Tutorial

Copyright © 1998-2012. Altova GmbH. All rights reserved. Use of this software is governed by and subject to an Altova software license agreement. XMLSpy, MapForce, StyleVision, SemanticWorks, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, AltovaXML, MissionKit, and ALTOVA as well as their logos are trademarks and/or registered trademarks of Altova GmbH. Protected by US Patent 7,739,292.

XML, XSL, XHTML, and W3C are trademarks (registered in numerous countries) of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions, MIT, INRIA, and Keio. UNICODE and the Unicode Logo are trademarks of Unicode Inc. This software contains 3rd party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at http://www.altova.com/legal_3rdparty.html

Altova XMLSpy 2013 Tutorial

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Published: 2012

© 2012 Altova GmbH

Table of Contents

1 XMLSpy Interface 2

2 XML Schemas: Basics 3

2.1 Creating a New XML Schema File ... 4
2.2 Defining Namespaces ... 7
2.3 Defining a Content Model ... 8
2.4 Adding Elements with Drag-and-Drop ... 12
2.5 Configuring the Content Model View ... 13
2.6 Completing the Basic Schema ... 15

3 XML Schemas: Advanced 18

3.1 Working with Complex Types and Simple Types ... 19
3.2 Referencing Global Elements ... 27
3.3 Attributes and Attribute Enumerations ... 29

4 XML Schemas: XMLSpy Features 32

4.1 Schema Navigation ... 33
4.2 Schema Documentation ... 35

5 XML Documents 39

5.1 Creating a New XML File ... 40
5.2 Specifying the Type of an Element ... 42
5.3 Entering Data in Grid View ... 44
5.4 Entering Data in Text View ... 45
5.5 Validating the Document ... 49
5.6 Adding Elements and Attributes ... 53
5.7 Editing in Database/Table View ... 55
5.8 Modifying the Schema ... 59

6 XSLT Transformations 61

6.1 Assigning an XSLT File ... 62
6.2 Transforming the XML File ... 63
6.3 Modifying the XSL File ... 64

7 Project Management 66

7.1 Benefits of Projects ... 67
7.2 Building a Project ... 68

8 That's It **70**

Index **71**

XMLSpy Tutorial

This tutorial provides an overview of XML and takes you through a number of key XML tasks. In the process you will learn how to use some of XMLSpy's most powerful features.

The tutorial is divided into the following parts:

- Creating an XML Schema. You will learn how to create an XML Schema in XMLSpy's intuitive Schema View, how to create complex content models using drag-and-drop mechanisms, and how to configure Schema View.
- Using Schema View features to create complex and simple types, global element references, and attribute enumerations.
- Learning how to navigate schemas in Schema View, and how to generate documentation of schemas.
- Creating an XML document. You will learn how to assign a schema for an XML document, edit an XML document in Grid View and Text View, and validate XML documents using XMLSpy's built-in validator.
- Transforming an XML file using an XSLT stylesheet. This involves assigning an XSLT file and carrying out the transformation using XMLSpy's built-in XSLT engines.
- Working with XMLSpy projects, which enable you to easily organize your XML documents.

Installation and configuration

This tutorial assumes that you have successfully installed XMLSpy on your computer and received a free evaluation key-code, or are a registered user. The evaluation version of XMLSpy is fully functional but limited to a 30-day period. You can request a regular license from our secure web server or through any one of our resellers.

Tutorial example files

The tutorial files are available in the application folder:

```
C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2013\
Examples\Tutorial
```

