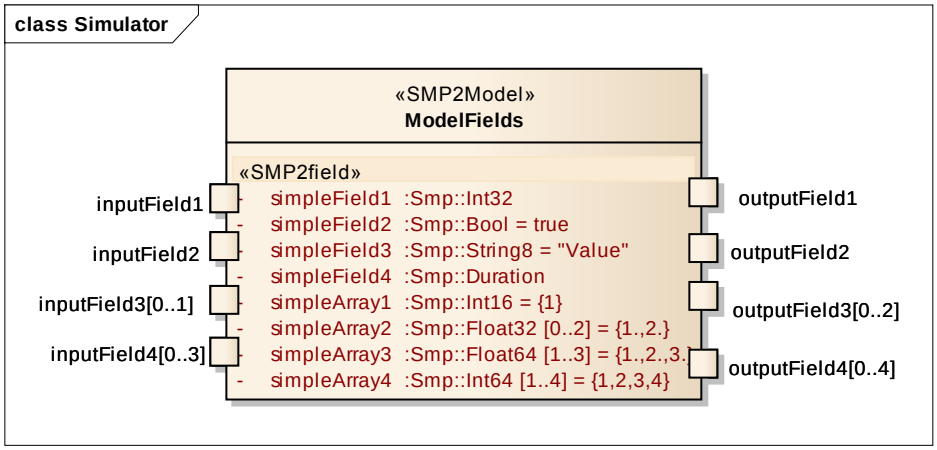
Simubox's SMP2 Code Generator: Test Cases

Sumário

1. Fields

The test cases for fields are specified for a model named *ModelFieds*. When the type of field is output or input, the *type* and *value* properties are described as tagged values.



		SMP Type (Smp::)	Init. Value	Tagged value					
#	Field	Type		State	VK	Output	Input	Lower	Upper
1	simpleField1	Int32		—	—	—	—	—	—
2	simpleField2	Bool	true	True	—	—	—	—	—
3	simpleField3	String8	“Value”	False	Debug	—	—	—	—
4	simpleField4	Duration		True	None	—	—	—	—
5	arrayField1	Int16	{1}	—	—	—	—	—	1
6	arrayField2	Float32	{1.,2.}	True	—	—	—	0	2
7	arrayField3	Float64	{1.,2.,3.}	False	Expert	—	—	—	3
8	arrayField4	Int64	{1,2,3,4}	False	All	—	—	—	4
9	inputField1	Int32		—	—	—	True	—	—
10	outputField1	Bool		True	—	True	False	—	—
11	inputField2	String8		False	Debug	False	True	—	—
12	onputField2	Duration	100	True	None	True	—	—	—
13	inputField3	Int16		—	—	—	True	—	1
14	outputField3	Float32		True	—	True	False	0	2
15	inputField4	Float64	{3.1,3.2, 3.3}	False	Expert	False	True	—	3
16	ouputField4	Int64		False	All	True	False	—	4

Expected results:

#	Definition	Implementation
1	private: Smp::Int32 simpleField1;	this->simpleField1 = 0 receiver->PublishField("simpleField1", "Simple Field1", &simpleField1, Smp::VK_None, true, false, false);
2	private: Smp::Bool simpleField2;	this->simpleField2 = false this->simpleField2 = true;

		receiver->PublishField("simpleField2", "", &simpleField2, Smp::VK_None, true, false, false);
3	private: Smp::String8 simpleField3;	this->simpleField3 = NULL this->simpleField3 = "Value"; receiver->PublishField("simpleField3", "", static_const<void*>(&simpleField3), Smp::PTK_String8, Smp::VK_Debug, false, false, false);
4	private: Smp::Duration simpleField4;	this->simpleField4 = 0 receiver->PublishField("simpleField4", "", static_const<void*>(&simpleField4), Smp::PTK_Duration, Smp::VK_None, true, false, false);
5	private: Smp::Int16 simpleArray1[1];	this->simpleArray1[0] = 0 this->simpleArray1[0] = 1; receiver->PublishArray("simpleArray1", "", 1, static_cast<void*>(simpleArray1), Smp::PTK_Int16, Smp::VK_None, true, false, false);
6	private: Smp::Float32 simpleArray2[2];	this->simpleArray2[0] = 0.0 this->simpleArray2[1] = 0.0 this->simpleArray2[0] = 1.; this->simpleArray2[1] = 2.; receiver->PublishArray("simpleArray2", "", 2, static_cast<void*>(simpleArray2), Smp::PTK_Float32,

		Smp::VK_None, true, false, false);
7	private: Smp::Float64 simpleArray3[3];	this->simpleArray3[0] = 0.0 this->simpleArray3[1] = 0.0 this->simpleArray3[2] = 0.0 this->simpleArray3[0] = 1.; this->simpleArray3[1] = 2.; this->simpleArray3[2] = 3.; receiver->PublishArray("simpleArray3", "", 3, static_cast<void*>(simpleArray3), Smp::PTK_Float64, Smp::VK_Expert, true, false, false);

8	private: Smp::Int64
	simpleArray4[4];
		this->simpleArray4[0] = 0
		this->simpleArray4[1] = 0
		this->simpleArray4[2] = 0
		this->simpleArray4[3] = 0
		this->simpleArray4[0] = 1;
		this->simpleArray4[1] = 2;
		this->simpleArray4[2] = 3;
		this->simpleArray4[3] = 4;
		
		receiver->PublishArray(
		"simpleArray4",
		"",
		4,
		static_cast<void*>(simpleArray4),
		Smp::PTK_Int64,
		Smp::VK_All,
		false,
		false,
		false);

