

ADMIRE D1.3 – ADMIRE DMI Model Prototype

DMI-WF: XML-Schema Specification Version 1.0

Project Title	ADMIRE
Document Title	ADMIRE DMI Model Prototype DMI-WF: XML-Schema Specification Version 1.0
Deliverable Number	D1.3
Authorship	Peter Brezany, Ivan Janciak, Sabri Pllana
Document Filename	ADMIRE-D1.3-Research prototypes
Document Version	1.0
Distribution Classification	Public
Distribution List	ADMIRE Project Team
Approval List	Amy Krause, Project Manager, Executive Board

History

<i>Personnel</i>	<i>Date</i>	<i>Comment</i>	<i>Version</i>
Ivan Janciak	26/01/2009	First draft	0.1
Ivan Janciak	16/02/2009	XML-Schema	0.2
Sabri Pllana	25/02/2009	Text tweaks	0.3
Ivan Janciak	26/02/2009	Second version	0.4
Peter Brezany	27/02/2009	Final version	0.5
Rob Baxter	02/03/2009	Final edit and signoff	1.0

Contents

1	Introduction	3
2	XML-Schema Documentation	4
2.1	Schema Document Properties	4
2.2	Global Declarations	4
2.3	Global Definitions	11
3	Glossary	18
	References	19

1 Introduction

This document describes the first version of the ADMIRE DMI-WF XML-Schema. The usage of the XML-Schema profile is two fold: (1) it captures the set of concepts defined in the DMI meta-model [2], (2) it defines elements of the DMI workflow language (DMI-WF). The DMI-WF is an XML-based language, which describes phases of a DMI project as defined by the CRISP-DMI reference model introduced in Deliverable D1.1 [3]. The DMI-WF is intended to be programatically transformed to the DMI Language (DMI-L) [1].

Basically, the DMI-WF defines a directed graph whose nodes represent activities that refer to data mining and integration tasks realized within one DMI project. These tasks are implemented by Processing Elements (PE) which are connected by data-flow links. There are two kinds of processing elements: atomic elements `<elements/>` and composite elements `<processes/>`. Both are identified by a unique ID within the scope of a DMI project and are described by their inputs and outputs, which are used to specify the data-flow between them. Control-flow is specified through explicit links among elements. Typically, in the DMI domain, data-flow also determines control-flow.

The DMI-WF documents are produced by the ADMIRE Process Designer using its workflow editor tool [6], which has been designed to allow DMI experts to concentrate on describing various data processing applications running on the ADMIRE Platform [5]. These experts are shielded from the details of the underlying platform and construct workflows on the higher level of abstraction described by the CRISP-DMI reference model [3].

The Process Designer provides a graphical environment for composition of the Processing Elements and their parameterization. As a result, the tool produces a process description in XML format defined by the DMI-WF schema introduced in this document. According to the ADMIRE architecture, this XML process description is used as input to the ADMIRE Gateway for further optimization and its transformation into an executable format. More details about the tool can be found in Deliverable D5.2 [6]. While the development of the tool follows the Model-Driven architecture (MDA) approach for the development of software systems, the XML-Schema described in this document plays a central role in this development process.

The DMI-WF is currently used to describe the ADMIRE pilot applications: the Flood and Analytical CRM applications (see Deliverable D6.2 [4]). The aim is to evaluate the expressiveness of the DMI-WF language using real world DMI applications. The results of the evaluation will be used for further refinement of the XML-Schema in subsequent versions.

2 XML-Schema Documentation

2.1 Schema Document Properties

Target Namespace	http://dmi-wf.admire-project.eu
Element and Attribute Namespaces	<ul style="list-style-type: none"> • Global element and attribute declarations belong to this schema's target namespace. • By default, local element declarations have no namespace. • By default, local attribute declarations have no namespace.
Schema Composition	<p>This schema imports schema(s) from the following namespace(s):</p> <ul style="list-style-type: none"> • http://www.omg.org/XMI (at platform:/plugin/org.eclipse.emf.ecore/model/XMI.xsd) • http://www.eclipse.org/uml2/schemas/Standard/1 (at StandardXMI.xsd) • http://www.eclipse.org/uml2/2.1.0/UML (at umlXMI.xsd)

Declared Namespaces

Prefix	Namespace
xml	http://www.w3.org/XML/1998/namespace
DMIProfile	http://dmi-wf.admire-project.eu
Standard	http://www.eclipse.org/uml2/schemas/Standard/1
uml	http://www.eclipse.org/uml2/2.1.0/UML
xmi	http://www.omg.org/XMI
xsd	http://www.w3.org/2001/XMLSchema

Schema Component Representation

```
<xsd:schema targetNamespace="http://DMIProfile.ecore">
  <xsd:import namespace="http://www.omg.org/XMI" schemaLocation="platform:/plugin
/org.eclipse.emf.ecore/model/XMI.xsd"/>
  <xsd:import namespace="http://www.eclipse.org/uml2/schemas/Standard/1"
schemaLocation="StandardXMI.xsd"/>
  <xsd:import namespace="http://www.eclipse.org/uml2/2.1.0/UML" schemaLocation="umlXMI.xsd"/>
  ...
</xsd:schema>
```

2.2 Global Declarations

XML-Schema allows one to declare elements immediately under the **xsd:schema** element or as part of another declaration. Declarations that appear directly under this root element have a global scope while other declarations have a local scope. The most noticeable difference between global and local declarations of elements is that global declarations can be referenced from other declarations, whereas local declarations only exist within their local scope it means that the global declarations must be unique. Local declarations, on the other hand, cause no conflict if they have different contexts.

