               <Description>Name of operation.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IDynamicInvocation.Invoke"
Name="Invoke" Visibility="public">
            <Description>Dynamically invoke an operation using
a request object that has been created and filled with
parameter values by the caller.</Description>
            <Parameter
Id="Smp.IDynamicInvocation.Invoke.request" Name="request">
               <Description>Request object to
invoke.</Description>
               <Type xlink:title="IRequest"
xlink:href="#Smp.IRequest"/>
            </Parameter>
            <RaisedException
xlink:title="InvalidParameterCount"

xlink:href="#Smp.IDynamicInvocation.InvalidParameterCount"/>
            <RaisedException
xlink:title="InvalidOperationName"

xlink:href="#Smp.IDynamicInvocation.InvalidOperationName"/>
```

```
            <RaisedException
xlink:title="InvalidParameterType"

xlink:href="#Smp.IDynamicInvocation.InvalidParameterType"/>
         </Operation>
         <Operation Id="Smp.IDynamicInvocation.DeleteRequest"
Name="DeleteRequest"
                    Visibility="public">
            <Description>Destroy a request object that has
been created with the CreateRequest() method
before.</Description>
            <Parameter
Id="Smp.IDynamicInvocation.DeleteRequest.request"
Name="request">
               <Description>Request object to
destroy.</Description>
               <Type xlink:title="IRequest"
xlink:href="#Smp.IRequest"/>
            </Parameter>
         </Operation>
         <Base xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
      </Type>
      <Type xsi:type="Catalogue:Interface"
Id="Smp.IArrayField" Name="IArrayField"
            Visibility="public"
            Uuid="d57796e5-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface of a field which is an array
of a simple type.</Description>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IArrayField.InvalidArrayValue"
                     Name="InvalidArrayValue"
                     Visibility="public"
                     Uuid="d5779709-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised when trying
to assign an illegal value to an array field.</Description>
            <Field
Id="Smp.IArrayField.InvalidArrayValue.fieldName"
Name="fieldName"
                   Visibility="public">
               <Description>Fully qualified field name the
value was assigned to.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Field>
            <Field
Id="Smp.IArrayField.InvalidArrayValue.invalidValueIndex"
                   Name="invalidValueIndex"
                   Visibility="public">
               <Description>Index in array where the first
invalid value was found.</Description>
               <Type xlink:title="Int32"
xlink:href="#Smp.Int32"/>
            </Field>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </NestedType>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IArrayField.InvalidArraySize"
                     Name="InvalidArraySize"
                     Visibility="public"
                     Uuid="d5779710-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised when an
invalid array size is specified.</Description>
            <Field
Id="Smp.IArrayField.InvalidArraySize.fieldName"
Name="fieldName"
                   Visibility="public">
               <Description>Name of field that has been
accessed.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Field>
            <Field
Id="Smp.IArrayField.InvalidArraySize.invalidSize"
Name="invalidSize"
                   Visibility="public">
               <Description>Invalid array size.</Description>
               <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
            </Field>
            <Field
Id="Smp.IArrayField.InvalidArraySize.arraySize"
Name="arraySize"
                   Visibility="public">
               <Description>Real array size.</Description>
               <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
            </Field>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </NestedType>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IArrayField.InvalidIndex"
                     Name="InvalidIndex"
                     Visibility="public"
                     Uuid="d5779719-e618-11dc-ab64-
bf8df6d7b83a">
```

```
            <Description>This exception is raised when an
invalid index is specified.</Description>
            <Field Id="Smp.IArrayField.InvalidIndex.fieldName"
Name="fieldName"
                   Visibility="public">
               <Description>Fully qualified field name the
value was assigned to.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Field>
            <Field Id="Smp.IArrayField.InvalidIndex.arraySize"
Name="arraySize"
                   Visibility="public">
               <Description>Real array size.</Description>
               <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
            </Field>
            <Field
Id="Smp.IArrayField.InvalidIndex.invalidIndex"
Name="invalidIndex"
                   Visibility="public">
               <Description>Invalid Index.</Description>
               <Type xlink:title="Int64"
xlink:href="#Smp.Int64"/>
            </Field>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </NestedType>
         <Operation Id="Smp.IArrayField.GetValue"
Name="GetValue" Visibility="public">
            <Description>Get a value from a specific index of
the array field.</Description>
            <Parameter Id="Smp.IArrayField.GetValue.return"
Name="return" Direction="return">
               <Description>Value from given
index.</Description>
               <Type xlink:title="AnySimple"
xlink:href="#Smp.AnySimple"/>
            </Parameter>
            <Parameter Id="Smp.IArrayField.GetValue.index"
Name="index">
               <Description>Index of value to
get.</Description>
               <Type xlink:title="UInt64"
xlink:href="#Smp.UInt64"/>
            </Parameter>
            <RaisedException xlink:title="InvalidIndex"
xlink:href="#Smp.IArrayField.InvalidIndex"/>
         </Operation>
         <Operation Id="Smp.IArrayField.SetValue"
Name="SetValue" Visibility="public">
```

```
            <Description>Set a value at given index of the
array field.</Description>
            <Parameter Id="Smp.IArrayField.SetValue.index"
Name="index">
               <Description>Index of value to
set.</Description>
               <Type xlink:title="UInt64"
xlink:href="#Smp.UInt64"/>
            </Parameter>
            <Parameter Id="Smp.IArrayField.SetValue.value"
Name="value">
               <Description>Value to set at given
index.</Description>
               <Type xlink:title="AnySimple"
xlink:href="#Smp.AnySimple"/>
            </Parameter>
            <RaisedException xlink:title="InvalidFieldValue"
xlink:href="#Smp.IField.InvalidFieldValue"/>
            <RaisedException xlink:title="InvalidIndex"
xlink:href="#Smp.IArrayField.InvalidIndex"/>
         </Operation>
         <Operation Id="Smp.IArrayField.GetValues"
Name="GetValues" Visibility="public">
            <Description>Get all values of the array
field.</Description>
            <Parameter Id="Smp.IArrayField.GetValues.length"
Name="length">
               <Description>Size of given values
array.</Description>
               <Type xlink:title="UInt64"
xlink:href="#Smp.UInt64"/>
            </Parameter>
            <Parameter Id="Smp.IArrayField.GetValues.values"
Name="values" Direction="inout">
               <Description>Pre-allocated array of values to
store result to.</Description>
               <Type xlink:title="AnySimpleArray"
xlink:href="#Smp.AnySimpleArray"/>
            </Parameter>
            <RaisedException xlink:title="InvalidArraySize"
xlink:href="#Smp.IArrayField.InvalidArraySize"/>
         </Operation>
         <Operation Id="Smp.IArrayField.SetValues"
Name="SetValues" Visibility="public">
            <Description>Set all values of the array
field.</Description>
            <Parameter Id="Smp.IArrayField.SetValues.length"
Name="length">
               <Description>Size of given values
array.</Description>
```