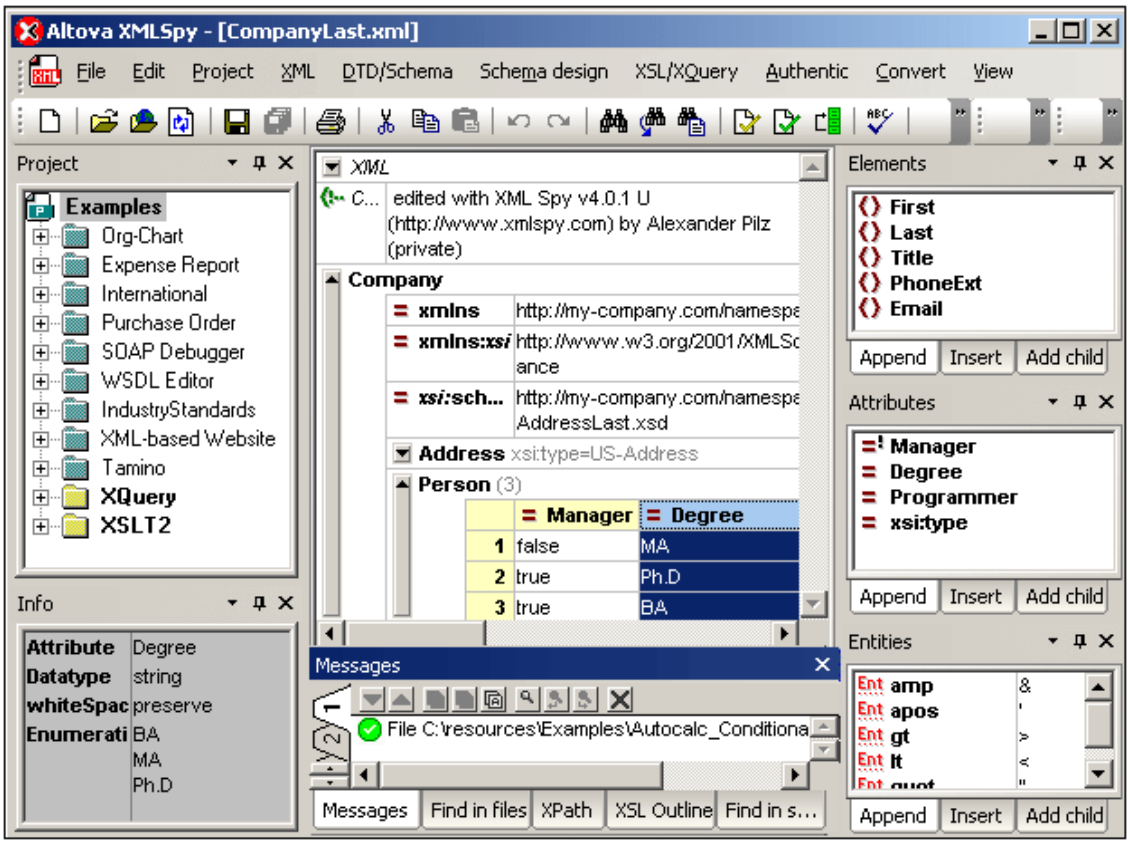
The `Examples` folder contains various XML files for you to experiment with, while the `Tutorial` folder contains all the files used in this tutorial.

The `Template` folder in the application folder (typically in `c:\Program Files\Altova`) contains all the XML template files that are used whenever you select the menu option **File | New**. These files supply the necessary data (namespaces and XML declarations) for you to start working with the respective XML document immediately.

1 XMLSpy Interface

The XMLSpy interface is structured into three vertical areas. The central area provides you with multiple views of your XML document. The areas on either side of this central area contain windows that provide information, editing help, and file management features.

- The left area consists of the **Project** and **Info** windows.
- The central area, called the **Main** window, is where you edit and view all types of XML documents. You can switch between different views: Text View, Grid View, Schema View, WSDL View, Authentic View, and Browser View. These views are described in detail in the individual sections about them in the User Manual.
- The right-hand area contains the three **Entry Helper** windows, which enable you to insert or append elements, attributes, and entities. What entries are displayed in the Entry Helper windows depends on the current selection or cursor location in the XML file.



The details of the interface are explained as we go along. Note that the interface changes dynamically according to the document that is active in the Main Window and according to the view selected.

 © 2012 Altova GmbH

2 XML Schemas: Basics

An XML Schema describes the structure of an XML document. An XML document can be validated against an XML Schema to check whether it conforms to the requirements specified in the schema. If it does, it is said to be **valid**; otherwise it is **invalid**. XML Schemas enable document designers to specify the allowed structure and content of an XML document and to check whether an XML document is valid.