Element: DMICConnection

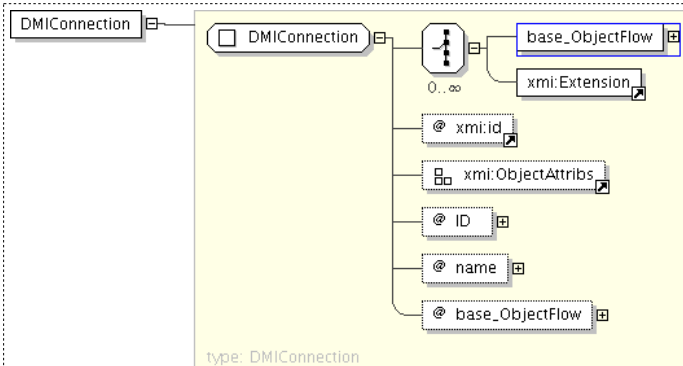
Name	DMICConnection
Type	DMIPProfile:DMICConnection
Nilable	no
Abstract	no

XML Instance Representation

```
<DMIPProfile:DMICConnection
  xmi:id="[0..1]"
  Attribute group reference (not shown): xmi:ObjectAttribs

  ID=" xsd:string [1]"
  name=" xsd:string [1]"
  base_ObjectFlow=" xsd:string [0..1]">
  Start Choice [0..*]
    <base_ObjectFlow> uml:ObjectFlow </base_ObjectFlow> [1]
    <xmi:Extension> ... </xmi:Extension> [1]
  End Choice
</DMIPProfile:DMICConnection>
```

Diagram



Schema Component Representation

```
<xsd:element name="DMICConnection" type=" DMIPProfile:DMICConnection " />
```

Element: DMIElement

Name	DMIElement
Type	DMIProfile:DMIElement
Nilable	no
Abstract	no

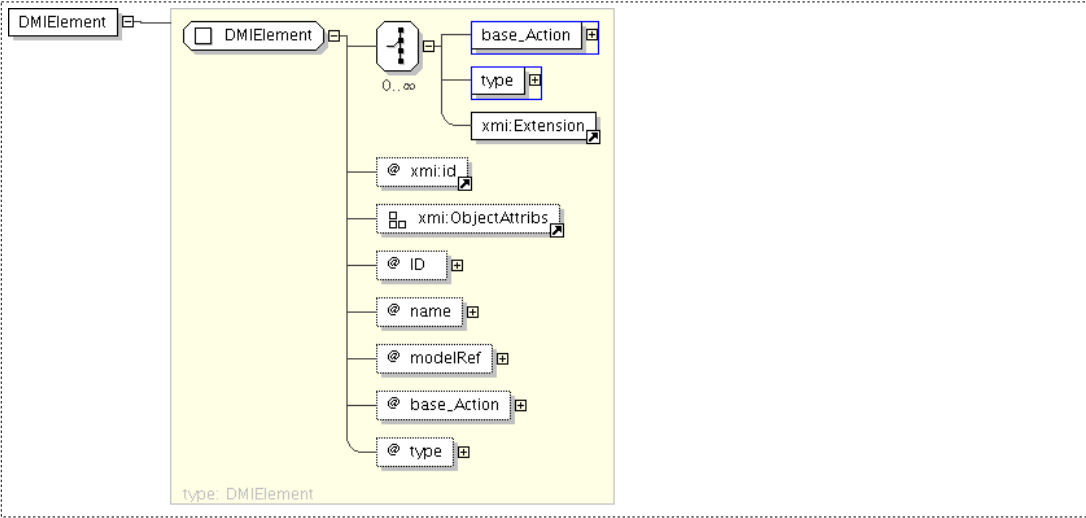
XML Instance Representation

```

<DMIProfile:DMIElement
  xmi:id="[0..1]"
  Attribute group reference (not shown): xmi:ObjectAttribs

  ID=" xsd:string [1]"
  name=" xsd:string [1]"
  modelRef=" xsd:string [1]"
  base_Action=" xsd:string [0..1]"
  type=" xsd:string [0..1]">
  Start Choice [0..*]
    <base_Action> uml:Action </base_Action> [1]
    <type> Standard:Type </type> [1]
    <xmi:Extension> ... </xmi:Extension> [1]
  End Choice
</DMIProfile:DMIElement>
    
```

Diagram



Schema Component Representation

```

<xsd:element name="DMIElement" type=" DMIProfile:DMIElement " />
    
```