```
               <Type xlink:title="UInt64"
xlink:href="#Smp.UInt64"/>
            </Parameter>
            <Parameter Id="Smp.IArrayField.SetValues.values"
Name="values">
               <Description>Array of values to store in array
field.</Description>
               <Type xlink:title="AnySimpleArray"
xlink:href="#Smp.AnySimpleArray"/>
            </Parameter>
            <RaisedException xlink:title="InvalidArraySize"
xlink:href="#Smp.IArrayField.InvalidArraySize"/>
            <RaisedException xlink:title="InvalidArrayValue"
xlink:href="#Smp.IArrayField.InvalidArrayValue"/>
         </Operation>
         <Operation Id="Smp.IArrayField.GetSize"
Name="GetSize" Visibility="public">
            <Description>Get the size (number of array items)
of the field.</Description>
            <Parameter Id="Smp.IArrayField.GetSize.return"
Name="return" Direction="return">
               <Description>Size (number of array items) of
the field.</Description>
               <Type xlink:title="UInt64"
xlink:href="#Smp.UInt64"/>
            </Parameter>
         </Operation>
         <Base xlink:title="IField" xlink:href="#Smp.IField"/>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.IRequest"
Name="IRequest"
            Visibility="public"
            Uuid="d55fc872-e618-11dc-ab64-bf8df6d7b83a">
         <Description>A request holds information, which is
passed between a client invoking an operation via the
IDynamicInvocation interface and a component being invoked.
</Description>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IRequest.InvalidParameterIndex"
                     Name="InvalidParameterIndex"
                     Visibility="public"
                     Uuid="d562128f-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised when using
an invalid parameter index to set (SetParameterValue()) or get
(GetParameterValue()) a parameter value of an operation in a
request.</Description>
            <Field
Id="Smp.IRequest.InvalidParameterIndex.operationName"
Name="operationName"
                   Visibility="public">
               <Description>Name of operation.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Field>
            <Field
Id="Smp.IRequest.InvalidParameterIndex.parameterIndex"
Name="parameterIndex"
                   Visibility="public">
               <Description>Invalid parameter index
used.</Description>
               <Type xlink:title="Int32"
xlink:href="#Smp.Int32"/>
            </Field>
            <Field
Id="Smp.IRequest.InvalidParameterIndex.parameterCount"
Name="parameterCount"
                   Visibility="public">
               <Description>Number of parameters of the
operation.</Description>
               <Type xlink:title="Int32"
xlink:href="#Smp.Int32"/>
            </Field>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </NestedType>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IRequest.InvalidParameterValue"
                     Name="InvalidParameterValue"
                     Visibility="public"
                     Uuid="d5621296-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised when trying
to assign an illegal value to a parameter of an operation in a
request using SetParameterValue().</Description>
            <Field
Id="Smp.IRequest.InvalidParameterValue.parameterName"
Name="parameterName"
                   Visibility="public">
               <Description>Name of parameter value was
assigned to.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Field>
            <Field
Id="Smp.IRequest.InvalidParameterValue.value" Name="value"
Visibility="public">
               <Description>Value that was passed as
parameter.</Description>
               <Type xlink:title="AnySimple"
xlink:href="#Smp.AnySimple"/>
            </Field>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </NestedType>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IRequest.InvalidReturnValue"
                     Name="InvalidReturnValue"
                     Visibility="public"
                     Uuid="d562129d-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised when trying
to assign an invalid return value of an operation in a request
using SetReturnValue().</Description>
            <Field
Id="Smp.IRequest.InvalidReturnValue.operationName"
Name="operationName"
                   Visibility="public">
               <Description>Name of operation the return value
was assigned to.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Field>
            <Field Id="Smp.IRequest.InvalidReturnValue.value"
Name="value" Visibility="public">
               <Description>Value that was passed as return
value.</Description>
               <Type xlink:title="AnySimple"
xlink:href="#Smp.AnySimple"/>
            </Field>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </NestedType>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IRequest.VoidOperation" Name="VoidOperation"
                     Visibility="public"
                     Uuid="d56212a4-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised when trying
to read (GetReturnValue()) or write (SetReturnValue()) the
return value of a void operation.</Description>
            <Field
Id="Smp.IRequest.VoidOperation.operationName"
Name="operationName"
                   Visibility="public">
               <Description>Name of operation.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Field>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </NestedType>
         <Operation Id="Smp.IRequest.GetOperationName"
Name="GetOperationName" Visibility="public">
            <Description>Return the name of the operation that
this request is for.</Description>
            <Parameter
Id="Smp.IRequest.GetOperationName.return" Name="return"
Direction="return">
               <Description>Name of the
operation.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IRequest.GetParameterCount"
Name="GetParameterCount"
                    Visibility="public">
            <Description>Return the number of parameters
stored in the request.</Description>
            <Parameter
Id="Smp.IRequest.GetParameterCount.return" Name="return"
Direction="return">
               <Description>Number of parameters in request
object.</Description>
               <Type xlink:title="Int32"
xlink:href="#Smp.Int32"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IRequest.GetParameterIndex"
Name="GetParameterIndex"
                    Visibility="public">
            <Description>Query for a parameter index by
parameter name.</Description>
            <Parameter
Id="Smp.IRequest.GetParameterIndex.return" Name="return"
Direction="return">
               <Description>Index of parameter with the given
name, or -1 if no parameter with the given name could be
found.</Description>
               <Type xlink:title="Int32"
xlink:href="#Smp.Int32"/>
            </Parameter>
            <Parameter
Id="Smp.IRequest.GetParameterIndex.parameterName"
Name="parameterName">
               <Description>Name of parameter.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IRequest.SetParameterValue"
Name="SetParameterValue"
```

```
                    Visibility="public">
            <Description>Assign a value to a parameter at a
given position.</Description>
            <Parameter
Id="Smp.IRequest.SetParameterValue.index" Name="index">
               <Description>Index of parameter (0-
based).</Description>
               <Type xlink:title="Int32"
xlink:href="#Smp.Int32"/>
            </Parameter>
            <Parameter
Id="Smp.IRequest.SetParameterValue.value" Name="value">
               <Description>Value of parameter.</Description>
               <Type xlink:title="AnySimple"
xlink:href="#Smp.AnySimple"/>
            </Parameter>
            <RaisedException
xlink:title="InvalidParameterIndex"

xlink:href="#Smp.IRequest.InvalidParameterIndex"/>
            <RaisedException
xlink:title="InvalidParameterValue"

xlink:href="#Smp.IRequest.InvalidParameterValue"/>
            <RaisedException xlink:title="InvalidAnyType"
xlink:href="#Smp.InvalidAnyType"/>
         </Operation>
         <Operation Id="Smp.IRequest.GetParameterValue"
Name="GetParameterValue"
                    Visibility="public">
            <Description>Query a value of a parameter at a
given position.</Description>
            <Parameter
Id="Smp.IRequest.GetParameterValue.return" Name="return"
Direction="return">
               <Description>Value of parameter. </Description>
               <Type xlink:title="AnySimple"
xlink:href="#Smp.AnySimple"/>
            </Parameter>
            <Parameter
Id="Smp.IRequest.GetParameterValue.index" Name="index">
               <Description>Index of parameter (0-
based).</Description>
               <Type xlink:title="Int32"
xlink:href="#Smp.Int32"/>
            </Parameter>
            <RaisedException
xlink:title="InvalidParameterIndex"

