SODIUS WILLERT

The Power of Connected Data

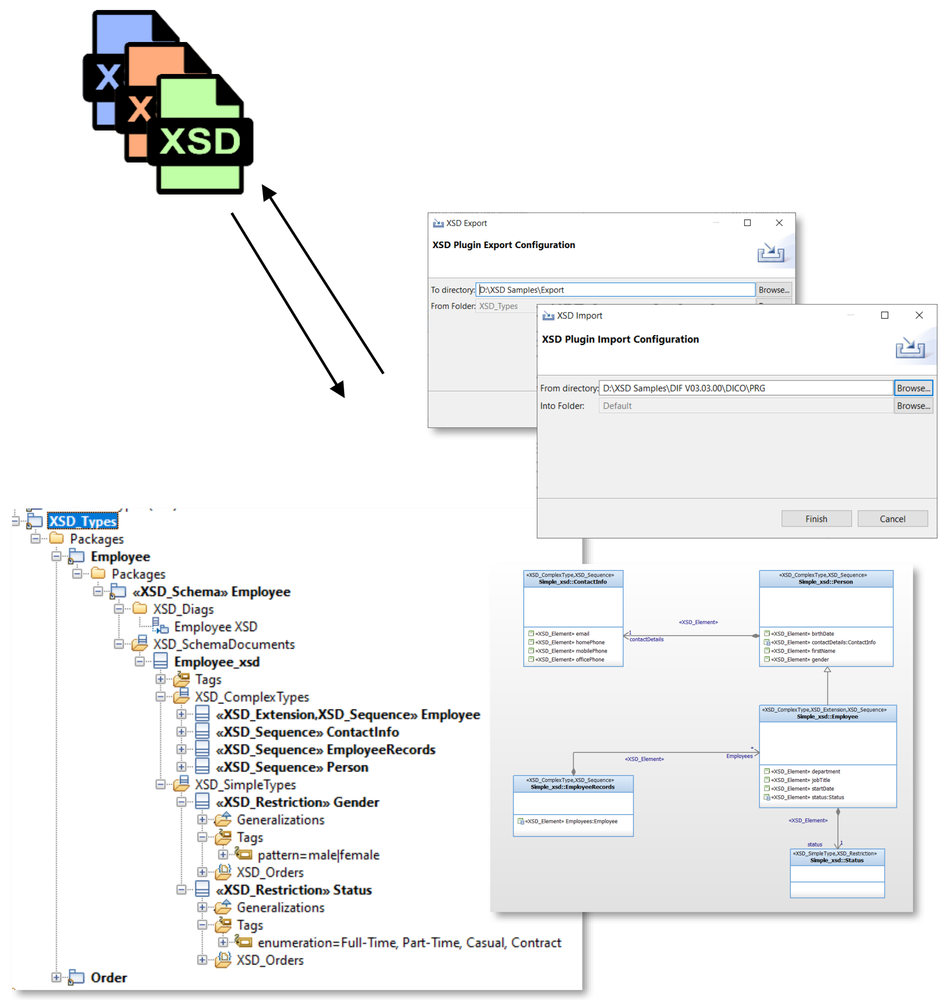
We design and distribute software solutions for Enterprise Interoperability, Data Transformation, and Model-Based Code Generation to improve traceability, exchange, and sharing of engineering data in highly regulated industries.

With offices in France, Germany and the USA, we deploy our solutions worldwide in Aerospace, Automotive, Transportation, Defense and Medical industries.

Buckeburg, Germany
Nantes, France
Detroit, USA

Rhapsody XSD

Import/Export XSD in Rhapsody


XSD
X
X
XSD Export
Browse...
XSD Import
Browse...
Browse...
Finish
Cancel
«XSD_ComplexType,XSD_Sequence» Simple_xsd::ContactInfo
«XSD_ComplexType,XSD_Sequence» Simple_xsd::Person
«XSD_ComplexType,XSD_Extension,XSD_Sequence» Simple_xsd::Employee
«XSD_ComplexType,XSD_Sequence» Simple_xsd::EmployeeRecords
«XSD_SimpleType,XSD_Restriction» Simple_xsd::Status
«XSD_Element»
contactDetails
Employees
status

XSD Plugin Export Configuration

To directory: D:\XSD Samples\Export

From Folder: XSD_Types

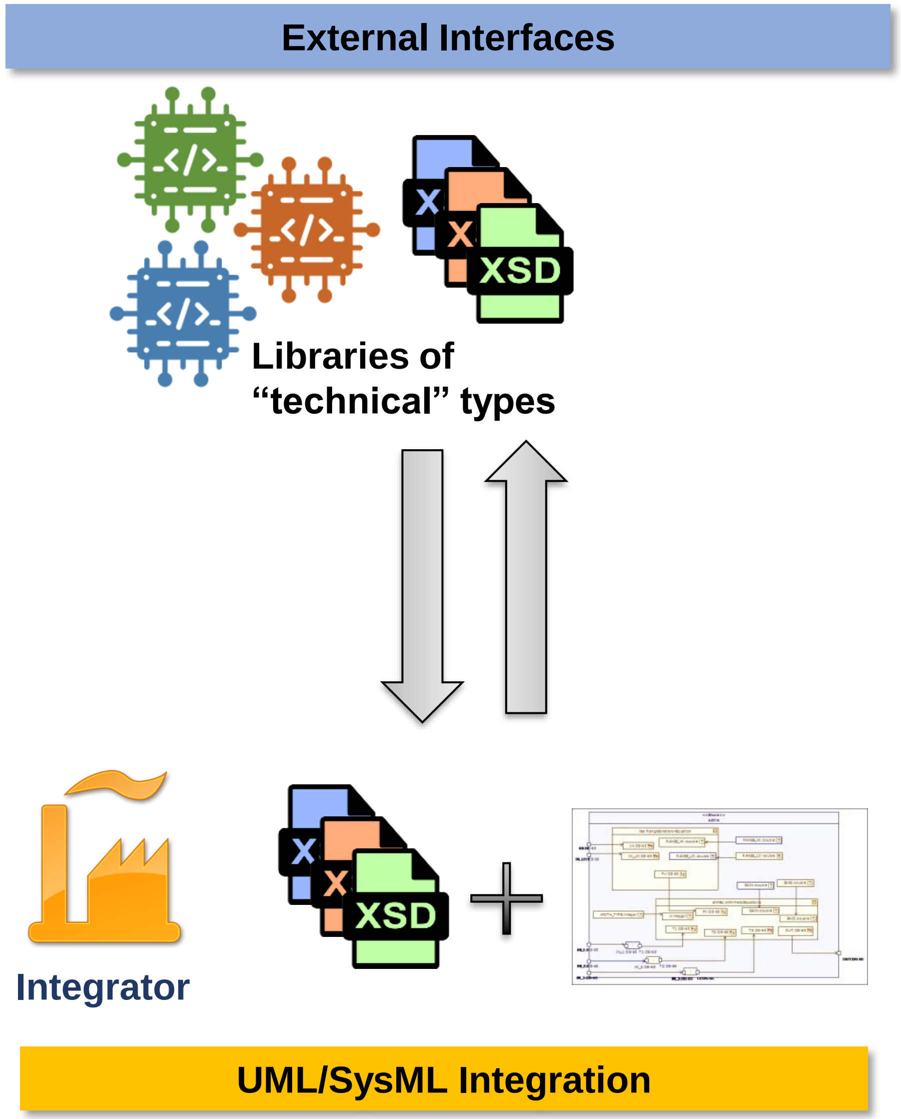
XSD Plugin Import Configuration

From directory: D:\XSD Samples\DIF V03.03.00\DICO\PRG

Into Folder: Default

```
XSD_Types
  Packages
    Employee
      Packages
        «XSD_Schema» Employee
          XSD_Diags
            Employee XSD
          XSD_SchemaDocuments
            Employee_xsd
              Tags
              XSD_ComplexTypes
                «XSD_Extension,XSD_Sequence» Employee
                «XSD_Sequence» ContactInfo
                «XSD_Sequence» EmployeeRecords
                «XSD_Sequence» Person
              XSD_SimpleTypes
                «XSD_Restriction» Gender
                  Generalizations
                  Tags
                    pattern=male|female
                  XSD_Orders
                «XSD_Restriction» Status
                  Generalizations
                  Tags
                    enumeration=Full-Time, Part-Time, Casual, Contract
                  XSD_Orders
    Order
```

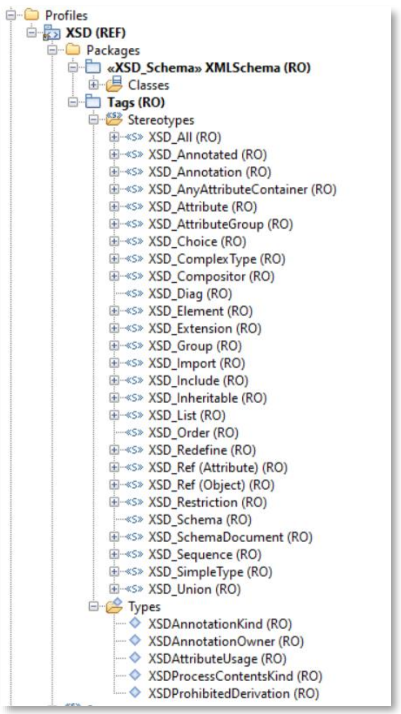
Context



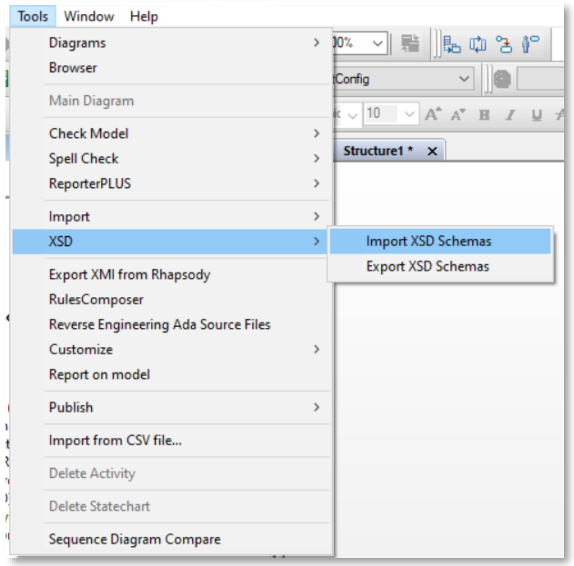
- During design, the data exchanged throughout external interfaces of a system are described by a set of technical XSD files
  - They have to be integrated in the UML/SysML models and types linked with the model.