Element: DMIInput

Name	DMIInput
Type	DMIProfile:DMIInput
Nilable	no
Abstract	no

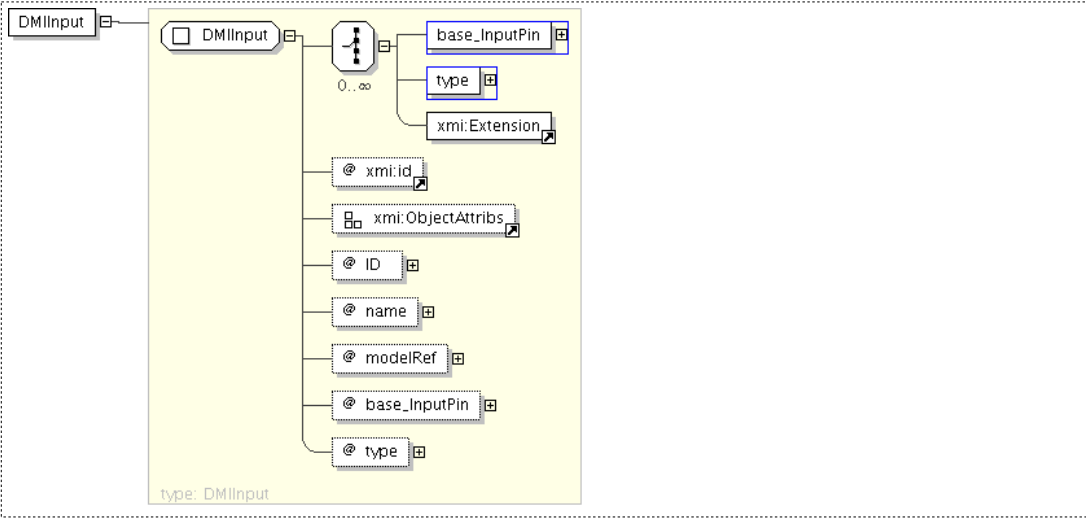
XML Instance Representation

```

<DMIProfile:DMIInput
  xmi:id="[0..1]"
  Attribute group reference (not shown): xmi:ObjectAttribs

  ID=" xsd:string [1]"
  name=" xsd:string [1]"
  modelRef=" xsd:string [1]"
  base_InputPin=" xsd:string [0..1]"
  type=" xsd:string [0..1]">
  Start Choice [0..*]
    <base_InputPin> uml:InputPin </base_InputPin> [1]
    <type> Standard:Type </type> [1]
    <xmi:Extension> ... </xmi:Extension> [1]
  End Choice
</DMIProfile:DMIInput>
    
```

Diagram



Schema Component Representation

```

<xsd:element name="DMIInput" type=" DMIProfile:DMIInput "/>
    
```

Element: DMIOutput

Name	DMIOutput
Type	DMIProfile:DMIOutput
Nilable	no
Abstract	no

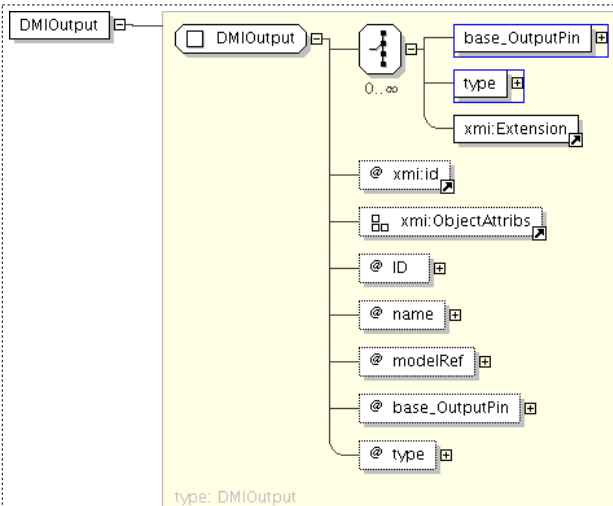
XML Instance Representation

```

<DMIProfile:DMIOutput
  xmi:id="[0..1]"
  Attribute group reference (not shown): xmi:ObjectAttribs

  ID=" xsd:string [1]"
  name=" xsd:string [1]"
  modelRef=" xsd:string [1]"
  base_OutputPin=" xsd:string [0..1]"
  type=" xsd:string [0..1]">
  Start Choice [0..*]
    <base_OutputPin> uml:OutputPin </base_OutputPin> [1]
    <type> Standard:Type </type> [1]
    <xmi:Extension> ... </xmi:Extension> [1]
  End Choice
</DMIProfile:DMIOutput>

```

Diagram**Schema Component Representation**

```

<xsd:element name="DMIOutput" type=" DMIProfile:DMIOutput " />

```

Element: DMIPProcess

Name	DMIPProcess
Type	DMIPProfile:DMIPProcess
Nilable	no
Abstract	no

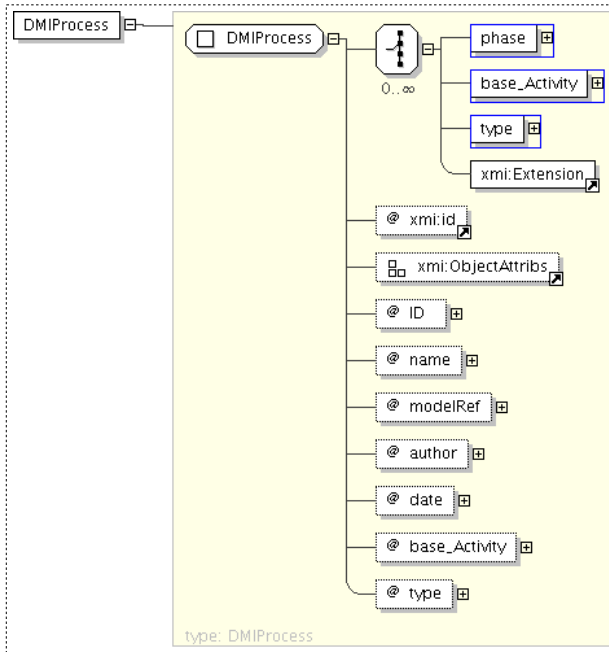
XML Instance Representation

```

<DMIPProfile:DMIPProcess
  xmi:id="[0..1]"
  Attribute group reference (not shown): xmi:ObjectAttribs

  ID=" xsd:string [1]"
  name=" xsd:string [1]"
  modelRef=" xsd:string [1]"
  author=" xsd:string [1]"
  date=" xsd:string [1]"
  base_Activity=" xsd:string [0..1]"
  type=" xsd:string [0..1]">
  Start Choice [0..*]
    <phase> DMIPProfile:DMIPPhase </phase> [1]
    <base_Activity> uml:Activity </base_Activity> [1]
    <type> Standard:Type </type> [1]
    <xmi:Extension> ... </xmi:Extension> [1]
  End Choice
</DMIPProfile:DMIPProcess>