xlink:href="#Smp.IRequest.InvalidParameterIndex"/>
         </Operation>
         <Operation Id="Smp.IRequest.SetReturnValue"
Name="SetReturnValue" Visibility="public">
            <Description>Assign the return value of the
operation.</Description>
            <Parameter Id="Smp.IRequest.SetReturnValue.value"
Name="value">
               <Description>Return value.</Description>
               <Type xlink:title="AnySimple"
xlink:href="#Smp.AnySimple"/>
            </Parameter>
            <RaisedException xlink:title="InvalidReturnValue"
xlink:href="#Smp.IRequest.InvalidReturnValue"/>
            <RaisedException xlink:title="VoidOperation"
xlink:href="#Smp.IRequest.VoidOperation"/>
            <RaisedException xlink:title="InvalidAnyType"
xlink:href="#Smp.InvalidAnyType"/>
         </Operation>
         <Operation Id="Smp.IRequest.GetReturnValue"
Name="GetReturnValue" Visibility="public">
            <Description>Query the return value of the
operation.</Description>
            <Parameter Id="Smp.IRequest.GetReturnValue.return"
Name="return" Direction="return">
               <Description>Return value of the
operation.</Description>
               <Type xlink:title="AnySimple"
xlink:href="#Smp.AnySimple"/>
            </Parameter>
            <RaisedException xlink:title="VoidOperation"
xlink:href="#Smp.IRequest.VoidOperation"/>
         </Operation>
      </Type>
      <Type xsi:type="Catalogue:Interface"
Id="Smp.ISimpleField" Name="ISimpleField"
            Visibility="public"
            Uuid="d579e020-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface of a field of simple
type.</Description>
         <Operation Id="Smp.ISimpleField.GetValue"
Name="GetValue" Visibility="public">
            <Description>Get the value of the simple
field.</Description>
            <Parameter Id="Smp.ISimpleField.GetValue.return"
Name="return" Direction="return">
               <Description>Field value.</Description>
               <Type xlink:title="AnySimple"
xlink:href="#Smp.AnySimple"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.ISimpleField.SetValue"
Name="SetValue" Visibility="public">
```

```
            <Description>Set the value of the simple
field.</Description>
            <Parameter Id="Smp.ISimpleField.SetValue.value"
Name="value">
               <Description>Field value.</Description>
               <Type xlink:title="AnySimple"
xlink:href="#Smp.AnySimple"/>
            </Parameter>
            <RaisedException xlink:title="InvalidFieldValue"
xlink:href="#Smp.IField.InvalidFieldValue"/>
         </Operation>
         <Base xlink:title="IField" xlink:href="#Smp.IField"/>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.IPersist"
Name="IPersist"
            Visibility="public"
            Uuid="d55fc858-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface of a self-persisting component
that provides operations to allow for storing and restoring
its state.</Description>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IPersist.CannotRestore" Name="CannotRestore"
                     Visibility="public"
                     Uuid="d55fc866-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised when the
content of the storage reader passed to the Restore() method
contains invalid data. </Description>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </NestedType>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IPersist.CannotStore" Name="CannotStore"
                     Visibility="public"
                     Uuid="d55fc86c-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised when the
component cannot store its data to the storage writer given to
the Store() method.</Description>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </NestedType>
         <Operation Id="Smp.IPersist.Restore" Name="Restore"
Visibility="public">
            <Description>Restore component state from
storage.</Description>
            <Parameter Id="Smp.IPersist.Restore.reader"
Name="reader">
               <Description>Interface that allows reading from
storage.</Description>
```

```
               <Type xlink:title="IStorageReader"
xlink:href="#Smp.IStorageReader"/>
            </Parameter>
            <RaisedException xlink:title="CannotRestore"
xlink:href="#Smp.IPersist.CannotRestore"/>
         </Operation>
         <Operation Id="Smp.IPersist.Store" Name="Store"
Visibility="public">
            <Description>Store component state to
storage.</Description>
            <Parameter Id="Smp.IPersist.Store.writer"
Name="writer">
               <Description>Interface that allows writing to
storage.</Description>
               <Type xlink:title="IStorageWriter"
xlink:href="#Smp.IStorageWriter"/>
            </Parameter>
            <RaisedException xlink:title="CannotStore"
xlink:href="#Smp.IPersist.CannotStore"/>
         </Operation>
         <Base xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
      </Type>
      <Type xsi:type="Catalogue:Interface"
Id="Smp.IForcibleField" Name="IForcibleField"
            Visibility="public"
            Uuid="d5779722-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface of a forcible
field.</Description>
         <Operation Id="Smp.IForcibleField.Force" Name="Force"
Visibility="public">
            <Description>Force field to given
value.</Description>
            <Parameter Id="Smp.IForcibleField.Force.value"
Name="value">
               <Description>Value to force field
to.</Description>
               <Type xlink:title="AnySimple"
xlink:href="#Smp.AnySimple"/>
            </Parameter>
            <RaisedException xlink:title="InvalidFieldValue"
xlink:href="#Smp.IField.InvalidFieldValue"/>
         </Operation>
         <Operation Id="Smp.IForcibleField.Unforce"
Name="Unforce" Visibility="public">
            <Description>Unforce field.</Description>
         </Operation>
         <Operation Id="Smp.IForcibleField.IsForced"
Name="IsForced" Visibility="public">
            <Description>Query for the forced state of the
field.</Description>
```

```
            <Parameter Id="Smp.IForcibleField.IsForced.return"
Name="return" Direction="return">
               <Description>Whether the field is forced or
not.</Description>
               <Type xlink:title="Bool"
xlink:href="#Smp.Bool"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IForcibleField.Freeze"
Name="Freeze" Visibility="public">
            <Description>Force field to its current
value.</Description>
         </Operation>
         <Base xlink:title="ISimpleField"
xlink:href="#Smp.ISimpleField"/>
      </Type>
      <Type xsi:type="Catalogue:Interface"
Id="Smp.IStorageReader" Name="IStorageReader"
            Visibility="public"
            Uuid="d55fc850-e618-11dc-ab64-bf8df6d7b83a">
         <Description>This interface provides functionality to
read data from storage.</Description>
      </Type>
      <Type xsi:type="Catalogue:Interface"
Id="Smp.IStorageWriter" Name="IStorageWriter"
            Visibility="public"
            Uuid="d55fc854-e618-11dc-ab64-bf8df6d7b83a">
         <Description>This interface provides functionality to
write data to storage.</Description>
      </Type>
      <Type xsi:type="Catalogue:Interface"
Id="Smp.IEntryPoint" Name="IEntryPoint"
            Visibility="public"
            Uuid="d55b0e86-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface of an entry
point.</Description>
         <Operation Id="Smp.IEntryPoint.GetOwner"
Name="GetOwner" Visibility="public">
            <Description>This method returns the Component
that owns the entry point.</Description>
            <Parameter Id="Smp.IEntryPoint.GetOwner.return"
Name="return" Direction="return">
               <Description>Owner of entry
point.</Description>
               <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IEntryPoint.Execute"
Name="Execute" Visibility="public">
```