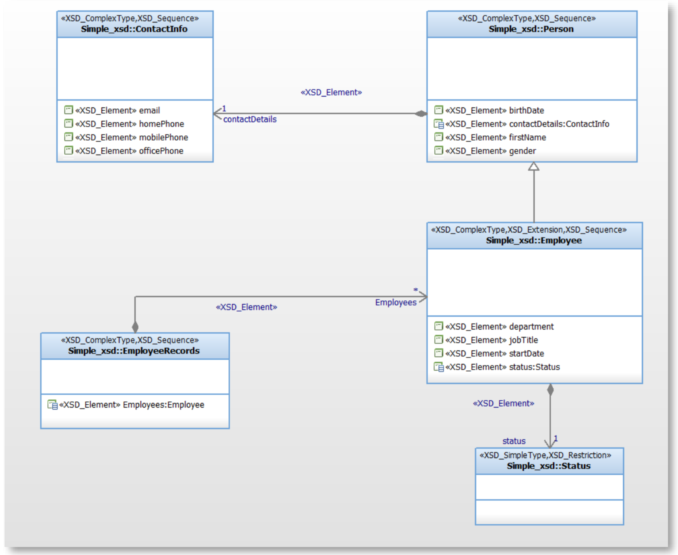
Rhapsody XSD – Key Features

- **Integrate XSD types** in Rhapsody
  - Import XSD files in Rhapsody
- **Make XSD Types understandable** in Rhapsody
  - Simple concepts but enough expressivity
  - Complete XSD Profile and Diagram Support
- **Use** Rhapsody **as an XSD editor**
  - XSD Previewer
  - Export XSD Rhapsody to XSD Files



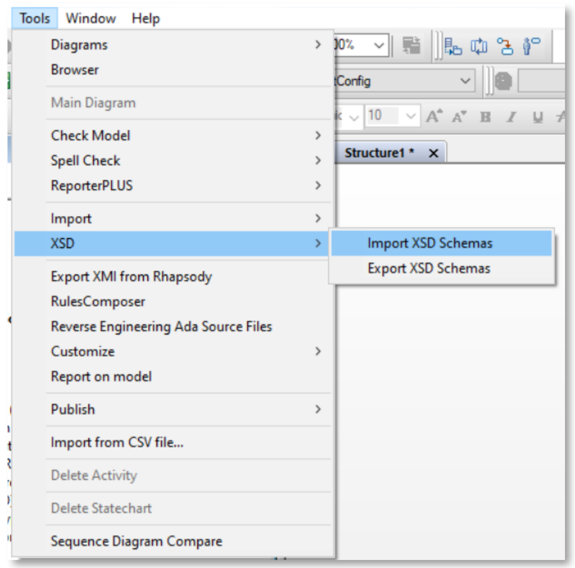
```
Profiles
  XSD (REF)
    Packages
      «XSD_Schema» XMLSchema (RO)
        Classes
      Tags (RO)
        Stereotypes
          XSD_All (RO)
          XSD_Annotated (RO)
          XSD_Annotation (RO)
          XSD_AnyAttributeContainer (RO)
          XSD_Attribute (RO)
          XSD_AttributeGroup (RO)
          XSD_Choice (RO)
          XSD_ComplexType (RO)
          XSD_Compositor (RO)
          XSD_Diag (RO)
          XSD_Element (RO)
          XSD_Extension (RO)
          XSD_Group (RO)
          XSD_Import (RO)
          XSD_Include (RO)
          XSD_Inheritable (RO)
          XSD_List (RO)
          XSD_Order (RO)
          XSD_Redefine (RO)
          XSD_Ref (Attribute) (RO)
          XSD_Ref (Object) (RO)
          XSD_Restriction (RO)
          XSD_Schema (RO)
          XSD_SchemaDocument (RO)
          XSD_Sequence (RO)
          XSD_SimpleType (RO)
          XSD_Union (RO)
        Types
          XSDAnnotationKind (RO)
          XSDAnnotationOwner (RO)
          XSDAttributeUsage (RO)
          XSDProcessContentsKind (RO)
          XSDProhibitedDerivation (RO)
```

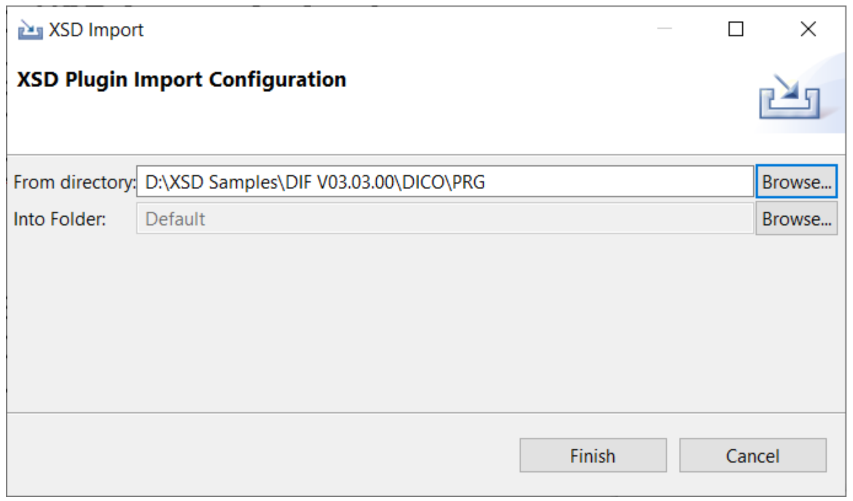



«XSD_ComplexType,XSD_Sequence»
Simple_xsd::ContactInfo
«XSD_Element» email
«XSD_Element» homePhone
«XSD_Element» mobilePhone
«XSD_Element» officePhone
«XSD_Element»
1
contactDetails
«XSD_ComplexType,XSD_Sequence»
Simple_xsd::Person
«XSD_Element» birthDate
«XSD_Element» contactDetails:ContactInfo
«XSD_Element» firstName
«XSD_Element» gender
«XSD_ComplexType,XSD_Extension,XSD_Sequence»
Simple_xsd::Employee
«XSD_Element» department
«XSD_Element» jobTitle
«XSD_Element» startDate
«XSD_Element» status:Status
«XSD_Element»
*
Employees
«XSD_ComplexType,XSD_Sequence»
Simple_xsd::EmployeeRecords
«XSD_Element» Employees:Employee
«XSD_Element»
status
1
«XSD_SimpleType,XSD_Restriction»
Simple_xsd::Status

How to import XSD types in Rhapsody?

- We provide a XSD **Import plugin**
- Can be integrated with standard or customer profiles
- XSD Profile
  - Schema, ComplexTypes…
  - Extending SysML blocks
  - Providing standard library “XMLSchema”
- Allows browsing and creation of Rhapsody Package
- Progress bar
- Textual logs, and visual “tree log”



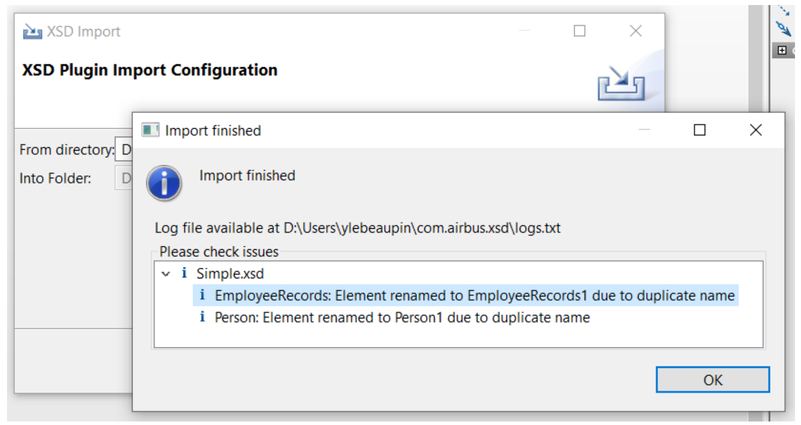


XSD Plugin Import Configuration

From directory: D:\XSD Samples\DIF V03.03.00\DICO\PRG

Into Folder: Default

.