```

Diagram**Schema Component Representation**

```
<xsd:element name="DMIPProcess" type=" DMIPProfile:DMIPProcess " />
```

Element: DMIProject

Name	DMIProject
Type	DMIProfile:DMIProject
Nilable	no
Abstract	no

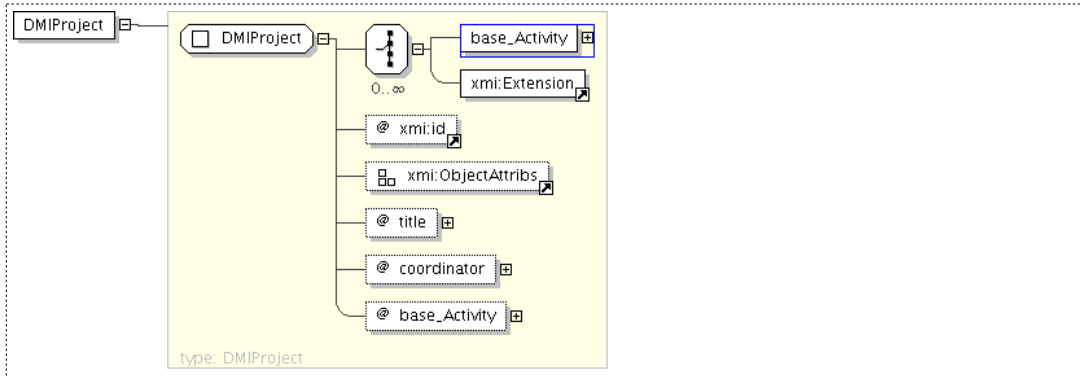
XML Instance Representation

```

<DMIProfile:DMIProject
  xmi:id="[0..1]"
  Attribute group reference (not shown): xmi:ObjectAttribs

  title=" xsd:string [1]"
  coordinator=" xsd:string [1]"
  base_Activity=" xsd:string [0..1]">
  Start Choice [0..*]
    <base_Activity> uml:Activity </base_Activity> [1]
    <xmi:Extension> ... </xmi:Extension> [1]
  End Choice
</DMIProfile:DMIProject>

```

Diagram**Schema Component Representation**

```

<xsd:element name="DMIProject" type=" DMIProfile:DMIProject " />

```

2.3 Global Definitions

Complex Type: [DMICConnection](#)

Super-types:	None
Sub-types:	None

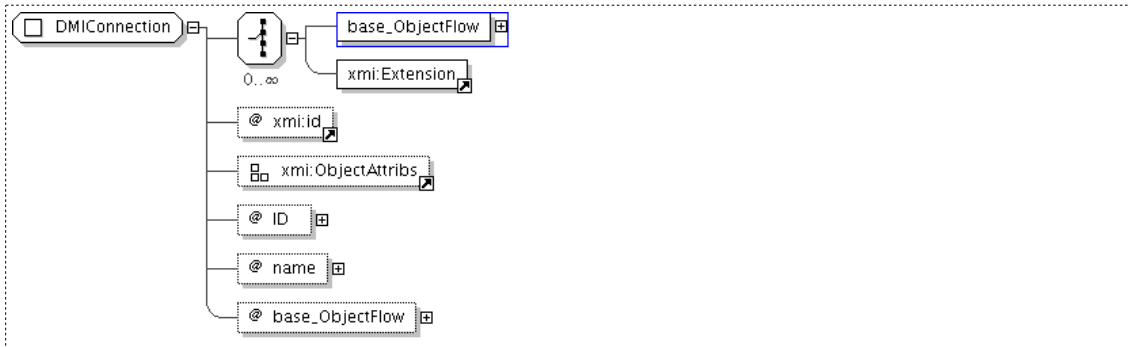
Name	DMICConnection
Used by (from the same schema document)	Element DMICConnection
Abstract	no

XML Instance Representation

```
<...
  xmi:id="[0..1]"
  Attribute group reference (not shown): xmi:ObjectAttribs

  ID=" xsd:string [1]"
  name=" xsd:string [1]"
  base_ObjectFlow=" xsd:string [0..1]">
  Start Choice [0..*]
    <base_ObjectFlow> uml:ObjectFlow </base_ObjectFlow> [1]
    <xmi:Extension> ... </xmi:Extension> [1]
  End Choice
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="DMICConnection">
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
    <xsd:element name="base_ObjectFlow" type="uml:ObjectFlow" />
    <xsd:element ref="xmi:Extension" />
  </xsd:choice>
  <xsd:attribute ref="xmi:id" />
  <xsd:attributeGroup ref="xmi:ObjectAttribs" />
  <xsd:attribute name="ID" type="xsd:string" use="required"/>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="base_ObjectFlow" type="xsd:string" />
</xsd:complexType>
```