```
            <Description>This method is called when an
associated event is emitted.</Description>
         </Operation>
         <Base xlink:title="IObject"
xlink:href="#Smp.IObject"/>
      </Type>
      <Type xsi:type="Types:Enumeration"
Id="Smp.SimulatorStateKind"
            Name="SimulatorStateKind"
            Visibility="public"
            Uuid="d56483dc-e618-11dc-ab64-bf8df6d7b83a">
         <Description>This is an enumeration of the available
states of the simulator. The Setup phase is split into three
different states, the Execution phase has five different
states, and the Termination phase has two
states.</Description>
         <Literal Id="Smp.SimulatorStateKind.SSK_Building"
Name="SSK_Building" Value="0">
            <Description>In Building state, the model
hierarchy is created. This is done by an external component,
not by the simulator.
This state is entered automatically after the simulation
environment has performed its initialisation.
This state is left with the Connect() state transition
method.</Description>
         </Literal>
         <Literal Id="Smp.SimulatorStateKind.SSK_Connecting"
Name="SSK_Connecting" Value="1">
            <Description>In Connecting state, the simulation
environment traverses the model hierarchy and calls the
Connect() method of each model.
This state is entered using the Connect() state transition.
After connecting all models to the simulator, an automatic
state transition to the Initialising state is performed.
</Description>
         </Literal>
         <Literal Id="Smp.SimulatorStateKind.SSK_Initialising"
Name="SSK_Initialising" Value="2">
            <Description>In Initialising state, the simulation
environment executes all initialisation entry points in the
order they have been added to the simulator using the
AddInitEntryPoint() method.
This state is either entered automatically after the
simulation environment has connected all models to the
simulator, or manually from Standby state using the
Initialise() state transition.
After calling all initialisation entry points, an automatic
state transition to the Standby state is performed.
</Description>
         </Literal>
         <Literal Id="Smp.SimulatorStateKind.SSK_Standby"
Name="SSK_Standby" Value="3">
            <Description>In Standby state, the simulation
environment (namely the Time Keeper Service) does not progress
simulation time. Only entry points registered relative to Zulu
time are executed.
This state is entered automatically from the Initialising,
Storing, and Restoring states, or manually from the Executing
state using the Hold() state transition.
This state is left with one of the Run(), Store(), Restore(),
Initialise(), Reconnect() or Exit() state
transitions.</Description>
         </Literal>
         <Literal Id="Smp.SimulatorStateKind.SSK_Executing"
Name="SSK_Executing" Value="4">
            <Description>In Executing state, the simulation
environment (namely the Time Keeper Service) does progress
simulation time. Entry points registered with any of the
available time kinds are executed.
This state is entered using the Run() state transition.
This state is left using the Hold() state
transition.</Description>
         </Literal>
         <Literal Id="Smp.SimulatorStateKind.SSK_Storing"
Name="SSK_Storing" Value="5">
            <Description>In Storing state, the simulation
environment first stores the values of all fields published
with the State attribute to storage (typically a file).
Afterwards, the Store() method of all components (Models and
Services) implementing the optional IPersist interface is
called, to allow custom storing of additional information.
While in this state, fields published with the State attribute
must not be modified by the models, to ensure that a
consistent set of field values is stored.
This state is entered using the Store() state transition.
After storing the simulator state, an automatic state
transition to the Standby state is performed. </Description>
         </Literal>
         <Literal Id="Smp.SimulatorStateKind.SSK_Restoring"
Name="SSK_Restoring" Value="6">
            <Description>In Restoring state, the simulation
environment first restores the values of all fields published
with the State attribute from storage. Afterwards, the
Restore() method of all components implementing the optional
IPersist interface is called, to allow custom restoring of
additional information. While in this state, fields published
with the State attribute must not be modified by the models,
to ensure that a consistent set of field values is restored.
This state is entered using the Restore() state transition.
After restoring the simulator state, an automatic state
transition to the Standby state is performed. </Description>
```

```
         </Literal>
         <Literal Id="Smp.SimulatorStateKind.SSK_Reconnecting"
Name="SSK_Reconnecting" Value="7">
            <Description>In Reconnecting state, the simulation
environment makes sure that models that have been added to the
simulator after leaving the Building state are properly
published, configured and connected.
This state is entered using the Reconnect() state transition.
After connecting all new models, an automatic state transition
to the Standby state is performed.</Description>
         </Literal>
         <Literal Id="Smp.SimulatorStateKind.SSK_Exiting"
Name="SSK_Exiting" Value="8">
            <Description>In Exiting state, the simulation
environment is properly terminating a running simulation.
This state is entered using the Exit() state transition. After
exiting, the simulator is in an undefined state.
</Description>
         </Literal>
         <Literal Id="Smp.SimulatorStateKind.SSK_Aborting"
Name="SSK_Aborting" Value="9">
            <Description>In this state, the simulation
environment performs an abnormal simulation shut-down.
This state is entered using the Abort() state transition.
After aborting, the simulator is in an undefined state.
</Description>
         </Literal>
      </Type>
      <Type xsi:type="Types:NativeType"
Id="Smp.EntryPointCollection"
            Name="EntryPointCollection"
            Visibility="public"
            Uuid="d55b0e92-e618-11dc-ab64-bf8df6d7b83a">
         <Description>An entry point collection is an ordered
collection of entry points, which allows iterating all
members.</Description>
         <Platform Name="cpp"
Type="vector&lt;IEntryPoint*&gt;" Namespace="std"
Location="vector"/>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.ISimulator"
Name="ISimulator"
            Visibility="public"
            Uuid="d566f4f2-e618-11dc-ab64-bf8df6d7b83a">
         <Description>This interface gives access to the
simulation environment state and state transitions. Further,
it provides convenience methods to add models, and to add and
retrieve simulation services.</Description>
         <Constant Id="Smp.ISimulator.SMP_SimulatorModels"
Name="SMP_SimulatorModels"
                   Visibility="public">
            <Description>Name of the model
container.</Description>
            <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            <Value xsi:type="Types:String8Value"
Value="Models"/>
         </Constant>
         <Constant Id="Smp.ISimulator.SMP_SimulatorServices"
Name="SMP_SimulatorServices"
                   Visibility="public">
            <Description>Name of the service
container.</Description>
            <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            <Value xsi:type="Types:String8Value"
Value="Services"/>
         </Constant>
         <Operation Id="Smp.ISimulator.Initialise"
Name="Initialise" Visibility="public">
            <Description>This method asks the simulation
environment to call all initialisation entry points
again.</Description>
         </Operation>
         <Operation Id="Smp.ISimulator.Publish" Name="Publish"
Visibility="public">
            <Description>This method asks the simulation
environment to call the Publish() method of all model
instances in the model hierarchy which are still in Created
state.
</Description>
         </Operation>
         <Operation Id="Smp.ISimulator.Configure"
Name="Configure" Visibility="public">
            <Description>This method asks the simulation
environment to call the Configure() method of all model
instances which are still in Publishing state.
</Description>
         </Operation>
         <Operation Id="Smp.ISimulator.Connect" Name="Connect"
Visibility="public">
            <Description>This method informs the simulation
environment that the hierarchy of model instances has been
configured, and can now be connected to the simulator. Thus,
the simulation environment calls the Connect() method of all
model instances.</Description>
         </Operation>
         <Operation Id="Smp.ISimulator.Hold" Name="Hold"
Visibility="public">
            <Description>This method changes from Executing to
Standby state.</Description>
         </Operation>
         <Operation Id="Smp.ISimulator.Store" Name="Store"
Visibility="public">
            <Description>This method is used to store a state
vector to file.</Description>
            <Parameter Id="Smp.ISimulator.Store.filename"
Name="filename">
               <Description>Name to use for simulation state
vector file.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.ISimulator.Restore" Name="Restore"
Visibility="public">
            <Description>This method is used to restore a
state vector from file.</Description>
            <Parameter Id="Smp.ISimulator.Restore.filename"
Name="filename">
               <Description>Name of simulation state vector
file.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.ISimulator.Reconnect"
Name="Reconnect" Visibility="public">
            <Description>This method asks the simulation
environment to reconnect the component hierarchy starting at
the given root component.</Description>
            <Parameter Id="Smp.ISimulator.Reconnect.root"
Name="root">
               <Description>Root component to start
reconnecting from. This can be the parent component of a newly
added model, or e.g. the simulator itself.</Description>
               <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.ISimulator.Run" Name="Run"
Visibility="public">
            <Description>This method changes from Standby to
Executing state.</Description>
         </Operation>
         <Operation Id="Smp.ISimulator.Exit" Name="Exit"
Visibility="public">
            <Description>This method is used for a normal
termination of a simulation.</Description>
         </Operation>
         <Operation Id="Smp.ISimulator.Abort" Name="Abort"
Visibility="public">
```