9	public: Smp::Int32 inputField1;	this->inputField1 = 0 receiver->PublishField("inputField1", "", &inputField1, Smp::VK_None, true, true, false);
10	public: Smp::Bool outputField1;	this->outputField1 = false receiver->PublishField("outputField1", "", &outputField1, Smp::VK_None,

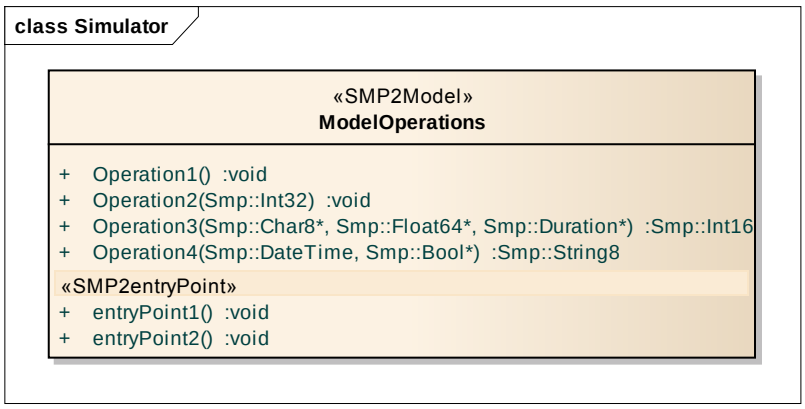
		true, false, true);
11	public: Smp::String8 inputField2;	this->inputField2 = NULL receiver->PublishField("inputField2", "", &inputField2, Smp::VK_Debug, false, true, false);
12	public: Smp::Duration outputField2;	this->outputField2 = 0 this->outputField2 = 100; receiver->PublishField("outputField2", "", &outputField2, Smp::VK_None, true, false, true);
13	public: Smp::Int16 inputField3[1];	this->inputField3[0] = 0 receiver->PublishArray("inputField3", "", 1, static_cast<void*>(inputField3), Smp::PTK_Int16, Smp::VK_None, true, true, false);
14	public: Smp::Float32 outputField3[2];	this->outputField3[0] = 0.0 this->outputField3[1] = 0.0 receiver->PublishArray("outputField3", "", 2, static_cast<void*>(outputField3), Smp::PTK_Float32, Smp::VK_None, true, false, true);

15	public:
	Smp::Float64
	inputField4[3];
		this->inputField4[0] = 0.0
		this->inputField4[1] = 0.0
		this->inputField4[2] = 0.0
		this->inputField4[0] = 3.1;
		this->inputField4[1] = 3.2;
		this->inputField4[2] = 3.3;

		receiver->PublishArray("inputField4", "", 3, static_cast<void*>(inputField4), Smp::PTK_Float64, Smp::VK_Expert, false, true, false);
16	public: Smp::Int64 outputField4[4];	this->outputField4[0] = 0 this->outputField4[1] = 0 this->outputField4[2] = 0 this->outputField4[3] = 0 receiver->PublishArray("outputField4", "", 4, static_cast<void*>(outputField4), Smp::PTK_Int64, Smp::VK_All, false, false, true);

2. Entry Points and Operations

The test cases for fields are specified for a model named *ModelOperations*.



		Tags	Return type			
#	**Operation**	**Type**		**1**	**2**	**3**
1	entryPoint1		void			
2	entryPoint2	inputField= ModelFields:: inputField1 outputField= ModelFields:: outputField1	void			

3	Operation1		void			
4	Operation2	viewKind= none	void	Smp::Int32 [in]		
5	Operation3	viewKind= expert	Smp::Int16	Smp::Duration [in/out]	Smp::Float64 [out]	Smp::Char8 [out]
6	Operation4		Smp::String8	Smp::Bool[2] [out]	Smp::DateTime [in]	

Expected results:

#	Definition	Implementation
1	public: ::Smp::IEntryPoint * entryPoint1; public: void _entryPoint1();	`this->entryPoint1 = new SB::EntryPoint< Test::ModelOperations >( "entryPoint1", "Entry point 1", this, &Test::ModelOperations::_entryPoint1); this->AddEntryPoint(entryPoint1);`
2	public: ::Smp::IEntryPoint * entryPoint2; public: void _entryPoint2();	`this->entryPoint2 = new SB::EntryPoint< Test::ModelOperations >( "entryPoint2", "Entry point 2", this, &Test::ModelOperations::_entryPoint2); this->AddEntryPoint(entryPoint2);`
3	void Operation1();	`// Operation1 operation = receiver->PublishOperation( "Operation1", "Operation 1", Smp::VK_None);`

4	void Operation2(Smp::Int32 param1);	`// Operation2 operation = receiver->PublishOperation( "Operation2", "", Smp::VK_None);`
		
		`if(operation) { operation->PublishParameter( "param1", "parameter 1", Smp::Uuid_Int32, Smp::Publication::PDK_In);`
		