Complex Type: DMIElement

Super-types:	None
Sub-types:	None

Name	DMIElement
Used by (from the same schema document)	Element DMIElement
Abstract	no

XML Instance Representation

```
<...
  xmi:id=" [0..1]"
  Attribute group reference (not shown): xmi:ObjectAttribs

  ID=" xsd:string [1]"
  name=" xsd:string [1]"
  modelRef=" xsd:string [1]"
  base_Action=" xsd:string [0..1]"
  type=" xsd:string [0..1]">
  Start Choice [0..*]
    <base_Action> uml:Action </base_Action> [1]
    <type> Standard:Type </type> [1]
    <xmi:Extension> ... </xmi:Extension> [1]
  End Choice
</...>
```

Diagram**Schema Component Representation**

```
<xsd:complexType name="DMIElement">
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
    <xsd:element name="base_Action" type="uml:Action" />
    <xsd:element name="type" type="Standard:Type" />
    <xsd:element ref="xmi:Extension" />
  </xsd:choice>
  <xsd:attribute ref="xmi:id" />
  <xsd:attributeGroup ref="xmi:ObjectAttribs" />
  <xsd:attribute name="ID" type="xsd:string" use="required"/>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="modelRef" type="xsd:string" use="required"/>
  <xsd:attribute name="base_Action" type="xsd:string" />
  <xsd:attribute name="type" type="xsd:string" />
</xsd:complexType>
```

Complex Type: DMIInput

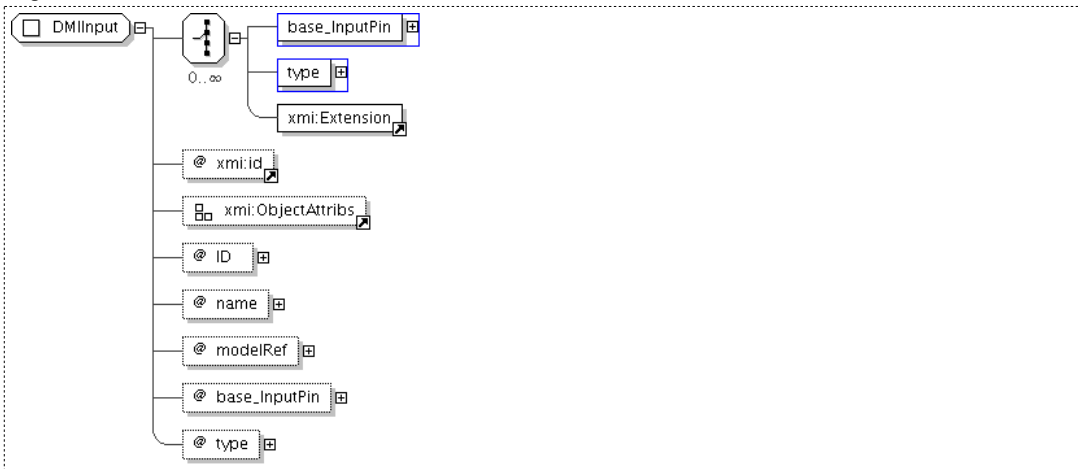
Super-types:	None
Sub-types:	None

Name	DMIInput
Used by (from the same schema document)	Element DMIInput
Abstract	no

XML Instance Representation

```
<...
  xmi:id="[0..1]"
  Attribute group reference (not shown): xmi:ObjectAttribs

  ID=" xsd:string [1]"
  name=" xsd:string [1]"
  modelRef=" xsd:string [1]"
  base_InputPin=" xsd:string [0..1]"
  type=" xsd:string [0..1]">
  Start Choice [0..*]
    <base_InputPin> uml:InputPin </base_InputPin> [1]
    <type> Standard:Type </type> [1]
    <xmi:Extension> ... </xmi:Extension> [1]
  End Choice
</...>
```

Diagram**Schema Component Representation**

```
<xsd:complexType name="DMIInput">
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
    <xsd:element name="base_InputPin" type="uml:InputPin" />
    <xsd:element name="type" type="Standard:Type" />
    <xsd:element ref="xmi:Extension" />
  </xsd:choice>
  <xsd:attribute ref="xmi:id" />
  <xsd:attributeGroup ref="xmi:ObjectAttribs" />
  <xsd:attribute name="ID" type="xsd:string" use="required"/>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="modelRef" type="xsd:string" use="required"/>
  <xsd:attribute name="base_InputPin" type="xsd:string" />
  <xsd:attribute name="type" type="xsd:string" />
</xsd:complexType>
```