Import finished

Log file available at D:\Users\ylebeaupin\com.airbus.xsd\logs.txt

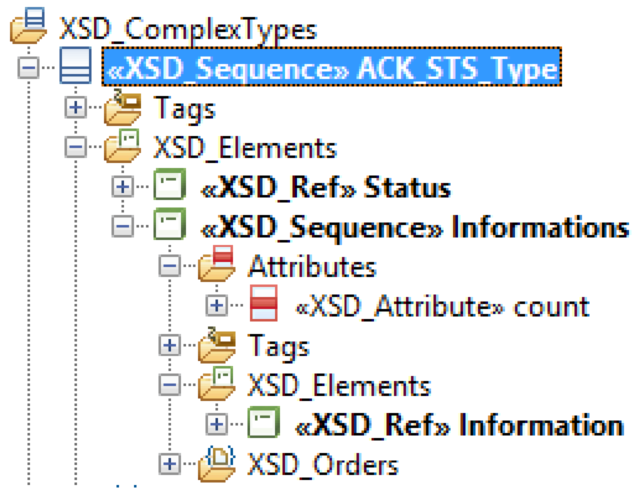
Please check issues

```
Simple.xsd
  EmployeeRecords: Element renamed to EmployeeRecords1 due to duplicate name
  Person: Element renamed to Person1 due to duplicate name
```

How to represent XSD Types in Rhapsody?

- Windows directory structure is reflected through Rhapsody Packages
- Each schema contains its own Package
- XSD structure simplified by tagging objects with stereotypes and usage of implicit parts

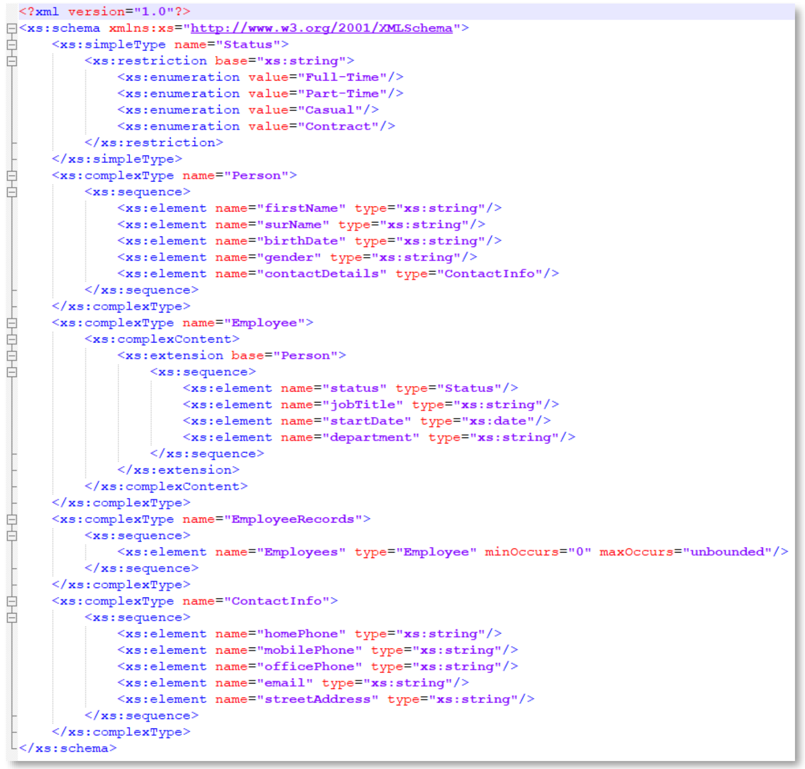
```
<xs:complexType name="ACK_STS_Type">
    <xs:sequence>
      <xs:element ref="edrs_ack_sts_enum:Status"/>
      <xs:element minOccurs="0" name="Informations">
        <xs:complexType>
          <xs:sequence>
            <xs:element maxOccurs="unbounded"
ref="edrs_ack_sts:Information"/>
          </xs:sequence>
          <xs:attribute name="count"
type="xs:positiveInteger" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
```



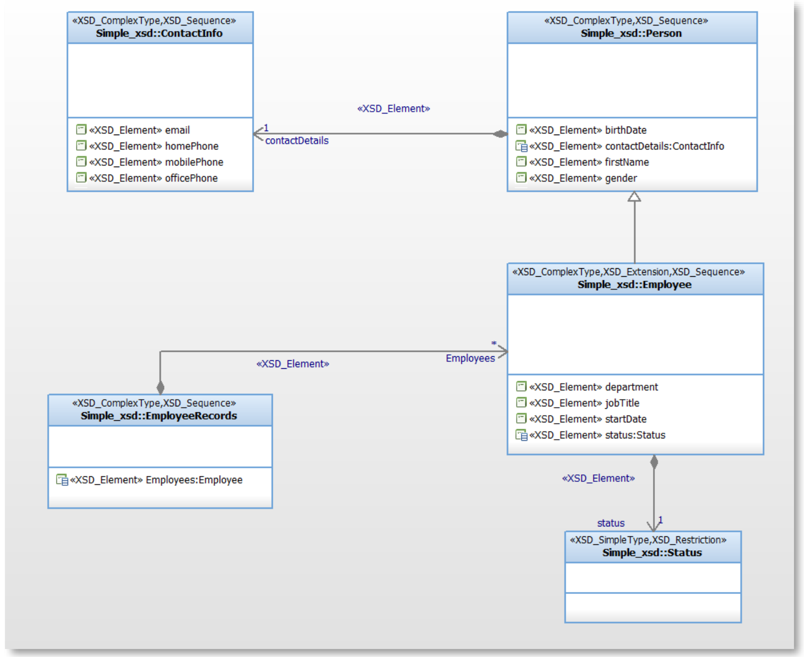
```
XSD_ComplexTypes
  «XSD_Sequence» ACK_STS_Type
    Tags
    XSD_Elements
      «XSD_Ref» Status
      «XSD_Sequence» Informations
        Attributes
          «XSD_Attribute» count
        Tags
        XSD_Elements
          «XSD_Ref» Information
        XSD_Orders
```

Populate of Diagrams

- You can display any XSD element in <<XSD_Diag>> diagrams (displaying internal structure of the selected type or relations between elements)

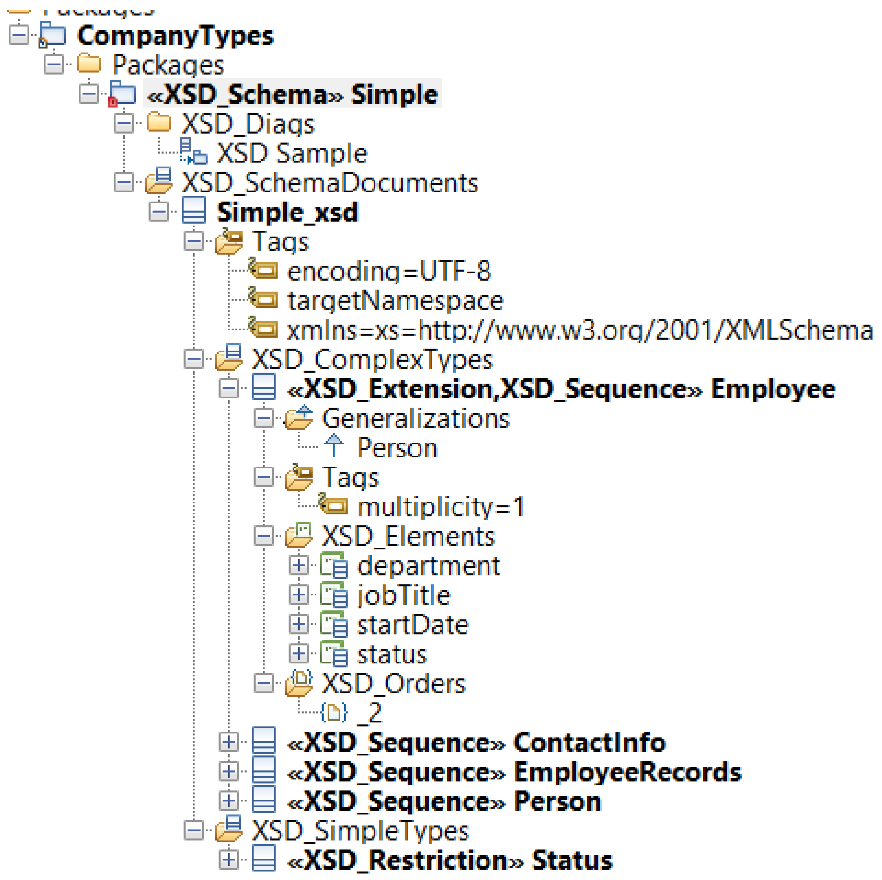


```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:simpleType name="Status">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Full-Time"/>
            <xs:enumeration value="Part-Time"/>
            <xs:enumeration value="Casual"/>
            <xs:enumeration value="Contract"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="Person">
        <xs:sequence>
            <xs:element name="firstName" type="xs:string"/>
            <xs:element name="surName" type="xs:string"/>
            <xs:element name="birthDate" type="xs:string"/>
            <xs:element name="gender" type="xs:string"/>
            <xs:element name="contactDetails" type="ContactInfo"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="Employee">
        <xs:complexContent>
            <xs:extension base="Person">
                <xs:sequence>
                    <xs:element name="status" type="Status"/>
                    <xs:element name="jobTitle" type="xs:string"/>
                    <xs:element name="startDate" type="xs:date"/>
                    <xs:element name="department" type="xs:string"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="EmployeeRecords">
        <xs:sequence>
            <xs:element name="Employees" type="Employee" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ContactInfo">
        <xs:sequence>
            <xs:element name="homePhone" type="xs:string"/>
            <xs:element name="mobilePhone" type="xs:string"/>
            <xs:element name="officePhone" type="xs:string"/>
            <xs:element name="email" type="xs:string"/>
            <xs:element name="streetAddress" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

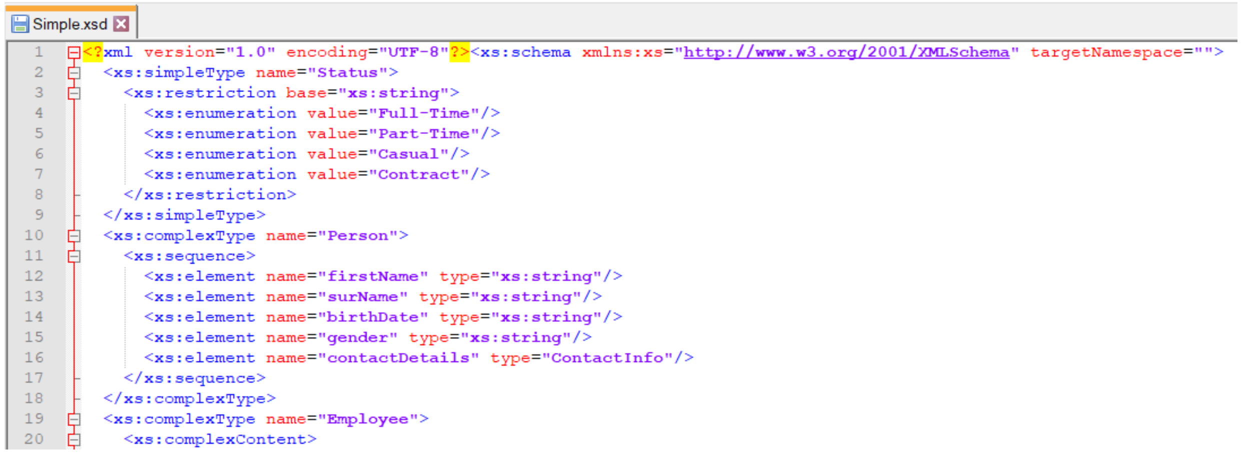

«XSD_ComplexType,XSD_Sequence»
Simple_xsd::ContactInfo
«XSD_Element» email
«XSD_Element» homePhone
«XSD_Element» mobilePhone
«XSD_Element» officePhone
«XSD_Element»
1
contactDetails
«XSD_ComplexType,XSD_Sequence»
Simple_xsd::Person
«XSD_Element» birthDate
«XSD_Element» contactDetails:ContactInfo
«XSD_Element» firstName
«XSD_Element» gender
«XSD_ComplexType,XSD_Extension,XSD_Sequence»
Simple_xsd::Employee
«XSD_Element» department
«XSD_Element» jobTitle
«XSD_Element» startDate
«XSD_Element» status:Status
«XSD_Element»
*
Employees
«XSD_ComplexType,XSD_Sequence»
Simple_xsd::EmployeeRecords
«XSD_Element» Employees:Employee
«XSD_Element»
status
1
«XSD_SimpleType,XSD_Restriction»
Simple_xsd::Status

XSD_Schema

An «XSD_schema» stereotyped Package acts as a container for the **XSD** constructs, from which XML Schema can be generated. All Classes in the Package are defined inside a root <<XSD_SchemaDocument>>.