		`}`

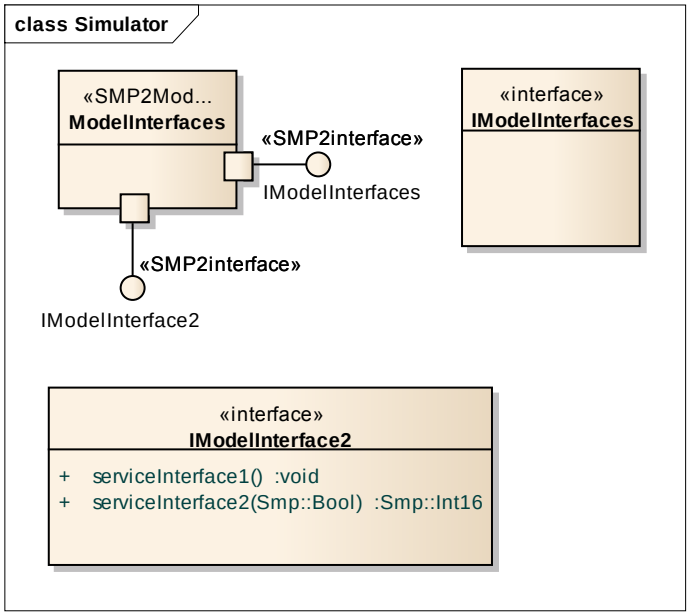
	Smp::Int16 Operation3(Smp::Char8 param3, Smp::Float64 param2, Smp::Duration param1);	`// Operation3 operation = receiver->PublishOperation( "Operation3", "", Smp::VK_None);`
		
		`if(operation) { operation->PublishParameter( "Return", "Return Parameter", Smp::Uuid_Int16,`

		Smp::Publication::PDK_Return); operation->PublishParameter("param3", "", Smp::Uuid_Char8, Smp::Publication::PDK_Out); operation->PublishParameter("param2", "", Smp::Uuid_Float64, Smp::Publication::PDK_Out); operation->PublishParameter("param1", "", Smp::Uuid_Duration, Smp::Publication::PDK_InOut); }

	Smp::String8 Operation4(Smp:: DateTime param2, Smp::Bool param1);	// Operation4
		operation = receiver->PublishOperation(
		"Operation4",
		"",
		Smp::VK_None);
		if(operation) {
		operation->PublishParameter(
		"Return",
		"Return Parameter",
		Smp::Uuid_String8,
		Smp::Publication::PDK_Return);
		
		operation->PublishParameter(
		"param2",
		"",
		Smp::Uuid_DateTime,
		Smp::Publication::PDK_In);
		operation->PublishParameter(
		"param1",
		"",
		Smp::Uuid_Bool,
		Smp::Publication::PDK_Out);
		}

3. Exposed interfaces

The test cases for fields are specified for a model named *ModelInterfaces*.



#	Exposed interterface	Inherited services
1	IModelInterfaces	*Void serviceInterface1*(); Smp::Int16 *serviceInterface2*(Smp::Bool param1);
2	IModelInterface2	

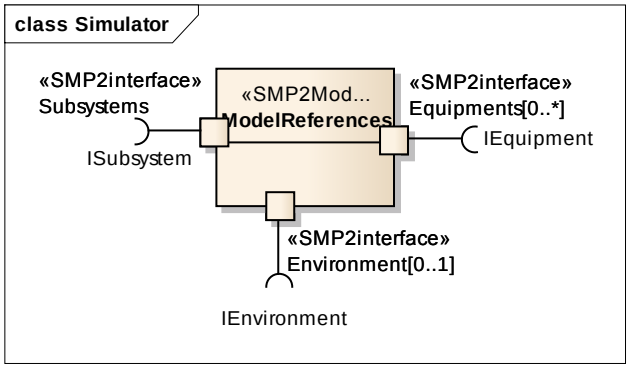
Expected results:

#	Class declaration	base interfaces declaration	Base interfaces implementation
1	class		

2	ModelInterfaces : public virtual SB::Management::ManagedModel, public virtual SB::Management::EntryPointPublisher, public virtual IModelInterfaces, public virtual IModelInterface2	virtual void *serviceInterface1*();
		virtual Smp::Int16 *serviceInterface2*(Smp::Bool param1);
			`void Catalogue::ModelInterfaces::serviceInterface1() { // User code here }`
			
			`Smp::Int16 Catalogue::ModelInterfaces::serviceInterface2(Smp::Bool param1) { // User code here return Smp::Int16() }`

4. Required References

The test cases for fields are specified for a model named *ModelReferences*.



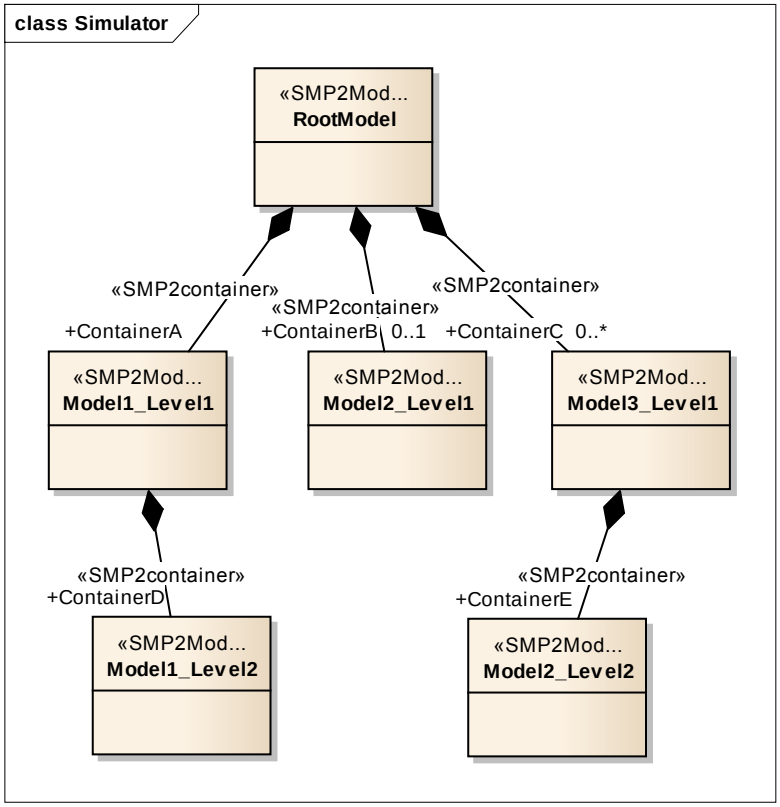
#	Req. Interface	Lower bound	Upper bound
1	ISubsystem		
2	IEquipment	0	1
3	IEnvironment	0	*

