XML 架构支持

1 简介

通过解释 XML 架构定义 (XSD) 文件，Mendix 派生用于 XML 导入/导出以及调用 SOAP/XML Web 服务的输入/输出格式。使用 Mendix Studio Pro 导入 XML 架构（.xsd 文件）或 Web 服务定义（.wsdl 文件）时，可能会遇到一个对话框，其中包含有关不支持构造的警告消息。这是因为 Mendix 目前不支持整个 XSD 标准。Mendix 中的映射基于实体和属性，而一些 XSD 构造不能轻松适合此格式。下表显示了 Mendix 中支持哪些 XSD 构造。

XSD 构造	支持
group（组）	是
sequence（序列）	仅当其恰好出现一次时
choice（选择）	仅当各个选项不是序列元素时
unique（唯一）	是
attributeGroup（属性组）	是
全部	仅当 **all** 的每个子元素最多出现一次时
union（并集）	是
any（任意）	否
anyAttribute（任意属性）	否
list（列表）	否

Mendix 中有两种 XML 映射：导入映射（将 XML 数据转换为 Mendix 对象）和导出映射（与前者正好相反）。导入映射在通过微流中的“导入 XML”活动导入 XML 文件时使用，旨在处理 Web 服务调用的响应。导出映射用于导出微流中的 XML 文件，以及为 Web 服务调用的请求生成 XML。

在导入映射的映射编辑器中，可根据 XSD 检查 XML 中可能出现的元素，然后为这些元素定义到 Mendix 对象的映射。元素出现的频率或顺序并不重要；每当一个元素出现时，系统都会应用该元素的相应映射。

如果 XSD 包含任何不支持的构造，则它们将在映射编辑器中高亮显示，并带有该警告图标：


该警告图标位于每个不支持的元素或属性旁边。选中任何此类元素或属性将导致一致性错误。

使用导出映射生成的 XML 必须严格遵循 XSD 规范。这意味着，由映射生成的 XML 标记必须符合 XSD 所指定的确切顺序。

2 元素和类型

XSD 定义了与 XML 中标记对应的元素。元素具有定义其内容的类型，例如基元值或子元素列表。元素可以具有简单类型或复杂类型，而复杂类型可包含简单内容或复杂内容。对于包含简单内容的简单类型和复杂类型，元素的内容都是基元值，例如整型或字符串值。对于包含复杂内容的复杂类型，元素的内容可以由若干子元素组成。复杂类型还能定义可以在 XML 标记内出现的属性。

以下示例显示了一个 XML 架构和一个遵循该架构的 XML 实例。该架构定义了一个具有复杂类型的“客户”元素，该复杂类型包含复杂内容，即“名字”和“鞋码”元素的序列。“名字”元素具有简单类型，即“字符串”。“鞋码”元素具有包含简单内容的复杂类型；它通过添加“国家/地区”属性扩展了简单类型“整型”。

XML 架构示例

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetnamespace=""http://www.example.com/"" elementformdefault=""qualified"" xmlns:tns=""http://www.example.com/"" xmlns:xs=""http://www.w3.org/2001/XMLSchema"">
<xs:element name=""customer"">
<xs:complexType>
<xs:sequence>
<xs:element name=""name"" type=""xs:string""/">
<xs:element name=""shoesize"" type=""tns:shoesizeType""/">
 </xs:element></xs:element></xs:sequence>
 </xs:complexType>

<xs:complexType name=""shoesizeType"">
<xs:simpleContent>
<xs:extension base=""xs:integer"">
<xs:attribute name=""country"" type=""xs:string""/">
```

</xs:attribute></xs:extension></xs:simpleContent>

</xs:complexType> </xs:element> </xs:schema>

XML 实例示例

```
<?xml version="1.0" encoding="utf-8"?>
<customer xmlns="http://www.example.com/">
  <name>John Doe</name>
  <shoesize country="GB">8</shoesize>
</customer>
```

由于 Mendix 中的 XML 映射基于由实体、属性和关联组成的域模型，因此创建映射时，XSD 元素将转换为一种对象模型。具有复杂类型或多次出现的元素会转换为映射到实体的所谓“对象元素”。具有简单类型且最多出现一次的元素会转换为“值元素”，这些元素会映射到其包含对象元素的实体属性。复杂类型的 XML 属性也会转换为值元素。

下图显示了上文示例的 XML 架构如何转换为 Mendix 中的映射。它是示例中“客户”元素的 XML 到域映射。“客户”元素转换为对象元素（灰色矩形），而“名称”和“鞋码”元素以及“鞋码”元素的“国家/地区”属性则转换为映射中的值元素。

3 属性

复杂类型能指定可在 XML 标记内出现的属性。这些属性始终具有简单类型，且最多只能出现一次。两种映射类型都完全支持属性。属性转换为映射中的值元素，因此可映射到实体的属性。

4 简单类型和包含简单内容的复杂类型

Mendix 完全支持包含基元内容的元素，例如简单类型或包含简单内容的复杂类型。但是，Mendix 目前并未考虑对简单类型的限制，例如将字符串限制为可能性的有限集或限制整型的范围。在这些情况下，允许基本类型（例如字符串）的所有可能值。

通常，包含基元内容的元素会转换为映射中的值元素。如果由于某些原因，包含基元内容的元素转换为映射中的对象元素，例如，因为它多次出现，或因为它具有同样定义了属性的复杂类型，则该元素的内容将转换为名为“(Contents)”的额外值元素。

十六进制二进制类型被识别为字符串，因此对二进制类型的操作不适用于十六进制二进制类型（例如 MTOM 附件）。

5 包含复杂内容的复杂类型

对于具有复杂类型且其复杂类型包含复杂内容的元素，该元素的内容可由若干子元素组成，这些子元素的顺序和多重性由一个或多个分组构造（例如选择和序列）定义。

6 混合内容

复杂类型可以将其内容定义为混合型。这意味着，元素同时具有文本和子元素作为内容。混合内容常见于 (X)HTML 等文档数据中，而在结构化数据中则比较少见。目前，Mendix 不支持混合内容。