```
CompanyTypes
  Packages
    «XSD_Schema» Simple
      XSD_Diags
        XSD Sample
      XSD_SchemaDocuments
        Simple_xsd
          Tags
            encoding=UTF-8
            targetNamespace
            xmlns=xs=http://www.w3.org/2001/XMLSchema
          XSD_ComplexTypes
            «XSD_Extension,XSD_Sequence» Employee
              Generalizations
                Person
              Tags
                multiplicity=1
              XSD_Elements
                department
                jobTitle
                startDate
                status
              XSD_Orders
                _2
            «XSD_Sequence» ContactInfo
            «XSD_Sequence» EmployeeRecords
            «XSD_Sequence» Person
          XSD_SimpleTypes
            «XSD_Restriction» Status
```



Simple.xsd

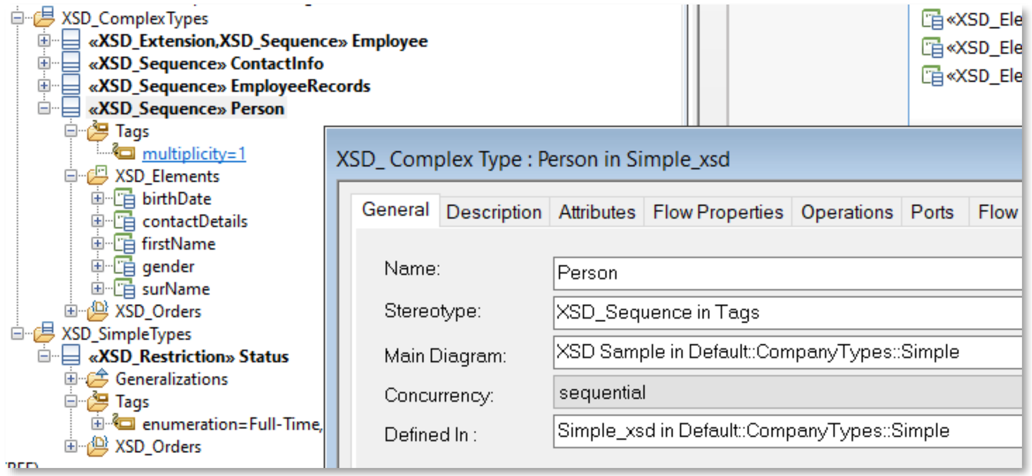
```xml
<?xml version="1.0" encoding="UTF-8"?><xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="">
  <xs:simpleType name="Status">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Full-Time"/>
      <xs:enumeration value="Part-Time"/>
      <xs:enumeration value="Casual"/>
      <xs:enumeration value="Contract"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="Person">
    <xs:sequence>
      <xs:element name="firstName" type="xs:string"/>
      <xs:element name="surName" type="xs:string"/>
      <xs:element name="birthDate" type="xs:string"/>
      <xs:element name="gender" type="xs:string"/>
      <xs:element name="contactDetails" type="ContactInfo"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Employee">
    <xs:complexContent>
```

Note : the root <<XSD_SchemaDocument>> allows direct reference of the whole schema as a type, which is not possible with the «XSD_schema» package

XSD_ComplexType

An «XSDcomplexType» stereotype is applied to a generic **UML Class**, to tailor the generation of a complexType definition in the Schema.