```
            <Description>This method is used for an abnormal
termination of a simulation.</Description>
         </Operation>
         <Operation Id="Smp.ISimulator.GetState"
Name="GetState" Visibility="public">
            <Description>Return the current simulator
state.</Description>
            <Parameter Id="Smp.ISimulator.GetState.return"
Name="return" Direction="return">
               <Description>Current simulator
state.</Description>
               <Type xlink:title="SimulatorStateKind"
xlink:href="#Smp.SimulatorStateKind"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.ISimulator.AddInitEntryPoint"
Name="AddInitEntryPoint"
                    Visibility="public">
            <Description>This method can be used to add entry
points that shall be executed in the Initialising
state.</Description>
            <Parameter
Id="Smp.ISimulator.AddInitEntryPoint.entryPoint"
Name="entryPoint">
               <Description>Entry point to execute in
Initialising state.</Description>
               <Type xlink:title="IEntryPoint"
xlink:href="#Smp.IEntryPoint"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.ISimulator.AddModel"
Name="AddModel" Visibility="public">
            <Description>This method adds a model to the
models collection of the simulator, i.e. to the "Models"
container.</Description>
            <Parameter Id="Smp.ISimulator.AddModel.model"
Name="model">
               <Description>New model to add to collection of
root models, i.e. to the "Models" container.</Description>
               <Type xlink:title="IModel"
xlink:href="#Smp.IModel"/>
            </Parameter>
            <RaisedException xlink:title="DuplicateName"
xlink:href="#Smp.DuplicateName"/>
         </Operation>
         <Operation Id="Smp.ISimulator.AddService"
Name="AddService" Visibility="public">
            <Description>This method adds a user-defined
service to the services collection, i.e. to the "Services"
container.</Description>
```

```
            <Parameter Id="Smp.ISimulator.AddService.service"
Name="service">
               <Description>Service to add to services
collection.</Description>
               <Type xlink:title="IService"
xlink:href="#Smp.IService"/>
            </Parameter>
            <RaisedException xlink:title="DuplicateName"
xlink:href="#Smp.DuplicateName"/>
         </Operation>
         <Operation Id="Smp.ISimulator.GetService"
Name="GetService" Visibility="public">
            <Description>Query for a service component by its
name.</Description>
            <Parameter Id="Smp.ISimulator.GetService.return"
Name="return" Direction="return">
               <Description>Service with the given name, or
null if no service with the given name could be
found.</Description>
               <Type xlink:title="IService"
xlink:href="#Smp.IService"/>
            </Parameter>
            <Parameter Id="Smp.ISimulator.GetService.name"
Name="name">
               <Description>Service name.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.ISimulator.GetLogger"
Name="GetLogger" Visibility="public">
            <Description>Return interface to logger service.
</Description>
            <Parameter Id="Smp.ISimulator.GetLogger.return"
Name="return" Direction="return">
               <Description>Interface to mandatory logger
service.</Description>
               <Type xlink:title="ILogger"
xlink:href="#Smp.Services.ILogger"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.ISimulator.GetScheduler"
Name="GetScheduler" Visibility="public">
            <Description>Return interface to scheduler
service.</Description>
            <Parameter Id="Smp.ISimulator.GetScheduler.return"
Name="return" Direction="return">
               <Description>Interface to mandatory scheduler
service.</Description>
               <Type xlink:title="IScheduler"
xlink:href="#Smp.Services.IScheduler"/>
```

```
            </Parameter>
         </Operation>
         <Operation Id="Smp.ISimulator.GetTimeKeeper"
Name="GetTimeKeeper" Visibility="public">
            <Description>Return interface to time keeper
service.</Description>
            <Parameter
Id="Smp.ISimulator.GetTimeKeeper.return" Name="return"
Direction="return">
               <Description>Interface to mandatory time keeper
service.</Description>
               <Type xlink:title="ITimeKeeper"
xlink:href="#Smp.Services.ITimeKeeper"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.ISimulator.GetEventManager"
Name="GetEventManager" Visibility="public">
            <Description>Return interface to event manager
service.</Description>
            <Parameter
Id="Smp.ISimulator.GetEventManager.return" Name="return"
Direction="return">
               <Description>Interface to mandatory event
manager service.</Description>
               <Type xlink:title="IEventManager"
xlink:href="#Smp.Services.IEventManager"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.ISimulator.GetResolver"
Name="GetResolver" Visibility="public">
            <Description>Return interface to resolver
service.</Description>
            <Parameter Id="Smp.ISimulator.GetResolver.return"
Name="return" Direction="return">
               <Description>Interface to mandatory resolver
service.</Description>
               <Type xlink:title="IResolver"
xlink:href="#Smp.Services.IResolver"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.ISimulator.GetLinkRegistry"
Name="GetLinkRegistry" Visibility="public">
            <Description>Return interface to link registry
service.</Description>
            <Parameter
Id="Smp.ISimulator.GetLinkRegistry.return" Name="return"
Direction="return">
               <Description>Interface to mandatory link
registry service.</Description>
               <Type xlink:title="ILinkRegistry"
xlink:href="#Smp.Services.ILinkRegistry"/>
```

```
            </Parameter>
         </Operation>
         <Base xlink:title="IComposite"
xlink:href="#Smp.IComposite"/>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.IFactory"
Name="IFactory"
            Visibility="public"
            Uuid="d56baf53-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface for a component
factory.</Description>
         <Operation Id="Smp.IFactory.GetSpecification"
Name="GetSpecification" Visibility="public">
            <Description>Get specification identifier of
factory.</Description>
            <Parameter
Id="Smp.IFactory.GetSpecification.return" Name="return"
Direction="return">
               <Description>Universally unique identifier of
component specification.</Description>
               <Type xlink:title="Uuid"
xlink:href="#Smp.Uuid"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IFactory.GetImplementation"
Name="GetImplementation"
                    Visibility="public">
            <Description>Get implementation identifier of
factory.</Description>
            <Parameter
Id="Smp.IFactory.GetImplementation.return" Name="return"
Direction="return">
               <Description>Universally unique identifier of
component implementation.</Description>
               <Type xlink:title="Uuid"
xlink:href="#Smp.Uuid"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IFactory.CreateInstance"
Name="CreateInstance" Visibility="public">
            <Description>Create a new instance.</Description>
            <Parameter Id="Smp.IFactory.CreateInstance.return"
Name="return" Direction="return">
               <Description>New component
instance.</Description>
               <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IFactory.DeleteInstance"
Name="DeleteInstance" Visibility="public">
```