Complex Type: DMIOutput

Super-types:	None
Sub-types:	None

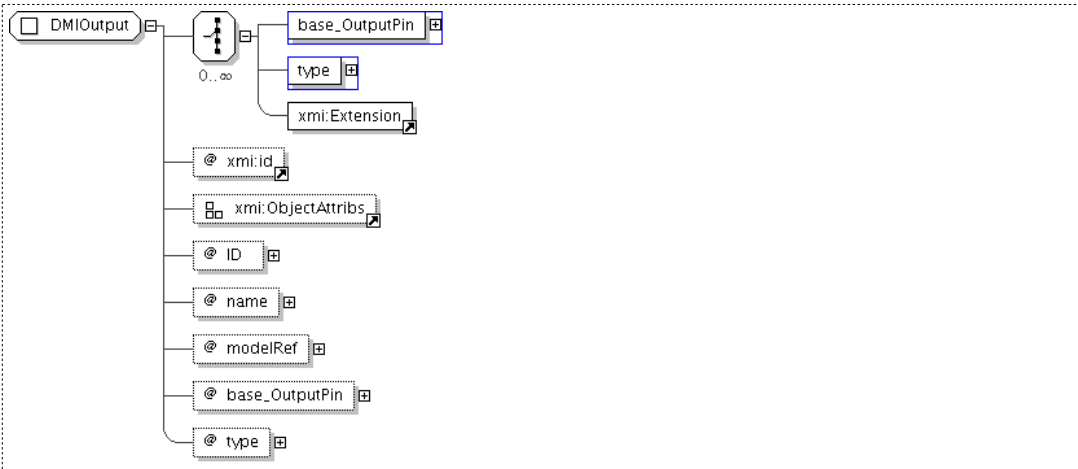
Name	DMIOutput
Used by (from the same schema document)	Element DMIOutput
Abstract	no

XML Instance Representation

```
<...
  xmi:id="[0..1]"
  Attribute group reference (not shown): xmi:ObjectAttribs

  ID=" xsd:string [1]"
  name=" xsd:string [1]"
  modelRef=" xsd:string [1]"
  base_OutputPin=" xsd:string [0..1]"
  type=" xsd:string [0..1]">
  Start Choice [0..*]
    <base_OutputPin> uml:OutputPin </base_OutputPin> [1]
    <type> Standard:Type </type> [1]
    <xmi:Extension> ... </xmi:Extension> [1]
  End Choice
</...>
```

Diagram



Schema Component Representation

```
<xsd:complexType name="DMIOutput">
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
    <xsd:element name="base_OutputPin" type="uml:OutputPin" />
    <xsd:element name="type" type="Standard:Type" />
    <xsd:element ref="xmi:Extension" />
  </xsd:choice>
  <xsd:attribute ref="xmi:id" />
  <xsd:attributeGroup ref="xmi:ObjectAttribs" />
  <xsd:attribute name="ID" type="xsd:string" use="required"/>
  <xsd:attribute name="name" type="xsd:string" use="required"/>
  <xsd:attribute name="modelRef" type="xsd:string" use="required"/>
  <xsd:attribute name="base_OutputPin" type="xsd:string" />
  <xsd:attribute name="type" type="xsd:string" />
</xsd:complexType>
```

Complex Type: DMIProcess

Super-types:	None
Sub-types:	None

Name	DMIProcess
Used by (from the same schema document)	Element DMIProcess
Abstract	no

XML Instance Representation

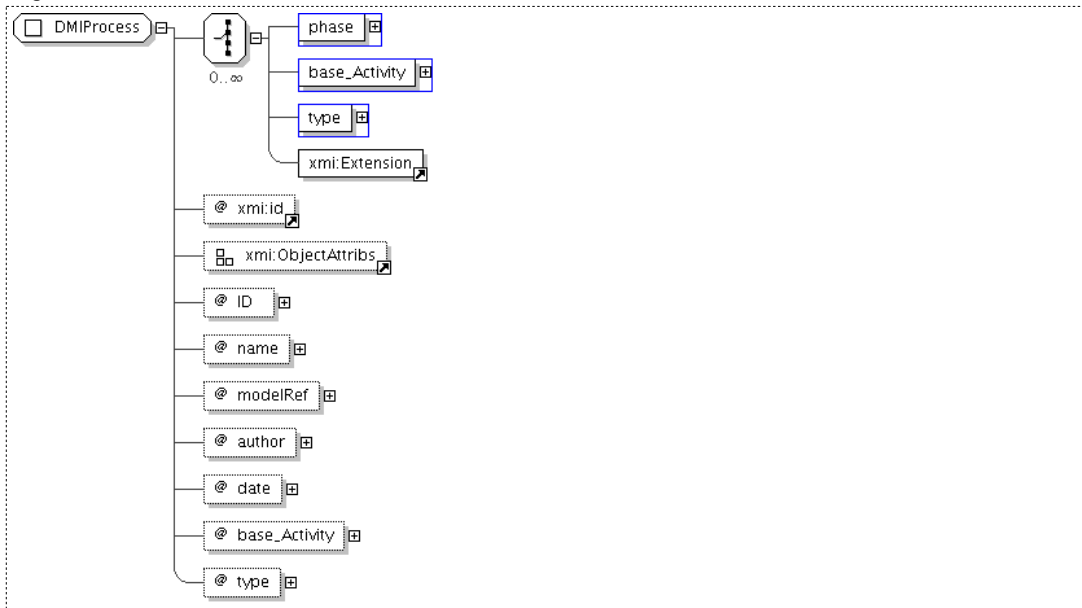
```

<...
  xmi:id="[0..1]"
  Attribute group reference (not shown): xmi:ObjectAttribs

  ID=" xsd:string [1]"
  name=" xsd:string [1]"
  modelRef=" xsd:string [1]"
  author=" xsd:string [1]"
  date=" xsd:string [1]"
  base_Activity=" xsd:string [0..1]"
  type=" xsd:string [0..1]">
    Start Choice [0..*]
      <phase> DMIProfile:DMIPhase </phase> [1]
      <base_Activity> uml:Activity </base_Activity> [1]
      <type> Standard:Type </type> [1]
      <xmi:Extension> ... </xmi:Extension> [1]
    End Choice
  </...>