```
<xs:complexType name="Person">
  <xs:sequence>
    <xs:element name="firstName" type="xs:string"/>
    <xs:element name="surName" type="xs:string"/>
    <xs:element name="birthDate" type="xs:string"/>
    <xs:element name="gender" type="xs:string"/>
    <xs:element name="contactDetails" type="ContactInfo"/>
  </xs:sequence>
</xs:complexType>
```

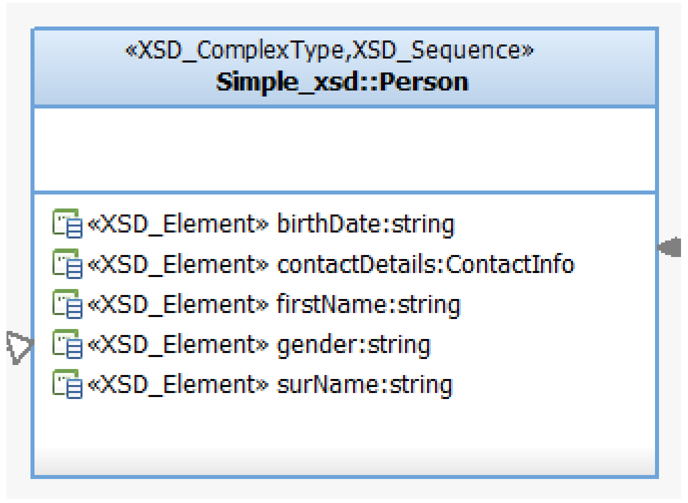


XSD_ComplexTypes
- «XSD_Extension,XSD_Sequence» Employee
- «XSD_Sequence» ContactInfo
- «XSD_Sequence» EmployeeRecords
- «XSD_Sequence» Person
  - Tags
    - multiplicity=1
  - XSD_Elements
    - birthDate
    - contactDetails
    - firstName
    - gender
    - surName
  - XSD_Orders

XSD_SimpleTypes
- «XSD_Restriction» Status
  - Generalizations
  - Tags
    - enumeration=Full-Time,
  - XSD_Orders

XSD_ Complex Type : Person in Simple_xsd

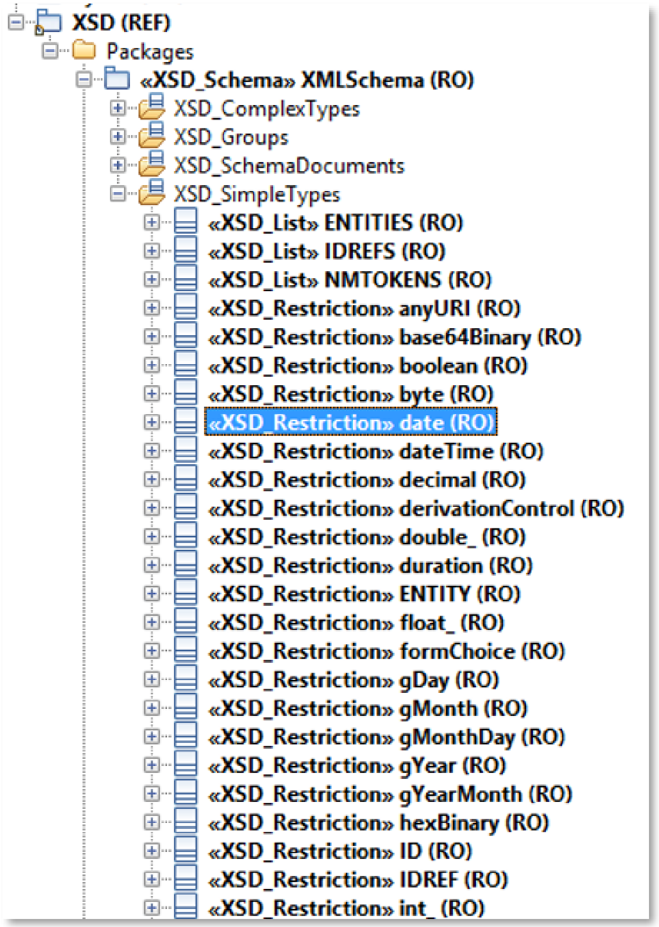
Name:	Person
Stereotype:	XSD_Sequence in Tags
Main Diagram:	XSD Sample in Default::CompanyTypes::Simple
Concurrency:	sequential
Defined In :	Simple_xsd in Default::CompanyTypes::Simple



«XSD_ComplexType,XSD_Sequence»
Simple_xsd::Person
«XSD_Element» birthDate:string
«XSD_Element» contactDetails:ContactInfo
«XSD_Element» firstName:string
«XSD_Element» gender:string
«XSD_Element» surName:string

Native XSD_SimpleType

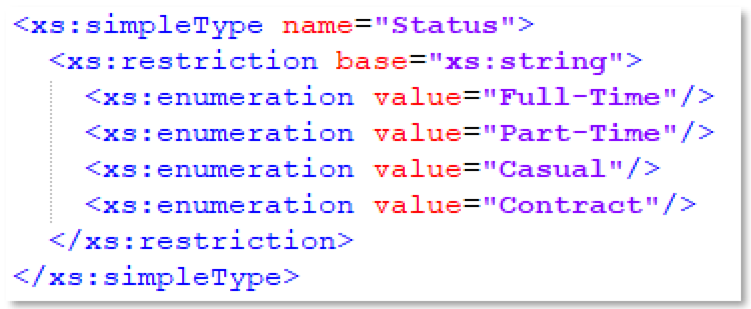
Creating a Schema requires to reference standard types defined in the XML Schema language, as “xsd:decimal”, “xsd:date” etc. The complete XSD Schema definition and its simpletypes is available in the XSD profile.



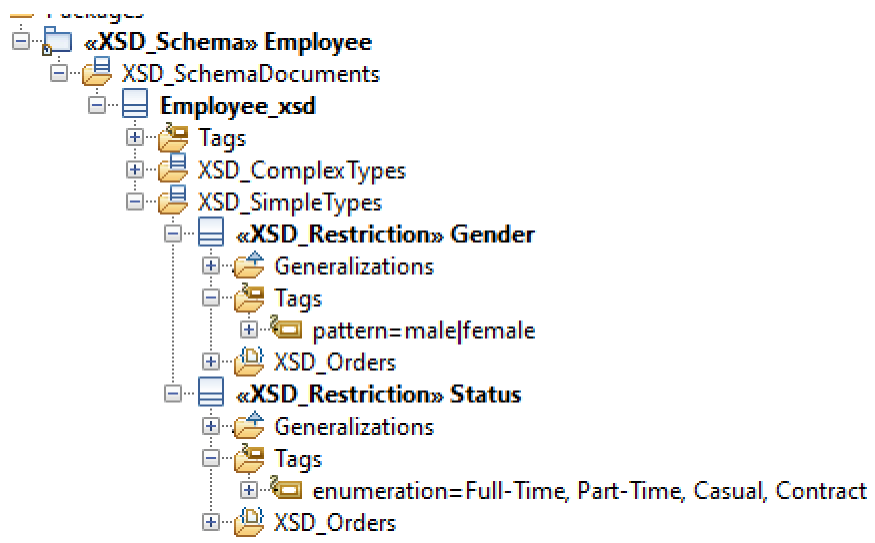
```
XSD (REF)
  Packages
    «XSD_Schema» XMLSchema (RO)
      XSD_ComplexTypes
      XSD_Groups
      XSD_SchemaDocuments
      XSD_SimpleTypes
        «XSD_List» ENTITIES (RO)
        «XSD_List» IDREFS (RO)
        «XSD_List» NMTOKENS (RO)
        «XSD_Restriction» anyURI (RO)
        «XSD_Restriction» base64Binary (RO)
        «XSD_Restriction» boolean (RO)
        «XSD_Restriction» byte (RO)
        «XSD_Restriction» date (RO)
        «XSD_Restriction» dateTime (RO)
        «XSD_Restriction» decimal (RO)
        «XSD_Restriction» derivationControl (RO)
        «XSD_Restriction» double_ (RO)
        «XSD_Restriction» duration (RO)
        «XSD_Restriction» ENTITY (RO)
        «XSD_Restriction» float_ (RO)
        «XSD_Restriction» formChoice (RO)
        «XSD_Restriction» gDay (RO)
        «XSD_Restriction» gMonth (RO)
        «XSD_Restriction» gMonthDay (RO)
        «XSD_Restriction» gYear (RO)
        «XSD_Restriction» gYearMonth (RO)
        «XSD_Restriction» hexBinary (RO)
        «XSD_Restriction» ID (RO)
        «XSD_Restriction» IDREF (RO)
        «XSD_Restriction» int_ (RO)
```

XSD_SimpleType

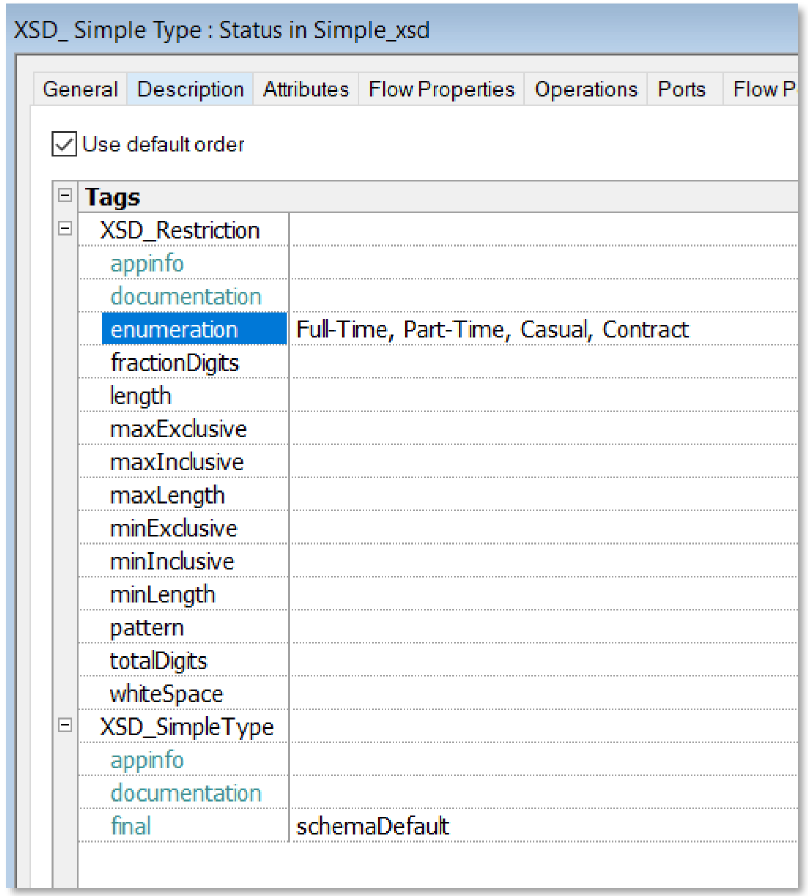
An «XSD_SimpleType» stereotype is applied to a generic **UML Class**, to tailor the generation of a SimpleType definition in the Schema. «XSD_Restriction» and associated tags are used to map enumerations for example.



```
<xs:simpleType name="Status">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Full-Time"/>
    <xs:enumeration value="Part-Time"/>
    <xs:enumeration value="Casual"/>
    <xs:enumeration value="Contract"/>
  </xs:restriction>
</xs:simpleType>
```