```
            <Description>Delete an existing
instance.</Description>
            <Parameter
Id="Smp.IFactory.DeleteInstance.instance" Name="instance">
               <Description>Instance to delete.</Description>
               <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
            </Parameter>
         </Operation>
         <Base xlink:title="IObject"
xlink:href="#Smp.IObject"/>
      </Type>
      <Type xsi:type="Types:NativeType"
Id="Smp.FactoryCollection" Name="FactoryCollection"
            Visibility="public"
            Uuid="d56baf66-e618-11dc-ab64-bf8df6d7b83a">
         <Description>A factory collection is an ordered
collection of factories, which allows iterating all members.
</Description>
         <Platform Name="cpp" Type="vector&lt;IFactory*&gt;"
Namespace="std" Location="vector"/>
      </Type>
      <Type xsi:type="Catalogue:Interface"
Id="Smp.IPublication" Name="IPublication"
            Visibility="public"
            Uuid="d56baf6b-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface that provides functionality to
allow publishing members, including fields, properties and
operations.</Description>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.IComponent"
Name="IComponent"
            Visibility="public"
            Uuid="d55b0e77-e618-11dc-ab64-bf8df6d7b83a">
         <Description>This is the base interface for all SMP
components.</Description>
         <Operation Id="Smp.IComponent.GetParent"
Name="GetParent" Visibility="public">
            <Description>Return the parent component of the
component ("property getter").</Description>
            <Parameter Id="Smp.IComponent.GetParent.return"
Name="return" Direction="return">
               <Description>Parent component of component or
null if component has no parent.</Description>
               <Type xlink:title="IComposite"
xlink:href="#Smp.IComposite"/>
            </Parameter>
         </Operation>
         <Base xlink:title="IObject"
xlink:href="#Smp.IObject"/>
      </Type>
      <Type xsi:type="Types:Enumeration"
Id="Smp.PrimitiveTypeKind"
            Name="PrimitiveTypeKind"
            Visibility="public"
            Uuid="d55b0e31-e618-11dc-ab64-bf8df6d7b83a">
         <Description>This is an enumeration of the available
primitive types.</Description>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_None"
Name="PTK_None" Value="0">
            <Description>No type, e.g. for void.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_Char8"
Name="PTK_Char8" Value="1">
            <Description>8 bit character type.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_Bool"
Name="PTK_Bool" Value="2">
            <Description>Boolean with true and
false.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_Int8"
Name="PTK_Int8" Value="3">
            <Description>8 bit signed integer
type.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_UInt8"
Name="PTK_UInt8" Value="4">
            <Description>8 bit unsigned integer
type.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_Int16"
Name="PTK_Int16" Value="5">
            <Description>16 bit signed integer
type.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_UInt16"
Name="PTK_UInt16" Value="6">
            <Description>16 bit unsigned integer
type.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_Int32"
Name="PTK_Int32" Value="7">
            <Description>32 bit signed integer
type.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_UInt32"
Name="PTK_UInt32" Value="8">
            <Description>32 bit unsigned integer
type.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_Int64"
Name="PTK_Int64" Value="9">
            <Description>64 bit signed integer
type.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_UInt64"
Name="PTK_UInt64" Value="10">
            <Description>64 bit unsigned integer
type.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_Float32"
Name="PTK_Float32" Value="11">
            <Description>32 bit single-precision floating-
point type.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_Float64"
Name="PTK_Float64" Value="12">
            <Description>64 bit double-precision floating-
point type.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_Duration"
Name="PTK_Duration" Value="13">
            <Description>Duration in
nanoseconds.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_DateTime"
Name="PTK_DateTime" Value="14">
            <Description>Absolute time in
nanoseconds.</Description>
         </Literal>
         <Literal Id="Smp.PrimitiveTypeKind.PTK_String8"
Name="PTK_String8" Value="15">
            <Description>8 bit character string.</Description>
         </Literal>
      </Type>
      <Type xsi:type="Types:NativeType"
Id="Smp.ComponentCollection"
            Name="ComponentCollection"
            Visibility="public"
            Uuid="d55b0e81-e618-11dc-ab64-bf8df6d7b83a">
         <Description>A component collection is an ordered
collection of components, which allows iterating all
members.</Description>
         <Platform Name="cpp" Type="vector&lt;IComponent*&gt;"
Namespace="std" Location="vector"/>
      </Type>
      <Type xsi:type="Types:NativeType"
Id="Smp.FailureCollection" Name="FailureCollection"
            Visibility="public"
            Uuid="d572db3a-e618-11dc-ab64-bf8df6d7b83a">
         <Description>A failure collection is an ordered
collection of failures, which allows iterating all
members.</Description>
         <Platform Name="cpp" Type="vector&lt;IFailure*&gt;"
Namespace="std" Location="vector"/>
      </Type>
      <Type xsi:type="Types:Enumeration"
Id="Smp.ModelStateKind" Name="ModelStateKind"
            Visibility="public"
            Uuid="d55d57c7-e618-11dc-ab64-bf8df6d7b83a">
         <Description>This is an enumeration of the available
states of a model. Each model is always in one of these four
model states.</Description>
         <Literal Id="Smp.ModelStateKind.MSK_Created"
Name="MSK_Created" Value="0">
            <Description>The Created state is the initial
state of a model. Model creation is done by an external
mechanism, e.g. by factories.
This state is entered automatically after the model has been
created.
This state is left via the Publish() state transition.
</Description>
         </Literal>
         <Literal Id="Smp.ModelStateKind.MSK_Publishing"
Name="MSK_Publishing" Value="1">
            <Description>In Publishing state, the model is
allowed to publish features. This includes publication of
fields, operations and properties. In addition, the model is
allowed to create other models.
This state is entered via the Publish() state transition.
This state is left via the Configure() state transition.
</Description>
         </Literal>
         <Literal Id="Smp.ModelStateKind.MSK_Configured"
Name="MSK_Configured" Value="2">
            <Description>In Configured state, the model has
been fully configured. This configuration may be done by
external components, or internally by the model itself, e.g.
by reading data from an external source.
This state is entered via the Configure() state transition.
This state is left via the Connect() state transition.
</Description>
         </Literal>
         <Literal Id="Smp.ModelStateKind.MSK_Connected"
Name="MSK_Connected" Value="3">
            <Description>In Connected state, the model is
connected to the simulator. In this state, neither publication
nor creation of other models is allowed anymore.
This state is entered via the Connect() state transition.
This is the final state of a model, and only left on
termination.</Description>
```