```

Diagram



Schema Component Representation

```

<xsd:complexType name="DMIProcess">
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
    <xsd:element name="phase" type=" DMIProfile:DMIPhase "/>
    <xsd:element name="base_Activity" type=" uml:Activity "/>
    <xsd:element name="type" type=" Standard:Type "/>
    <xsd:element ref=" xmi:Extension "/>
  </xsd:choice>
  <xsd:attribute ref=" xmi:id "/>
  <xsd:attributeGroup ref=" xmi:ObjectAttribs "/>
  <xsd:attribute name="ID" type=" xsd:string " use="required"/>
  <xsd:attribute name="name" type=" xsd:string " use="required"/>
  <xsd:attribute name="modelRef" type=" xsd:string " use="required"/>
  <xsd:attribute name="author" type=" xsd:string " use="required"/>
  <xsd:attribute name="date" type=" xsd:string " use="required"/>
  <xsd:attribute name="base_Activity" type=" xsd:string "/>
  <xsd:attribute name="type" type=" xsd:string "/>
</xsd:complexType>

```

Complex Type: [DMIProject](#)

Super-types:	None
Sub-types:	None

Name	DMIProject
Used by (from the same schema document)	Element DMIProject
Abstract	no

XML Instance Representation

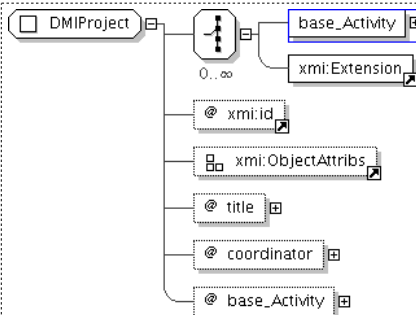
```

<...
  xmi:id="[0..1]"
  Attribute group reference (not shown): xmi:ObjectAttribs

  title=" xsd:string [1]"
  coordinator=" xsd:string [1]"
  base_Activity=" xsd:string [0..1]">
  Start Choice [0..*]
    <base_Activity> uml:Activity </base_Activity> [1]
    <xmi:Extension> ... </xmi:Extension> [1]
  End Choice
</...>

```

Diagram



Schema Component Representation

```

<xsd:complexType name="DMIProject">
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
    <xsd:element name="base_Activity" type=" uml:Activity "/>
    <xsd:element ref=" xmi:Extension "/>
  </xsd:choice>
  <xsd:attribute ref=" xmi:id "/>
  <xsd:attributeGroup ref=" xmi:ObjectAttribs "/>
  <xsd:attribute name="title" type=" xsd:string " use="required"/>
  <xsd:attribute name="coordinator" type=" xsd:string " use="required"/>
  <xsd:attribute name="base_Activity" type=" xsd:string "/>
</xsd:complexType>

```

Simple Type: DMIPhase

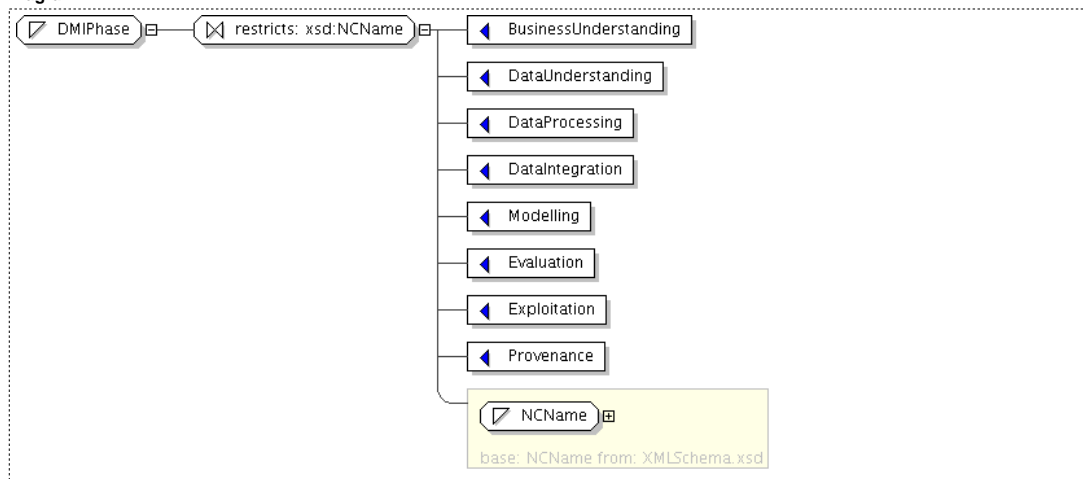
Super-types: [xsd:NCName](#) < **DMIPhase** (by restriction)
Sub-types: None

Name DMIPhase

**Used by
(from the
same
schema
document)**
Complex Type [DMIProcess](#)

Content

- Base XSD Type: NCName
- *value* comes from list:
{'BusinessUnderstanding'|'DataUnderstanding'|'DataProcessing'|'DataIntegration'|'Modelling'|'Evaluation'|'Exploitation'|'Provenance'}

Diagram**Schema Component Representation**

```

<xsd:simpleType name="DMIPhase">
  <xsd:restriction base="xsd:NCName">
    <xsd:enumeration value="BusinessUnderstanding"/>
    <xsd:enumeration value="DataUnderstanding"/>
    <xsd:enumeration value="DataProcessing"/>
    <xsd:enumeration value="DataIntegration"/>
    <xsd:enumeration value="Modelling"/>
    <xsd:enumeration value="Evaluation"/>
    <xsd:enumeration value="Exploitation"/>
    <xsd:enumeration value="Provenance"/>
  </xsd:restriction>
</xsd:simpleType>