Expected results:

#	Definition	Implementation
1	protected: SB::Management::ManagedReference< ISubsystem >* _Subsystems; protected: Smp::Management::IManagedReference* Subsystems;	_Subsystems = SB::Management::ManagedReference< ISubsystem >::CreateInstance("Subsystems" , "" , this); Subsystems = _Subsystems;
2	protected: SB::Management::ManagedReference< IEquipment >* _Equipments; protected: Smp::Management::IManagedReference* Equipments;	Equipments = SB::Management::ManagedReference< IEquipment >::CreateInstance("Equipments" , "" , this , 0); Equipments = _Equipments;
3	protected: SB::Management::ManagedReference< IEnvironment >* _Environment; protected: Smp::Management::IManagedReference* Environment;	_Environment = SB::Management::ManagedReference< IEnvironment >::CreateInstance("Environment" , "" , this , 0 , 1); Environment = _Environment;

5. Containers

The test cases for fields are specified for a set of models, composed in an hierarchy: RootModel, Model1_Level1, Model2_Level1, Model3_Level1, Model1_Level2, Model2_Level2.



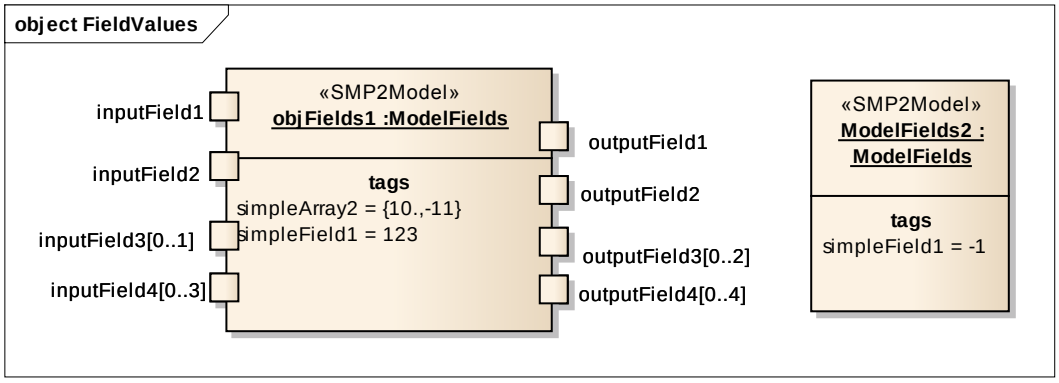
#	Req. Interface	Child	Container		
1	RootModel	Model1_Level1	ContainerA		
2	RootModel	Model2_Level1	ContainerB	0	1
3	RootModel	Model3_Level1	ContainerC	0	*
4	Model1_Level1	Model1_Level2	ContainerD		
5	Model3_Level1	Model2_Level2	ContainerE		
6	Model2_Level1	none			

Expected results:

#	Definition	Implementation
1	protected: SB::Management::ManagedContainer<Model1_Level1 >* _ContainerA; public: Smp::Management::IManagedContainer* ContainerA;	_ContainerA = SB::Management::ManagedContainer<Model1_Level1 >::CreateInstance("ContainerA" , "" , this); ContainerA = _ContainerA;
2	protected: SB::Management::ManagedContainer<	_ContainerB = SB::Management::ManagedContainer<Model2_Level1 >::CreateInstance(

	Model2_Level1 >* _ContainerB; public: Smp::Management::IManagedContainer* ContainerB;	"ContainerB" , "" , this , 0 , 1); ContainerB = _ContainerB;
3	protected: SB::Management::ManagedContainer< Model3_Level1 >* _ContainerC; public: Smp::Management::IManagedContainer* ContainerC;	_ContainerC = SB::Management::ManagedContainer< Model3_Level1 >::CreateInstance("ContainerC" , "" , this , 0); ContainerC = _ContainerC;
4	protected: SB::Management::ManagedContainer< Model1_Level2 >* _ContainerD; public: Smp::Management::IManagedContainer* ContainerD;	_ContainerD = SB::Management::ManagedContainer< Model1_Level2 >::CreateInstance("ContainerD" , "" , this); ContainerD = _ContainerD;
5	protected: SB::Management::ManagedContainer< Model2_Level2 >* _ContainerE; public: Smp::Management::IManagedContainer* ContainerE;	_ContainerE = SB::Management::ManagedContainer< Model2_Level2 >::CreateInstance("ContainerE" , "" , this); ContainerE = _ContainerE;
6	None code generated for containers. The class doesn't derive from composite and not includes any other model.	