```
         </Literal>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.IModel"
Name="IModel"
            Visibility="public"
            Uuid="d55d57da-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface for a model.</Description>
         <NestedType xsi:type="Types:Exception"
Id="Smp.IModel.InvalidModelState"
                     Name="InvalidModelState"
                     Visibility="public"
                     Uuid="d55d57f6-e618-11dc-ab64-
bf8df6d7b83a">
            <Description>This exception is raised by a model
when one of the state transition commands is called in an
invalid state.</Description>
            <Field
Id="Smp.IModel.InvalidModelState.invalidState"
Name="invalidState"
                   Visibility="public">
               <Description>State that is not
valid.</Description>
               <Type xlink:title="ModelStateKind"
xlink:href="#Smp.ModelStateKind"/>
            </Field>
            <Field
Id="Smp.IModel.InvalidModelState.expectedState"
Name="expectedState"
                   Visibility="public">
               <Description>State that was
expected.</Description>
               <Type xlink:title="ModelStateKind"
xlink:href="#Smp.ModelStateKind"/>
            </Field>
            <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         </NestedType>
         <Operation Id="Smp.IModel.GetState" Name="GetState"
Visibility="public">
            <Description>Returns the state the model is
currently in.</Description>
            <Parameter Id="Smp.IModel.GetState.return"
Name="return" Direction="return">
               <Description>Current model state.</Description>
               <Type xlink:title="ModelStateKind"
xlink:href="#Smp.ModelStateKind"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IModel.Publish" Name="Publish"
Visibility="public">
```

```
            <Description>Request the model to publish its
fields, properties and operations against the provided
publication receiver.</Description>
            <Parameter Id="Smp.IModel.Publish.receiver"
Name="receiver">
               <Description>Publication
receiver.</Description>
               <Type xlink:title="IPublication"
xlink:href="#Smp.IPublication"/>
            </Parameter>
            <RaisedException xlink:title="InvalidModelState"
xlink:href="#Smp.IModel.InvalidModelState"/>
         </Operation>
         <Operation Id="Smp.IModel.Configure" Name="Configure"
Visibility="public">
            <Description>Request the model to perform any
custom configuration. The model can create and configure other
models using the field values of its published
fields.</Description>
            <Parameter Id="Smp.IModel.Configure.logger"
Name="logger">
               <Description>Logger service for logging of
error messages during configuration.</Description>
               <Type xlink:title="ILogger"
xlink:href="#Smp.Services.ILogger"/>
            </Parameter>
            <RaisedException xlink:title="InvalidModelState"
xlink:href="#Smp.IModel.InvalidModelState"/>
         </Operation>
         <Operation Id="Smp.IModel.Connect" Name="Connect"
Visibility="public">
            <Description>Allow the model to connect to the
simulator.</Description>
            <Parameter Id="Smp.IModel.Connect.simulator"
Name="simulator">
               <Description>Simulation Environment that hosts
the model.</Description>
               <Type xlink:title="ISimulator"
xlink:href="#Smp.ISimulator"/>
            </Parameter>
            <RaisedException xlink:title="InvalidModelState"
xlink:href="#Smp.IModel.InvalidModelState"/>
         </Operation>
         <Base xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
      </Type>
      <Type xsi:type="Types:Exception" Id="Smp.DuplicateName"
Name="DuplicateName"
            Visibility="public"
            Uuid="d56baf6e-e618-11dc-ab64-bf8df6d7b83a">
         <Description>This exception is raised when trying to
add an object to a collection of objects, which have to have
unique names, but another object with the same name does exist
already in this collection. This would lead to duplicate
names.</Description>
         <Field Id="Smp.DuplicateName.duplicateName"
Name="duplicateName" Visibility="public">
            <Description>Name that already exists in the
collection.</Description>
            <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
         </Field>
         <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
      </Type>
      <Type xsi:type="Types:NativeType"
Id="Smp.AnySimpleArray" Name="AnySimpleArray"
            Visibility="public"
            Uuid="d55b0e53-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Array of AnySimple values.</Description>
         <Platform Name="cpp" Type="AnySimple*"/>
      </Type>
      <Type xsi:type="Types:NativeType"
Id="Smp.ModelCollection" Name="ModelCollection"
            Visibility="public"
            Uuid="d55d57fd-e618-11dc-ab64-bf8df6d7b83a">
         <Description>A model collection is an ordered
collection of models, which allows iterating all
members.</Description>
         <Platform Name="cpp" Type="vector&lt;IModel*&gt;"
Namespace="std" Location="vector"/>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.IService"
Name="IService"
            Visibility="public"
            Uuid="d55d57d1-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Base interface for all SMP
services.</Description>
         <Base xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
      </Type>
      <Type xsi:type="Types:Exception" Id="Smp.InvalidAnyType"
Name="InvalidAnyType"
            Visibility="public"
            Uuid="d56baf73-e618-11dc-ab64-bf8df6d7b83a">
         <Description>This exception is raised when trying to
use an AnySimple argument of wrong type.</Description>
         <Field Id="Smp.InvalidAnyType.invalidType"
Name="invalidType" Visibility="public">
            <Description>Type that is not valid.</Description>
            <Type xlink:title="PrimitiveTypeKind"
xlink:href="#Smp.PrimitiveTypeKind"/>
         </Field>
         <Field Id="Smp.InvalidAnyType.expectedType"
Name="expectedType" Visibility="public">
            <Description>Type that was expected.</Description>
            <Type xlink:title="PrimitiveTypeKind"
xlink:href="#Smp.PrimitiveTypeKind"/>
         </Field>
         <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.ITask"
Name="ITask" Visibility="public"
            Uuid="d5752566-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface for a task, which is an
ordered collection of entry points.</Description>
         <Operation Id="Smp.ITask.AddEntryPoint"
Name="AddEntryPoint" Visibility="public">
            <Description>Add an entry point to the
task.</Description>
            <Parameter Id="Smp.ITask.AddEntryPoint.entryPoint"
Name="entryPoint">
               <Description>Entry point to add to
task.</Description>
               <Type xlink:title="IEntryPoint"
xlink:href="#Smp.IEntryPoint"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.ITask.GetEntryPoints"
Name="GetEntryPoints" Visibility="public">
            <Description>Query for the collection of all entry
points. The order of entry points in the collection is the
order in which they have been added to the task.</Description>
            <Parameter Id="Smp.ITask.GetEntryPoints.return"
Name="return" Direction="return">
               <Description>Collection of entry
points.</Description>
               <Type xlink:title="EntryPointCollection"
xlink:href="#Smp.EntryPointCollection"/>
            </Parameter>
         </Operation>
         <Base xlink:title="IEntryPoint"
xlink:href="#Smp.IEntryPoint"/>
      </Type>
      <Type xsi:type="Types:NativeType"
Id="Smp.ServiceCollection" Name="ServiceCollection"
            Visibility="public"
            Uuid="d55d57d5-e618-11dc-ab64-bf8df6d7b83a">
         <Description>A service collection is an ordered
collection of services, which allows iterating all
members.</Description>
         <Platform Name="cpp" Type="vector&lt;IService*&gt;"
Namespace="std" Location="vector"/>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.IAggregate"
Name="IAggregate"
            Visibility="public"
            Uuid="d55fc83f-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface for an aggregate
component.</Description>
         <Operation Id="Smp.IAggregate.GetReferences"
Name="GetReferences" Visibility="public">
            <Description>Query for the collection of all
references of the aggregate component. </Description>
            <Parameter
Id="Smp.IAggregate.GetReferences.return" Name="return"
Direction="return">
               <Description>Collection of
references.</Description>
               <Type xlink:title="ReferenceCollection"
xlink:href="#Smp.ReferenceCollection"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IAggregate.GetReference"
Name="GetReference" Visibility="public">
            <Description>Query for a reference of this
aggregate component by its name.</Description>
            <Parameter Id="Smp.IAggregate.GetReference.return"
Name="return" Direction="return">
               <Description>Reference queried for by name, or
null if no reference with this name exists.</Description>
               <Type xlink:title="IReference"
xlink:href="#Smp.IReference"/>
            </Parameter>
            <Parameter Id="Smp.IAggregate.GetReference.name"
Name="name">
               <Description>Reference name.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
         <Base xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
      </Type>
      <Type xsi:type="Types:Exception"
Id="Smp.InvalidObjectType" Name="InvalidObjectType"
            Visibility="public"
            Uuid="d56baf7b-e618-11dc-ab64-bf8df6d7b83a">
         <Description>This exception is raised when trying to
pass an object of wrong type.</Description>
         <Base xlink:title="Exception"
xlink:href="#Smp.Exception"/>
         <Association Id="Smp.InvalidObjectType.invalidObject"
Name="invalidObject"
                      Visibility="public">
            <Description>Object that is not of valid
type.</Description>
            <Type xlink:title="IObject"
xlink:href="#Smp.IObject"/>
         </Association>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.IReference"
Name="IReference"
            Visibility="public"
            Uuid="d55fc829-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface for a reference.</Description>
         <Operation Id="Smp.IReference.GetComponents"
Name="GetComponents" Visibility="public">
            <Description>Query for the collection of all
referenced components.</Description>
            <Parameter
Id="Smp.IReference.GetComponents.return" Name="return"
Direction="return">
               <Description>Collection of referenced
components.</Description>
               <Type xlink:title="ComponentCollection"
xlink:href="#Smp.ComponentCollection"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IReference.GetComponent"
Name="GetComponent" Visibility="public">
            <Description>Query for a referenced component by
its name.</Description>
            <Parameter Id="Smp.IReference.GetComponent.return"
Name="return" Direction="return">
               <Description>Referenced component with the
given name, or null if no referenced component with the given
name could be found. </Description>
               <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
            </Parameter>
            <Parameter Id="Smp.IReference.GetComponent.name"
Name="name">
               <Description>Component name.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
         <Base xlink:title="IObject"
xlink:href="#Smp.IObject"/>
      </Type>
      <Type xsi:type="Types:NativeType"
Id="Smp.ReferenceCollection"
            Name="ReferenceCollection"
            Visibility="public"
            Uuid="d55fc83a-e618-11dc-ab64-bf8df6d7b83a">
         <Description>A reference collection is an ordered
collection of references, which allows iterating all
members.</Description>
         <Platform Name="cpp" Type="vector&lt;IReference*&gt;"
Namespace="std" Location="vector"/>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.IComposite"
Name="IComposite"
            Visibility="public"
            Uuid="d55d5818-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface for a composite
component.</Description>
         <Operation Id="Smp.IComposite.GetContainers"
Name="GetContainers" Visibility="public">
            <Description>Query for the collection of all
containers of the composite component.</Description>
            <Parameter
Id="Smp.IComposite.GetContainers.return" Name="return"
Direction="return">
               <Description>Collection of
containers.</Description>
               <Type xlink:title="ContainerCollection"
xlink:href="#Smp.ContainerCollection"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IComposite.GetContainer"
Name="GetContainer" Visibility="public">
            <Description>Query for a container of this
composite component by its name.</Description>
            <Parameter Id="Smp.IComposite.GetContainer.return"
Name="return" Direction="return">
               <Description>Container queried for by name, or
null if no container with this name exists.</Description>
               <Type xlink:title="IContainer"
xlink:href="#Smp.IContainer"/>
            </Parameter>
            <Parameter Id="Smp.IComposite.GetContainer.name"
Name="name">
               <Description>Container name.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
         <Base xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.IContainer"
Name="IContainer"
            Visibility="public"
            Uuid="d55d5802-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface for a container.</Description>
         <Operation Id="Smp.IContainer.GetComponents"
Name="GetComponents" Visibility="public">
            <Description>Query for the collection of all
components in the container. </Description>
            <Parameter
Id="Smp.IContainer.GetComponents.return" Name="return"
Direction="return">
               <Description>Collection of contained
components.</Description>
               <Type xlink:title="ComponentCollection"
xlink:href="#Smp.ComponentCollection"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IContainer.GetComponent"
Name="GetComponent" Visibility="public">
            <Description>Query for a component contained in
the container by name.</Description>
            <Parameter Id="Smp.IContainer.GetComponent.return"
Name="return" Direction="return">
               <Description>Child component, or null if no
child component with the given name exists.</Description>
               <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
            </Parameter>
            <Parameter Id="Smp.IContainer.GetComponent.name"
Name="name">
               <Description>Child name.</Description>
               <Type xlink:title="String8"
xlink:href="#Smp.String8"/>
            </Parameter>
         </Operation>
         <Base xlink:title="IObject"
xlink:href="#Smp.IObject"/>
      </Type>
      <Type xsi:type="Types:NativeType"
Id="Smp.ContainerCollection"
            Name="ContainerCollection"
            Visibility="public"
            Uuid="d55d5813-e618-11dc-ab64-bf8df6d7b83a">
         <Description>A container collection is an ordered
collection of containers, which allows iterating all
members.</Description>
```