```

3 Glossary

- **Abstract** (Applies to complex type definitions and element declarations). An abstract element or complex type cannot be used to validate an element instance. If there is a reference to an abstract element, only element declarations that can substitute the abstract element can be used to validate the instance. For references to abstract type definitions, only derived types can be used.
- **All Model Group** Child elements can be provided in any order in instances.
See: <http://www.w3.org/TR/xmlschema-1/#element-all>.
- **Choice Model Group** Only one from the list of child elements and model groups can be provided in instances. See: <http://www.w3.org/TR/xmlschema-1/#element-choice>.
- **Collapse Whitespace Policy** Replace tab, line feed, and carriage return characters with space character (Unicode character 32). Then, collapse contiguous sequences of space characters into single space character, and remove leading and trailing space characters.
- **Disallowed Substitutions** (Applies to element declarations). If substitution is specified, then substitution group members cannot be used in place of the given element declaration to validate element instances. If derivation methods, e.g. extension, restriction, are specified, then the given element declaration will not validate element instances that have types derived from the element declaration's type using the specified derivation methods. Normally, element instances can override their declaration's type by specifying an `xsi:type` attribute.
- **Key Constraint** Like Uniqueness Constraint, but additionally requires that the specified value(s) must be provided.
See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.
- **Key Reference Constraint** Ensures that the specified value(s) must match value(s) from a Key Constraint or Uniqueness Constraint.
See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.
- **Model Group** Groups together element content, specifying the order in which the element content can occur and the number of times the group of element content may be repeated.
See: http://www.w3.org/TR/xmlschema-1/#Model_Groups.
- **Nillable** (Applies to element declarations). If an element declaration is nillable, instances can use the `xsi:nil` attribute. The `xsi:nil` attribute is the boolean attribute, `nil`, from the <http://www.w3.org/2001/XMLSchema-instance> namespace. If an element instance has an `xsi:nil` attribute set to true, it can be left empty, even though its element declaration may have required content.
- **Notation** A notation is used to identify the format of a piece of data. Values of elements and attributes that are of type, NOTATION, must come from the names of declared notations.
See: http://www.w3.org/TR/xmlschema-1/#cNotation_Declarations.
- **Preserve Whitespace Policy** Preserve whitespaces exactly as they appear in instances.
- **Prohibited Derivations** (Applies to type definitions). Derivation methods that cannot be used to create sub-types from a given type definition.
- **Prohibited Substitutions** (Applies to complex type definitions). Prevents sub-types that have been derived using the specified derivation methods from validating element instances in place of the given type definition.
- **Replace Whitespace Policy** Replace tab, line feed, and carriage return characters with space character (Unicode character 32).

- **Sequence Model Group** Child elements and model groups must be provided in the specified order in instances.
See: <http://www.w3.org/TR/xmlschema-1/#element-sequence>.
- **Substitution Group** Elements that are members of a substitution group can be used wherever the head element of the substitution group is referenced.
- **Substitution Group Exclusions** (Applies to element declarations). Prohibits element declarations from nominating themselves as being able to substitute a given element declaration, if they have types that are derived from the original element's type using the specified derivation methods.
- **Target Namespace** The target namespace identifies the namespace that components in this schema belongs to. If no target namespace is provided, then the schema components do not belong to any namespace.
- **Uniqueness Constraint** Ensures uniqueness of an element/attribute value, or a combination of values, within a specified scope.
See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

References

- [1] Malcolm Atkinson, Peter Brezany, Oscar Corcho, Liangxiu Han, Jano van Hemert, Ladislav Hluchý, Ally Hume, Ivan Janciak, Amy Krause, and Dave Snelling. ADMIRE White Paper: Motivation, Strategy, Overview and Impact. Technical Report v0.9, the ADMIRE Project, January 2009.
- [2] Peter Brezany, Carlos Buil, Ivan Janciak, and Sabri Pllana. ADMIRE – DMI Model, Language and Ontology. Deliverable report D1.2, the ADMIRE Project, Feb 2009.
- [3] Peter Brezany, Ehtesham ul Haq Dar, Ibrahim Elsayed, Yuzhang Han, Ivan Janciak, Fakhri Alam Khan, Sabri Pllana, Yuan Tian, Alexander Wöhrer, Malcolm Atkinson, Jano van Hemert, Oscar Corcho, Carlos Buil Aranda, Marian Babik, and Rob Baxter. ADMIRE – Towards the High-Level DMI Model, Language and Ontology. Deliverable report D1.1, the ADMIRE Project, August 2008.
- [4] Ondrej Habala, Rafał Gąsiorowski, Karol Strzelecki, Maciej Gańczyk, and Viet Tran. ADMIRE – Report on Validation of Initial ADMIRE Model and Architecture. Deliverable report D6.2, the ADMIRE Project, Feb 2009.
- [5] Ally Hume, Liangxiu Han, Jano van Hemert, and Malcolm Atkinson. ADMIRE – Architecture. Deliverable report D2.1, the ADMIRE Project, Aug 2008.
- [6] Amy Krause, Carlos Buil, Rafał Gąsiorowski, Branislav Simo, Michal Laclavik, Ivan Janciak, and Rob Baxter. ADMIRE – Tools Development Report and Requirements Analysis. Deliverable report D5.2, the ADMIRE Project, Feb 2009.