6. Default Field Values for an Assembly



#	Model Instance	Parameters	Value	Value source
1	ObjFields1	simpleField1	123	**instance**
2		simpleField2	true	Model
3		simpleField3	“Value”	Model
4		arrayField1	{1}	Model
5		arrayField2	{10., -11}	**instance**
6		arrayField3	{1.,2.,3.}	Model
7		arrayField4	{1,2,3,4}	Model
8	ObjFields2	simpleField1	-1	**instance**
9		simpleField2	true	Model
10		simpleField3	“Value”	Model
11		arrayField1	{1}	Model
12		arrayField2	{1.,2.}	Model
13		arrayField3	{1.,2.,3.}	Model
14		arrayField4	{1,2,3,4}	Model

Expected results:

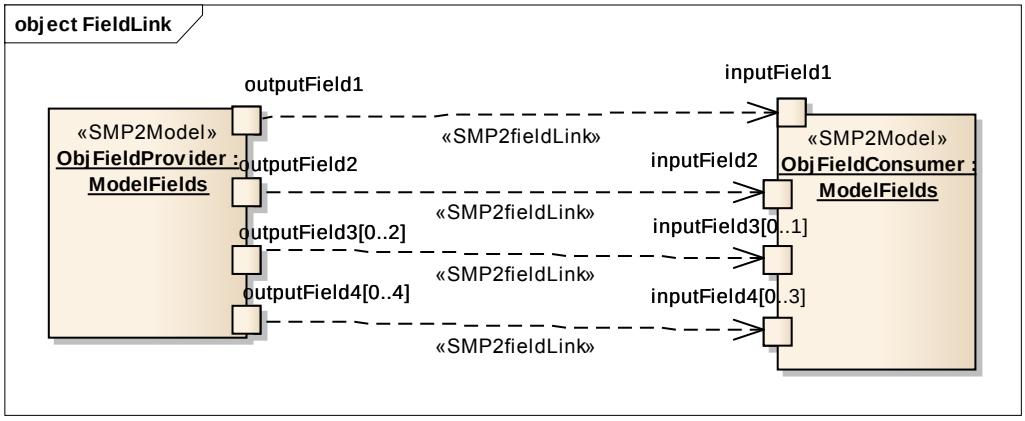
SMP2Assembly

```
<ModelInstance Id="ModelFields.objFields1" Name="objFields1"
Implementation="378B9821-00B9-4301-8A61-1B5260A0E3E1">
	<Model xlink:title="ModelFields" xlink:href="./Catalogue.cat#ModelFields"
/>
	<FieldValue xsi:type="Types:BoolValue"
Field="./Catalogue.cat#ModelFields.simpleField2" Value="true"/>
	<FieldValue xsi:type="Types:String8Value"
Field="./Catalogue.cat#ModelFields.simpleField3" Value="Value"/>
	<FieldValue xsi:type="Types:Int16ArrayValue"
Field="./Catalogue.cat#ModelFields.simpleArray1">
		<ItemValue Field="./simpleArray1[0]" Value="1"/>
	</FieldValue>
	<FieldValue xsi:type="Types:Float64ArrayValue"
Field="./Catalogue.cat#ModelFields.simpleArray3">
		<ItemValue Field="./simpleArray3[0]" Value="1."/>
		<ItemValue Field="./simpleArray3[1]" Value="2."/>
		<ItemValue Field="./simpleArray3[2]" Value="3."/>
	</FieldValue>
	<FieldValue xsi:type="Types:Int64ArrayValue"
Field="./Catalogue.cat#ModelFields.simpleArray4">
		<ItemValue Field="./simpleArray4[0]" Value="1"/>
		<ItemValue Field="./simpleArray4[1]" Value="2"/>
		<ItemValue Field="./simpleArray4[2]" Value="3"/>
```

```
            <ItemValue Field="./simpleArray4[3]" Value="4"/>
      </FieldValue>
      <FieldValue xsi:type="Types:Float32ArrayValue"
Field="./Catalogue.cat#ModelFields.simpleArray2">
            <ItemValue Field="./simpleArray2[0]" Value="10."/>
            <ItemValue Field="./simpleArray2[1]" Value="-11"/>
      </FieldValue>
      <FieldValue xsi:type="Types:Int32Value"
Field="./Catalogue.cat#ModelFields.simpleField1" Value="123"/>
      <Container xlink:title="ModelsContainer" xlink:href="#??#"/>
</ModelInstance>

<ModelInstance Id="ModelFields.ModelFields2" Name="ModelFields2"
Implementation="378B9821-00B9-4301-8A61-1B5260A0E3E1">
      <Model xlink:title="ModelFields" xlink:href="./Catalogue.cat#ModelFields"
/>
      <FieldValue xsi:type="Types:BoolValue"
Field="./Catalogue.cat#ModelFields.simpleField2" Value="true"/>
      <FieldValue xsi:type="Types:String8Value"
Field="./Catalogue.cat#ModelFields.simpleField3" Value="Value"/>
      <FieldValue xsi:type="Types:Int16ArrayValue"
Field="./Catalogue.cat#ModelFields.simpleArray1">
            <ItemValue Field="./simpleArray1[0]" Value="1"/>
      </FieldValue>
      <FieldValue xsi:type="Types:Float32ArrayValue"
Field="./Catalogue.cat#ModelFields.simpleArray2">
            <ItemValue Field="./simpleArray2[0]" Value="1."/>
            <ItemValue Field="./simpleArray2[1]" Value="2."/>
      </FieldValue>
      <FieldValue xsi:type="Types:Float64ArrayValue"
Field="./Catalogue.cat#ModelFields.simpleArray3">
            <ItemValue Field="./simpleArray3[0]" Value="1."/>
            <ItemValue Field="./simpleArray3[1]" Value="2."/>
            <ItemValue Field="./simpleArray3[2]" Value="3."/>
      </FieldValue>
      <FieldValue xsi:type="Types:Int64ArrayValue"
Field="./Catalogue.cat#ModelFields.simpleArray4">
            <ItemValue Field="./simpleArray4[0]" Value="1"/>
            <ItemValue Field="./simpleArray4[1]" Value="2"/>
            <ItemValue Field="./simpleArray4[2]" Value="3"/>
            <ItemValue Field="./simpleArray4[3]" Value="4"/>
      </FieldValue>
      <FieldValue xsi:type="Types:Int32Value"
Field="./Catalogue.cat#ModelFields.simpleField1" Value="-1"/>
      <Container xlink:title="ModelsContainer" xlink:href="#??#"/>
</ModelInstance>
```