```
         <Platform Name="cpp" Type="vector&lt;IContainer*&gt;"
Namespace="std" Location="vector"/>
      </Type>
      <Type xsi:type="Catalogue:Interface" Id="Smp.IEventSink"
Name="IEventSink"
            Visibility="public"
            Uuid="d55d5787-e618-11dc-ab64-bf8df6d7b83a">
         <Description>Interface of an event sink that can be
subscribed to an event source (IEventSource).</Description>
         <Operation Id="Smp.IEventSink.GetOwner"
Name="GetOwner" Visibility="public">
            <Description>This method returns the Component
that owns the event sink.</Description>
            <Parameter Id="Smp.IEventSink.GetOwner.return"
Name="return" Direction="return">
               <Description>Owner of event sink.</Description>
               <Type xlink:title="IComponent"
xlink:href="#Smp.IComponent"/>
            </Parameter>
         </Operation>
         <Operation Id="Smp.IEventSink.Notify" Name="Notify"
Visibility="public">
            <Description>This event handler method is called
when an event is emitted.</Description>
            <Parameter Id="Smp.IEventSink.Notify.sender"
Name="sender">
               <Description>Object emitting the
event.</Description>
               <Type xlink:title="IObject"
xlink:href="#Smp.IObject"/>
            </Parameter>
            <Parameter Id="Smp.IEventSink.Notify.arg"
Name="arg">
               <Description>Event argument.</Description>
               <Type xlink:title="AnySimple"
xlink:href="#Smp.AnySimple"/>
            </Parameter>
         </Operation>
         <Base xlink:title="IObject"
xlink:href="#Smp.IObject"/>
      </Type>
      <Type xsi:type="Types:NativeType"
Id="Smp.EventSinkCollection"
            Name="EventSinkCollection"
            Visibility="public"
            Uuid="d55d5797-e618-11dc-ab64-bf8df6d7b83a">
         <Description>An event sink collection is an ordered
collection of event sinks, which allows iterating all
members.</Description>
         <Platform Name="cpp" Type="vector&lt;IEventSink*&gt;"
Namespace="std" Location="vector"/>
```

```
      </Type>
   </Namespace>
</Catalogue:Catalogue>
```