```
«XSD_Schema» Employee
  XSD_SchemaDocuments
    Employee_xsd
      Tags
      XSD_ComplexTypes
      XSD_SimpleTypes
        «XSD_Restriction» Gender
          Generalizations
          Tags
            pattern=male|female
          XSD_Orders
        «XSD_Restriction» Status
          Generalizations
          Tags
            enumeration=Full-Time, Part-Time, Casual, Contract
          XSD_Orders
```



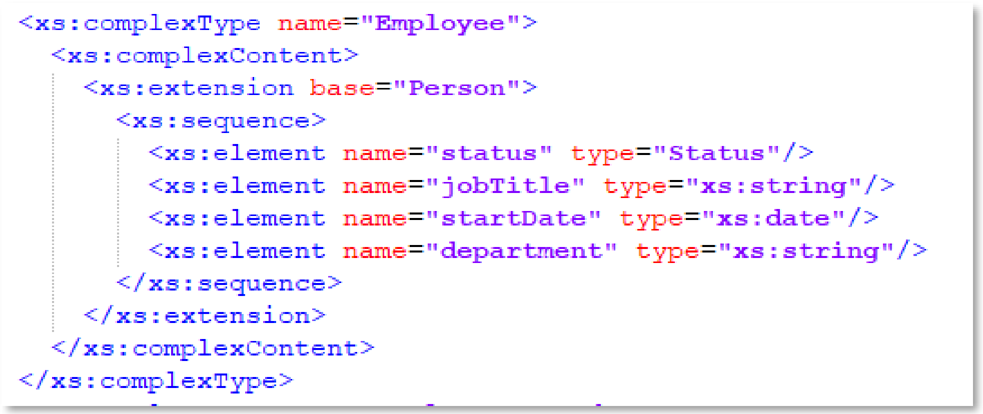
XSD_ Simple Type : Status in Simple_xsd

☑ Use default order

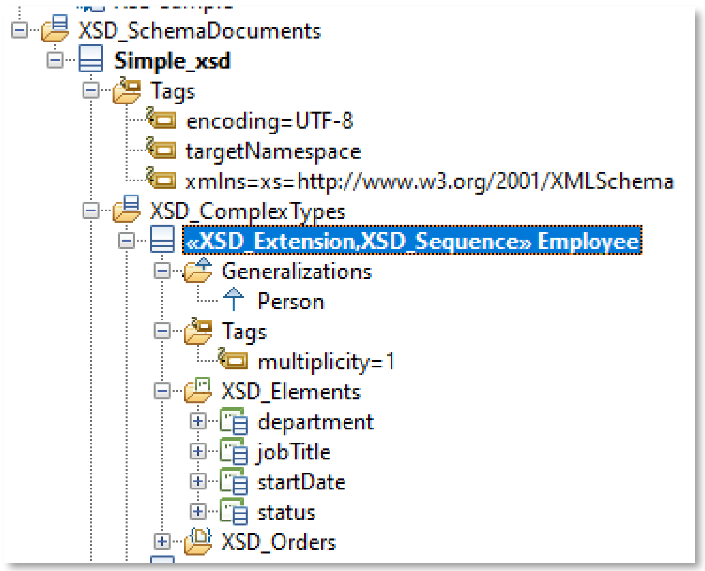
Tags	
XSD_Restriction	
appinfo	
documentation	
enumeration	Full-Time, Part-Time, Casual, Contract
fractionDigits	
length	
maxExclusive	
maxInclusive	
maxLength	
minExclusive	
minInclusive	
minLength	
pattern	
totalDigits	
whiteSpace	
XSD_SimpleType	
appinfo	
documentation	
final	schemaDefault

XSD_Extension

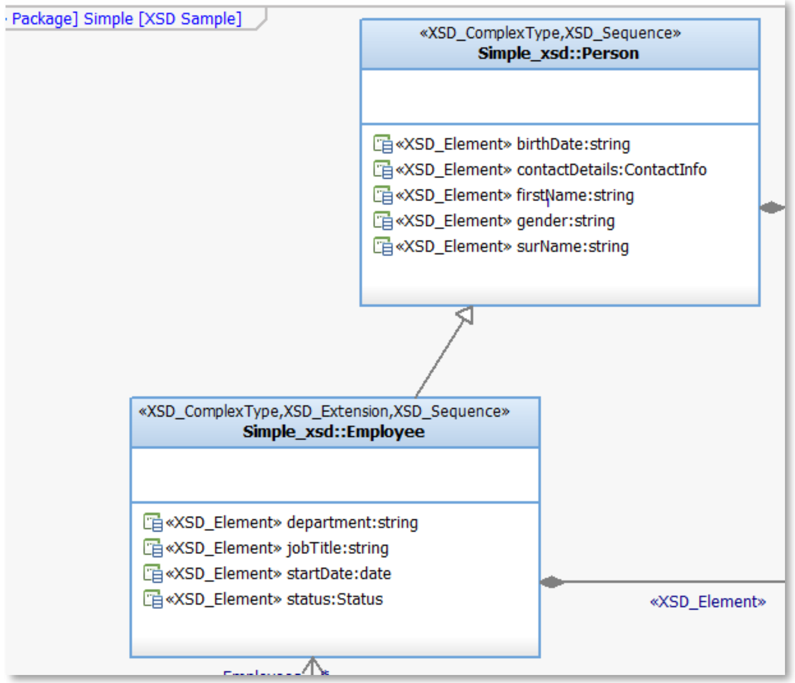
The extension element extends an existing simpleType or complexType element. An «XSD_Extension» stereotype is applied to a generic **UML Class**, to tailor the generation of an Extension definition in the Schema.



```
<xs:complexType name="Employee">
  <xs:complexContent>
    <xs:extension base="Person">
      <xs:sequence>
        <xs:element name="status" type="Status"/>
        <xs:element name="jobTitle" type="xs:string"/>
        <xs:element name="startDate" type="xs:date"/>
        <xs:element name="department" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

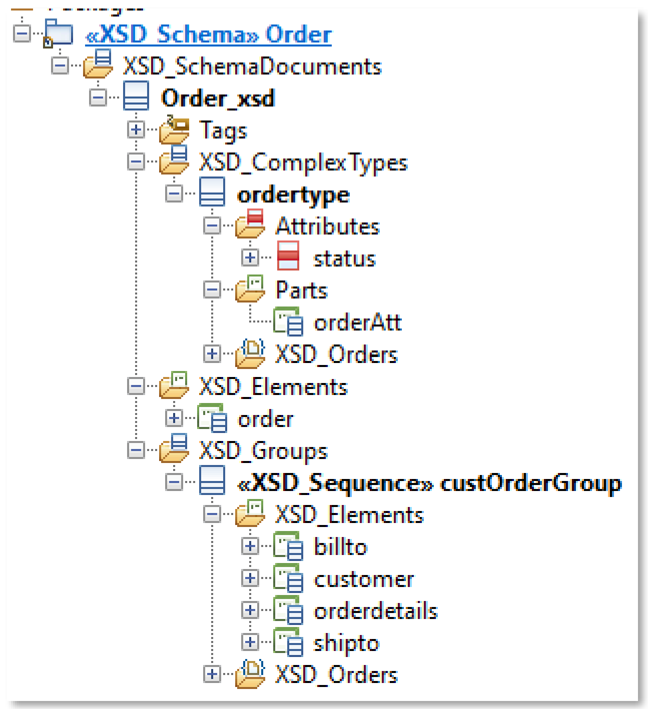


```
XSD_SchemaDocuments
  Simple_xsd
    Tags
      encoding=UTF-8
      targetNamespace
      xmlns=xs=http://www.w3.org/2001/XMLSchema
    XSD_ComplexTypes
      «XSD_Extension,XSD_Sequence» Employee
        Generalizations
          Person
        Tags
          multiplicity=1
        XSD_Elements
          department
          jobTitle
          startDate
          status
      XSD_Orders
```

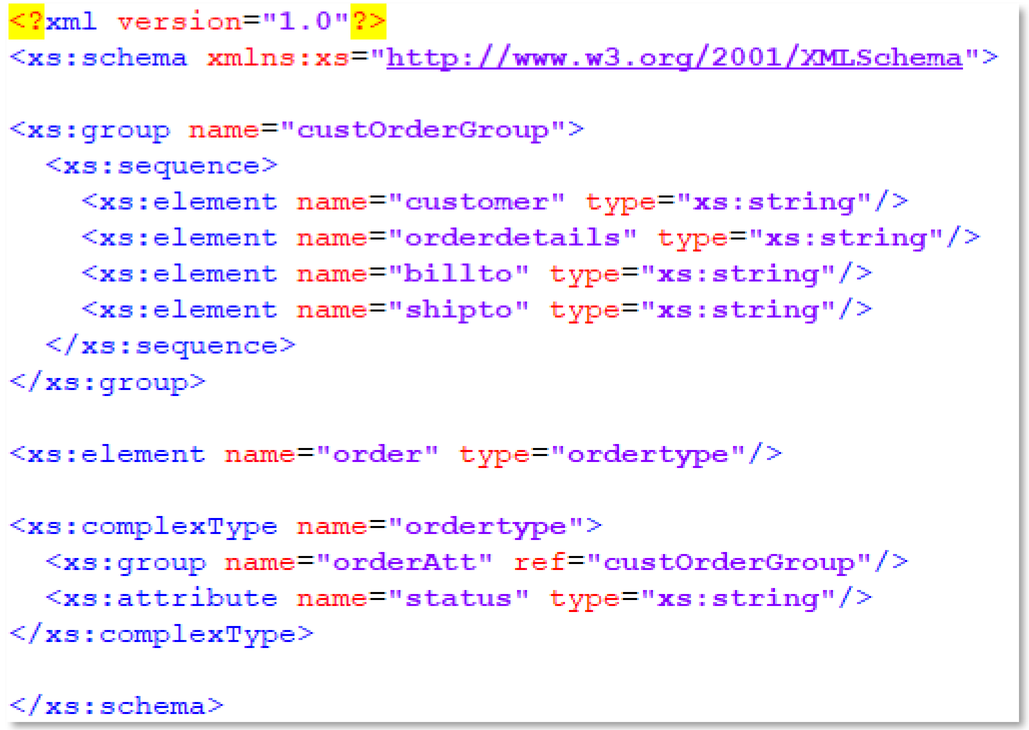

Package] Simple [XSD Sample]
«XSD_ComplexType,XSD_Sequence»
Simple_xsd::Person
«XSD_Element» birthDate:string
«XSD_Element» contactDetails:ContactInfo
«XSD_Element» firstName:string
«XSD_Element» gender:string
«XSD_Element» surName:string
«XSD_ComplexType,XSD_Extension,XSD_Sequence»
Simple_xsd::Employee
«XSD_Element» department:string
«XSD_Element» jobTitle:string
«XSD_Element» startDate:date
«XSD_Element» status:Status
«XSD_Element»

XSD_Group

An «XSD_Group» stereotype is applied to a generic **UML Class**, to tailor the generation of a Group definition in the Schema.