7. Assembly with field links

During the production of the assembly, the data types of the fields are not checked. Then, for this test case, the fields are linked ignoring the type constraints (see the next image). For the sake of assembly production validation, these restrictions are not critical, but be aware that this assembly will fail in case of building a simulation.



#	Provider instance	Output field	Consumer instance	Input field
1	ObjFieldProvider	outputField1	ObjFieldConsumer	inputField1
2	ObjFieldProvider	outputField2	ObjFieldConsumer	inputField2
3	ObjFieldProvider	outputField3	ObjFieldConsumer	inputField3
4	ObjFieldProvider	outputField4	ObjFieldConsumer	inputField4

Expected results:

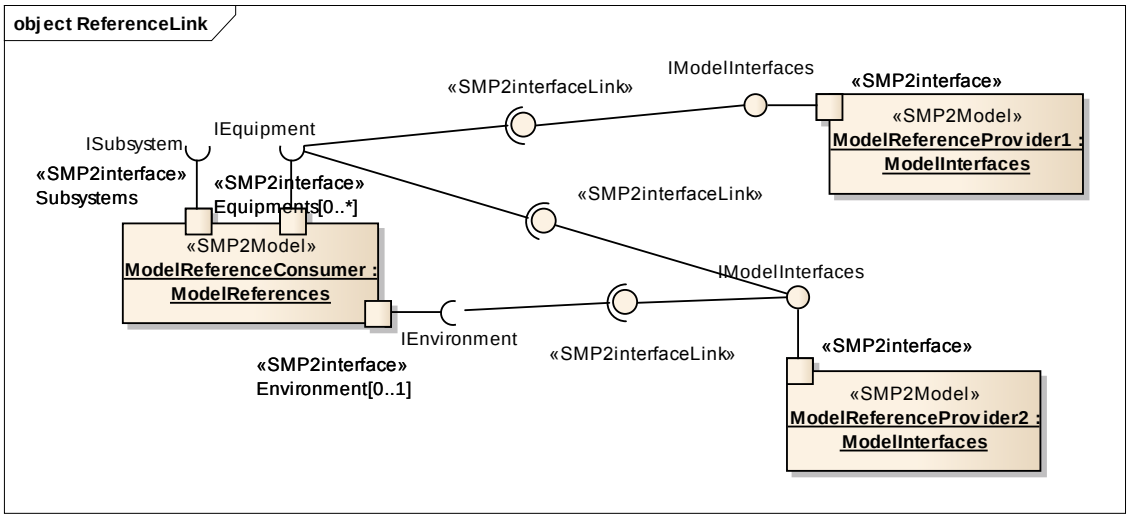
SMP2Assembly

```
...

<ModelInstance Id="ModelFields.ObjFieldProvider" Name="ObjFieldProvider"
Implementation="378B9821-00B9-4301-8A61-1B5260A0E3E1">
   <Model xlink:title="ModelFields" xlink:href="./Catalogue.cat#ModelFields" />

...

</ModelInstance>

<ModelInstance Id="ModelFields.ObjFieldConsumer" Name="ObjFieldConsumer"
Implementation="378B9821-00B9-4301-8A61-1B5260A0E3E1">
   <Model xlink:title="ModelFields" xlink:href="./Catalogue.cat#ModelFields" />

   ...

      <Link xsi:type="Assembly:FieldLink" Id="ObjFieldConsumer.inputField11"
Name="inputField11">
       <Input xlink:title="inputField1"
xlink:href="./Catalogue.cat#Catalogue.ModelFields.inputField1" />
       <Source xlink:title="ObjFieldProvider" xlink:href="#ObjFieldProvider" />
       <Output xlink:title="outputField1"
xlink:href="./Catalogue.cat#Catalogue.ModelFields.outputField1" />
      </Link>

      <Link xsi:type="Assembly:FieldLink" Id="ObjFieldConsumer.inputField21"
Name="inputField21">
       <Input xlink:title="inputField2"
xlink:href="./Catalogue.cat#Catalogue.ModelFields.inputField2" />
       <Source xlink:title="ObjFieldProvider" xlink:href="#ObjFieldProvider" />
       <Output xlink:title="outputField2"
xlink:href="./Catalogue.cat#Catalogue.ModelFields.outputField2" />
      </Link>
```

```
      <Link xsi:type="Assembly:FieldLink" Id="ObjFieldConsumer.inputField31"
Name="inputField31">
       <Input xlink:title="inputField3"
xlink:href="./Catalogue.cat#Catalogue.ModelFields.inputField3" />
       <Source xlink:title="ObjFieldProvider" xlink:href="#ObjFieldProvider" />
       <Output xlink:title="outputField3"
xlink:href="./Catalogue.cat#Catalogue.ModelFields.outputField3" />
      </Link>

      <Link xsi:type="Assembly:FieldLink" Id="ObjFieldConsumer.inputField41"
Name="inputField41">
       <Input xlink:title="inputField4"
xlink:href="./Catalogue.cat#Catalogue.ModelFields.inputField4" />
       <Source xlink:title="ObjFieldProvider" xlink:href="#ObjFieldProvider" />
       <Output xlink:title="outputField4"
xlink:href="./Catalogue.cat#Catalogue.ModelFields.outputField4" />
      </Link>

...
```

8. Assembly with interface links

During the production of the assembly, the interface types of the references are not checked. Then, for this test case, the interfaces are linked ignoring the type constraints (see the next image). For the sake of assembly production validation, these restrictions are not critical, but be aware that this assembly will fail in case of building a simulation.



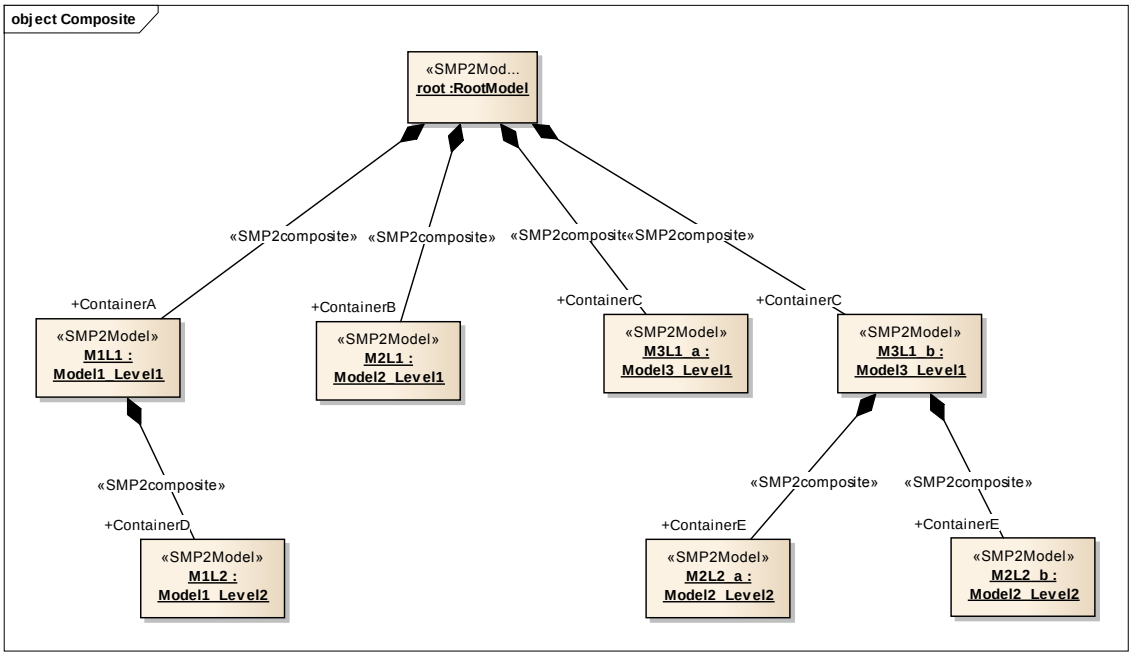
#	Consumer instance	Reference	Provider instance	Provided interface
1	ModelReferenceConsumer	Equipments	ModelReferenceProvider1	IModelInterfaces
2	ModelReferenceConsumer	Equipments	ModelReferenceProvider2	IModelInterfaces
3	ModelReferenceConsumer	Environment	ModelReferenceProvider2	IModelInterfaces

Expected results:

SMP2Assembly

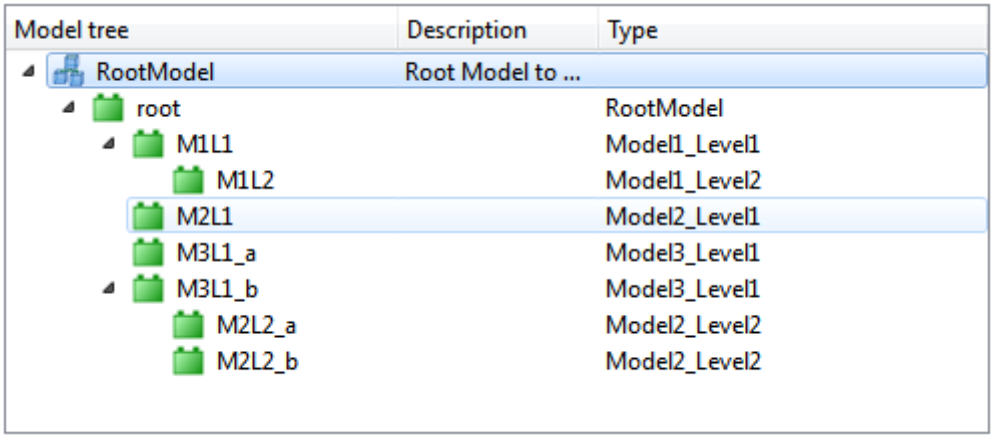
```
<ModelInstance Id="ModelReferences.ModelReferenceConsumer"
            Name="ModelReferenceConsumer" Implementation="1F570F47-BDBC-4619-
            9778-B8883C186509">
     <Model xlink:title="ModelReferences"
            xlink:href="./Catalogue.cat#ModelReferences" />
     <Link xsi:type="Assembly:InterfaceLink"
            Id="ModelReferenceConsumer.Equipments1" Name="Equipments1">
         <Reference xlink:title="Equipments"
            xlink:href="./Catalogue.cat#Catalogue.ModelReferences.Equipments" />
         <Provider xlink:title="ModelReferenceProvider1"
            xlink:href="#ModelReferenceProvider1" />
     </Link>
     <Link xsi:type="Assembly:InterfaceLink"
            Id="ModelReferenceConsumer.Equipments2" Name="Equipments2">
         <Reference xlink:title="Equipments"
            xlink:href="./Catalogue.cat#Catalogue.ModelReferences.Equipments" />
         <Provider xlink:title="ModelReferenceProvider2"
            xlink:href="#ModelReferenceProvider2" />
     </Link>
     <Link xsi:type="Assembly:InterfaceLink"
            Id="ModelReferenceConsumer.Environment1" Name="Environment1">
         <Reference xlink:title="Environment"
            xlink:href="./Catalogue.cat#Catalogue.ModelReferences.Environment"
            />
         <Provider xlink:title="ModelReferenceProvider2"
            xlink:href="#ModelReferenceProvider2" />
     </Link>
     <Container xlink:title="ModelsContainer" xlink:href="#??#"/>
</ModelInstance>

<ModelInstance Id="ModelInterfaces.ModelReferenceProvider1"
            Name="ModelReferenceProvider1" Implementation="1567EAE2-7F4D-4dd6-
            9634-1CD30BB39536">
     <Model xlink:title="ModelInterfaces"
            xlink:href="./Catalogue.cat#ModelInterfaces" />
     <Container xlink:title="ModelsContainer" xlink:href="#??#"/>
</ModelInstance>
<ModelInstance Id="ModelInterfaces.ModelReferenceProvider2"
            Name="ModelReferenceProvider2" Implementation="1567EAE2-7F4D-4dd6-
            9634-1CD30BB39536">
     <Model xlink:title="ModelInterfaces"
            xlink:href="./Catalogue.cat#ModelInterfaces" />
     <Container xlink:title="ModelsContainer" xlink:href="#??#"/>
</ModelInstance>
```