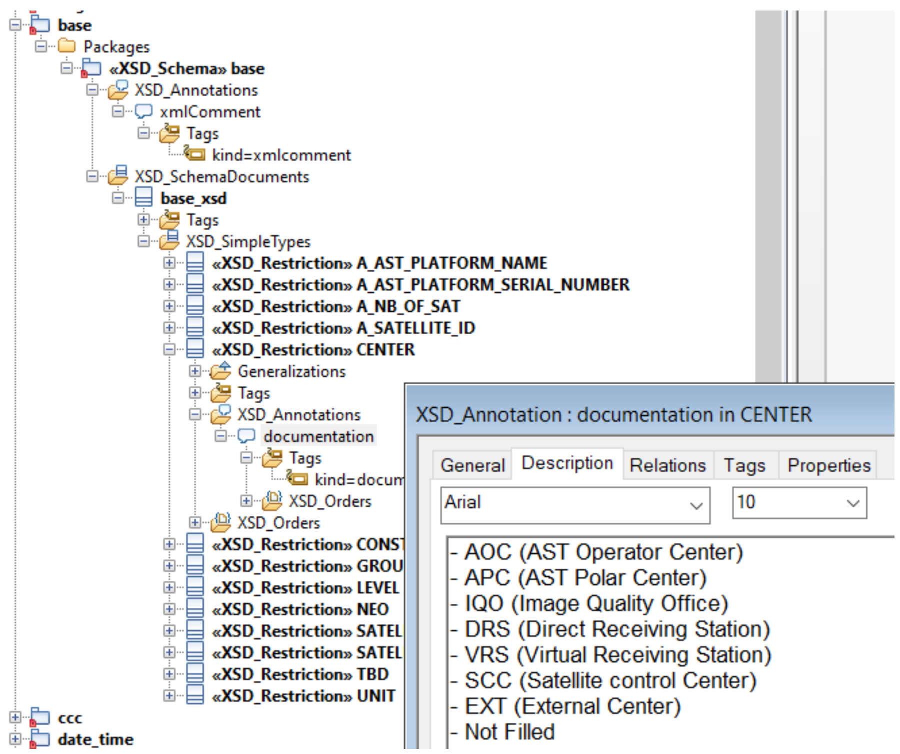
```
«XSD Schema» Order
  XSD_SchemaDocuments
    Order_xsd
      Tags
      XSD_ComplexTypes
        ordertype
          Attributes
            status
          Parts
            orderAtt
          XSD_Orders
      XSD_Elements
        order
      XSD_Groups
        «XSD_Sequence» custOrderGroup
          XSD_Elements
            billto
            customer
            orderdetails
            shipto
          XSD_Orders
```



```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:group name="custOrderGroup">
  <xs:sequence>
    <xs:element name="customer" type="xs:string"/>
    <xs:element name="orderdetails" type="xs:string"/>
    <xs:element name="billto" type="xs:string"/>
    <xs:element name="shipto" type="xs:string"/>
  </xs:sequence>
</xs:group>

<xs:element name="order" type="ordertype"/>

<xs:complexType name="ordertype">
  <xs:group name="orderAtt" ref="custOrderGroup"/>
  <xs:attribute name="status" type="xs:string"/>
</xs:complexType>

</xs:schema>
```

XSD_Annotation

The annotation element is a top level element that specifies schema comments. The comments serve as inline documentation. Rhapsody description is used to store the XSD documentation or appInfo.



```
base
  Packages
    «XSD_Schema» base
      XSD_Annotations
        xmlComment
          Tags
            kind=xmlcomment
      XSD_SchemaDocuments
        base_xsd
          Tags
          XSD_SimpleTypes
            «XSD_Restriction» A_AST_PLATFORM_NAME
            «XSD_Restriction» A_AST_PLATFORM_SERIAL_NUMBER
            «XSD_Restriction» A_NB_OF_SAT
            «XSD_Restriction» A_SATELLITE_ID
            «XSD_Restriction» CENTER
              Generalizations
              Tags
              XSD_Annotations
                documentation
                  Tags
                    kind=docum
                  XSD_Orders
              XSD_Orders
            «XSD_Restriction» CONST
            «XSD_Restriction» GROU
            «XSD_Restriction» LEVEL
            «XSD_Restriction» NEO
            «XSD_Restriction» SATEL
            «XSD_Restriction» SATEL
            «XSD_Restriction» TBD
            «XSD_Restriction» UNIT
ccc
date_time
```

XSD_Annotation : documentation in CENTER

General | Description | Relations | Tags | Properties

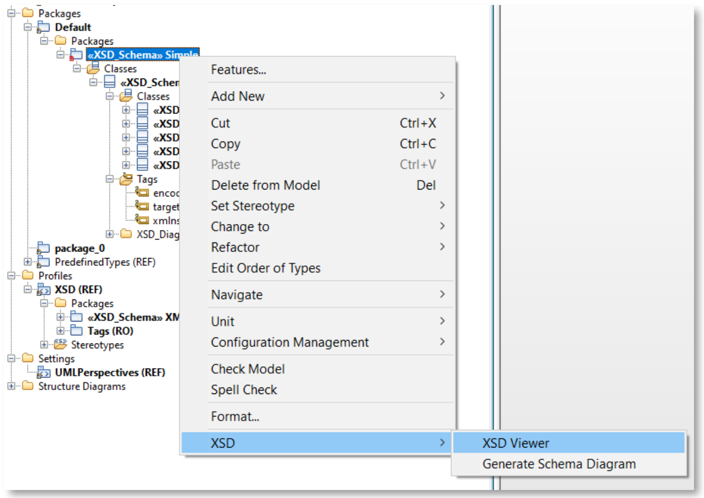
Arial | 10

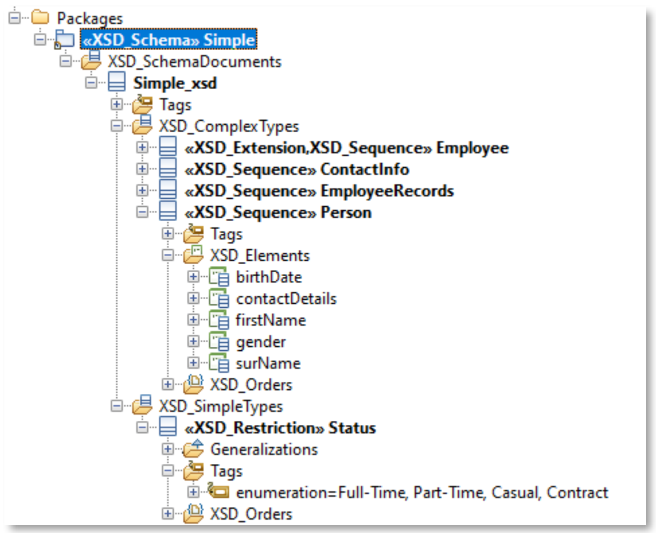
- AOC (AST Operator Center)
- APC (AST Polar Center)
- IQO (Image Quality Office)
- DRS (Direct Receiving Station)
- VRS (Virtual Receiving Station)
- SCC (Satellite control Center)
- EXT (External Center)
- Not Filled

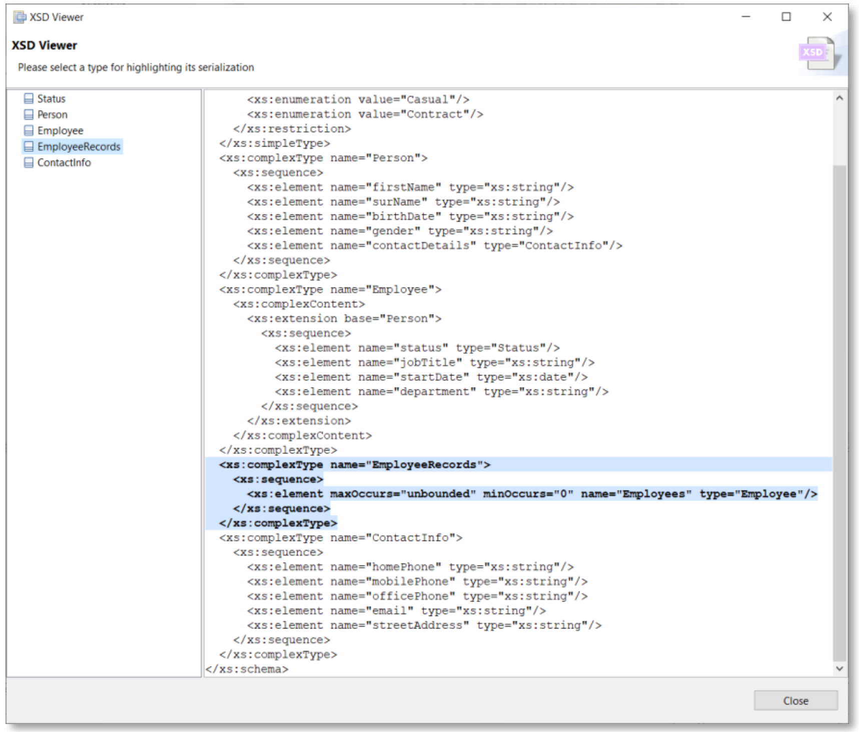
```
<xs:simpleType name="CENTER">
    <xs:annotation>
        <xs:documentation>- AOC (AST Operator Center)
- APC (AST Polar Center)
- IQO (Image Quality Office)
- DRS (Direct Receiving Station)
- VRS (Virtual Receiving Station)
- SCC (Satellite control Center)
- EXT (External Center)
- Not Filled</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
        <xs:enumeration value="AOC"/>
        <xs:enumeration value="APC"/>
        <xs:enumeration value="IQO"/>
        <xs:enumeration value="DRS"/>
        <xs:enumeration value="VRS"/>
        <xs:enumeration value="SCC"/>
        <xs:enumeration value="EXT"/>
        <xs:enumeration value="Not Filled"/>
    </xs:restriction>
</xs:simpleType>
```

How to use Rhapsody as an XSD editor?