9. Assembly using composite



Model tree (instances)	Model Type	Container
rootModel	RootModel	
M1L1	Model1_Level1	ContainerA
M1L2	Model1_Level2	ContainerD
M2L1	Model2_Level1	ContainerB
M3L1_a	Model3_Level1	ContainerC
M3L1_b	Model3_Level1	ContainerC
M2L2_a	Model2_Level2	ContainerE
M2L2_b	Model2_Level2	ContainerE

The resulted assembly can be loaded in an actual simulation, once the models are compiled in a library. The result will be:



Expected results:

SMP2Assembly

```
<ModelInstance Id="RootModel.root" Name="root" Implementation="FBFA7FA6-0D14-
            4ae7-AE8B-EE9E9D8E2FE5">
  <Model xlink:title="RootModel" xlink:href="./Catalogue.cat#RootModel" />
```

```
  <ModelInstance Id="Model1_Level1.M1L1" Name="M1L1" Implementation="9DA9886F-
            56C3-4511-8251-53DB7B2ACE54">
    <Model xlink:title="Model1_Level1"
            xlink:href="./Catalogue.cat#Model1_Level1" />
    <ModelInstance Id="Model1_Level2.M1L2" Name="M1L2" Implementation="FA49988D-
            1C0B-4088-AFCB-3E6405EA99A0">
      <Model xlink:title="Model1_Level2"
            xlink:href="./Catalogue.cat#Model1_Level2" />
      <Container xlink:title="ContainerD" xlink:href="#??#"/>
    </ModelInstance>
    <Container xlink:title="ContainerA" xlink:href="#??#"/>
  </ModelInstance>
  <ModelInstance Id="Model2_Level1.M2L1" Name="M2L1" Implementation="500CE014-
            0068-44f1-82E2-8088D83D68AC">
    <Model xlink:title="Model2_Level1"
            xlink:href="./Catalogue.cat#Model2_Level1" />
    <Container xlink:title="ContainerB" xlink:href="#??#"/>
  </ModelInstance>
  <ModelInstance Id="Model3_Level1.M3L1_a" Name="M3L1_a"
            Implementation="6A4A8248-CD3F-4979-A427-9A2A9CF7B4DE">
    <Model xlink:title="Model3_Level1"
            xlink:href="./Catalogue.cat#Model3_Level1" />
    <Container xlink:title="ContainerC" xlink:href="#??#"/>
  </ModelInstance>
    <ModelInstance Id="Model3_Level1.M3L1_b" Name="M3L1_b"
            Implementation="6A4A8248-CD3F-4979-A427-9A2A9CF7B4DE">
      <Model xlink:title="Model3_Level1"
            xlink:href="./Catalogue.cat#Model3_Level1" />
      <ModelInstance Id="Model2_Level2.M2L2_a" Name="M2L2_a"
            Implementation="72A03B2B-AFE2-492b-80F2-924C81537E72">
        <Model xlink:title="Model2_Level2"
            xlink:href="./Catalogue.cat#Model2_Level2" />
        <Container xlink:title="ContainerE" xlink:href="#??#"/>
      </ModelInstance>
      <ModelInstance Id="Model2_Level2.M2L2_b" Name="M2L2_b"
            Implementation="72A03B2B-AFE2-492b-80F2-924C81537E72">
        <Model xlink:title="Model2_Level2"
            xlink:href="./Catalogue.cat#Model2_Level2" />
        <Container xlink:title="ContainerE" xlink:href="#??#"/>
      </ModelInstance>
      <Container xlink:title="ContainerC" xlink:href="#??#"/>
    </ModelInstance>
  <Container xlink:title="ModelsContainer" xlink:href="#??#"/>
</ModelInstance>
```