- New elements available in the “add new” menu
- XSD Diagrams have their tool menu extended for having new types
- We provide an export plugin, based on the same GUI than import
- We provide an XSD Viewer plugin
  - Bidirectional highlighting
  - Highlight types in Rhapsody browser



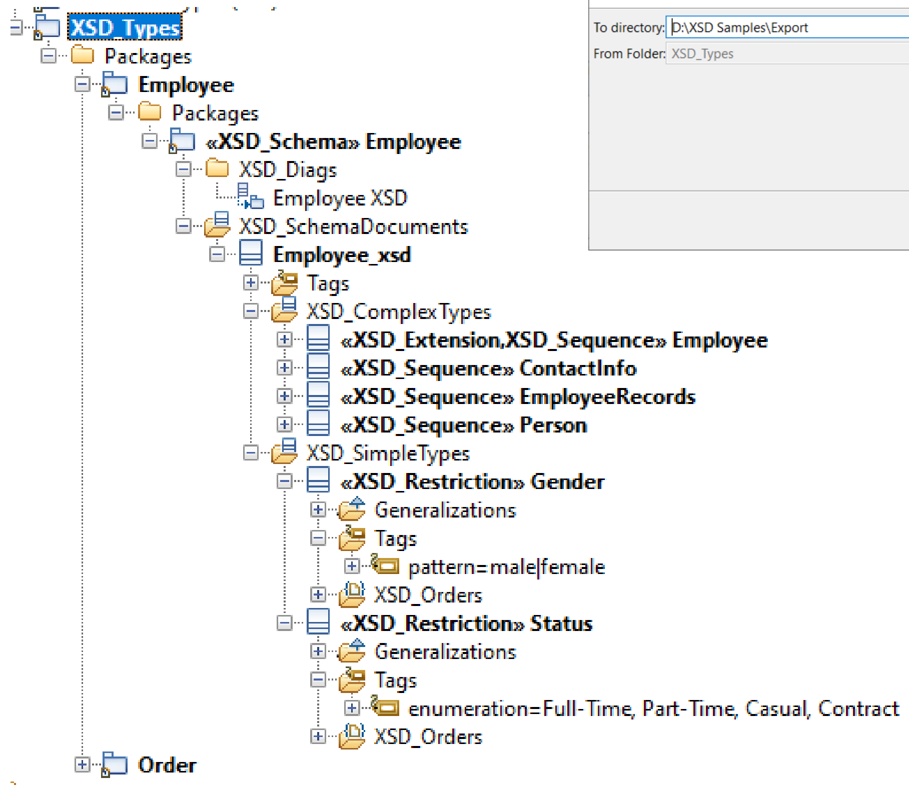




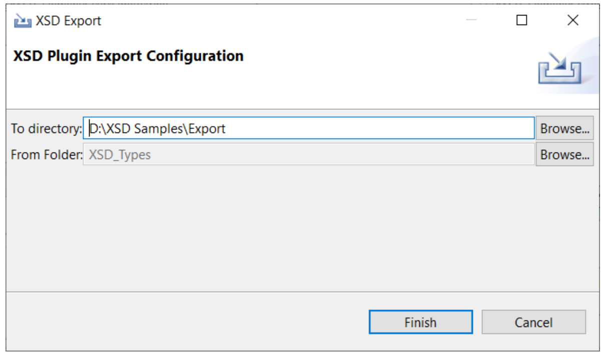
```
      <xs:enumeration value="Casual"/>
      <xs:enumeration value="Contract"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="Person">
    <xs:sequence>
      <xs:element name="firstName" type="xs:string"/>
      <xs:element name="surName" type="xs:string"/>
      <xs:element name="birthDate" type="xs:string"/>
      <xs:element name="gender" type="xs:string"/>
      <xs:element name="contactDetails" type="ContactInfo"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Employee">
    <xs:complexContent>
      <xs:extension base="Person">
        <xs:sequence>
          <xs:element name="status" type="Status"/>
          <xs:element name="jobTitle" type="xs:string"/>
          <xs:element name="startDate" type="xs:date"/>
          <xs:element name="department" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="EmployeeRecords">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="Employees" type="Employee"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ContactInfo">
    <xs:sequence>
      <xs:element name="homePhone" type="xs:string"/>
      <xs:element name="mobilePhone" type="xs:string"/>
      <xs:element name="officePhone" type="xs:string"/>
      <xs:element name="email" type="xs:string"/>
      <xs:element name="streetAddress" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

How to export XSD from Rhapsody?

- We provide a **XSD Export plugin**
- Rhapsody Packages are reflected in Windows directory structure



```
XSD_Types
  Packages
    Employee
      Packages
        «XSD_Schema» Employee
          XSD_Diags
            Employee XSD
          XSD_SchemaDocuments
            Employee_xsd
              Tags
              XSD_ComplexTypes
                «XSD_Extension,XSD_Sequence» Employee
                «XSD_Sequence» ContactInfo
                «XSD_Sequence» EmployeeRecords
                «XSD_Sequence» Person
              XSD_SimpleTypes
                «XSD_Restriction» Gender
                  Generalizations
                  Tags
                    pattern=male|female
                  XSD_Orders
                «XSD_Restriction» Status
                  Generalizations
                  Tags
                    enumeration=Full-Time, Part-Time, Casual, Contract
                  XSD_Orders
    Order
```

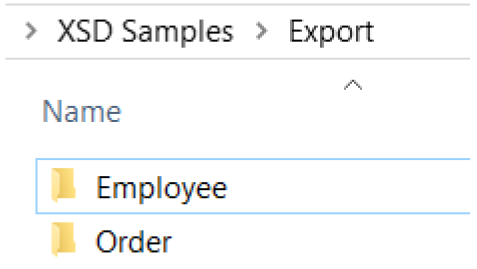


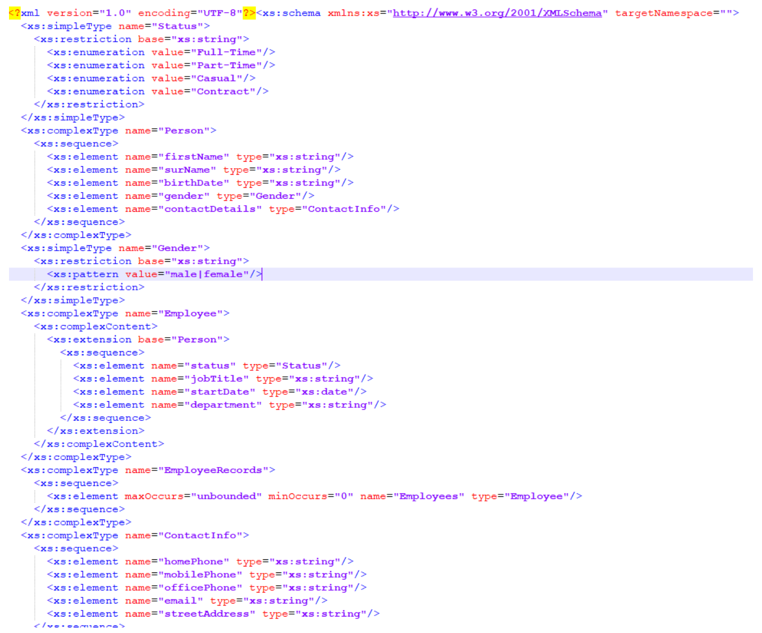
XSD Export

XSD Plugin Export Configuration

To directory: D:\XSD Samples\Export

From Folder: XSD_Types





```
<?xml version="1.0" encoding="UTF-8"?><xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="">
  <xs:simpleType name="Status">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Full-Time"/>
      <xs:enumeration value="Part-Time"/>
      <xs:enumeration value="Casual"/>
      <xs:enumeration value="Contract"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="Person">
    <xs:sequence>
      <xs:element name="firstName" type="xs:string"/>
      <xs:element name="surName" type="xs:string"/>
      <xs:element name="birthDate" type="xs:string"/>
      <xs:element name="gender" type="Gender"/>
      <xs:element name="contactDetails" type="ContactInfo"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="Gender">
    <xs:restriction base="xs:string">
      <xs:pattern value="male|female"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="Employee">
    <xs:complexContent>
      <xs:extension base="Person">
        <xs:sequence>
          <xs:element name="status" type="Status"/>
          <xs:element name="jobTitle" type="xs:string"/>
          <xs:element name="startDate" type="xs:date"/>
          <xs:element name="department" type="xs:string"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="EmployeeRecords">
    <xs:sequence>
      <xs:element maxOccurs="unbounded" minOccurs="0" name="Employees" type="Employee"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ContactInfo">
    <xs:sequence>
      <xs:element name="homePhone" type="xs:string"/>
      <xs:element name="mobilePhone" type="xs:string"/>
      <xs:element name="officePhone" type="xs:string"/>
      <xs:element name="email" type="xs:string"/>
      <xs:element name="streetAddress" type="xs:string"/>
    </xs:sequence>
```

SODIUS WILLERT

The Power of Connected Data

SODIUS SAS
34 Boulevard du Maréchal A. Juin
44100 Nantes
+33 (0)228 236 060

SODIUS CORP
418 N. Main Street 2nd Floor
Royal Oak, MI 48067
+1 (248) 270-2950

WILLERT SOFTWARE TOOLS GmbH
Hannoversche Str. 21,
31675 Bückeburg
+49 5722 9678 60

For more information visit sodiuswillert.com