The structure and syntax of an XML Schema document is complex, and being an XML document itself, an XML Schema must be valid according to the rules of the XML Schema specification. In XMLSpy, Schema View enables you to easily build valid XML Schemas by using graphical drag-and-drop techniques. The XML Schema document you construct is also editable in Text View and Grid View, but is much easier to create and modify in Schema View.

Objective
In this section of the tutorial, you will learn how to edit XML Schemas in Schema View. Specifically, you will learn how to do the following:

- Create a new schema file
- Define namespaces for the schema
- Define a basic content model
- Add elements to the content model using context menus and drag-and-drop
- Configure the Content Model View

After you have completed creating the basic schema, you can go to the next section of the tutorial, which teaches you how to work with the more advanced features of XML Schema in XMLSpy. This advanced section is followed by a section about schema navigation and documentation in XMLSpy.

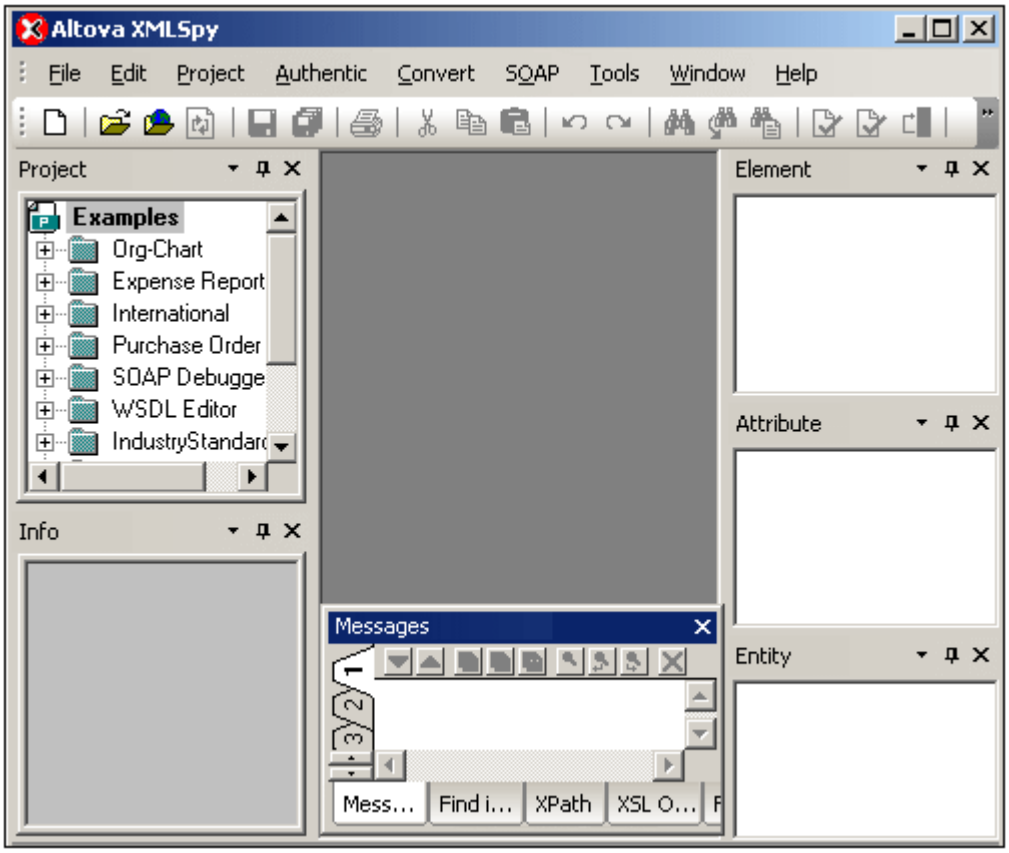
Commands used in this section
In this section of the tutorial, you will use Schema View exclusively. The following commands are used:

Display Diagram (or Display Content Model View). This icon is located to the left of all global components in Schema Overview. Click this icon to display the content model of the associated global component.

2.1 Creating a New XML Schema File

To create a new XML Schema file in XMLSpy, you must first start XMLSpy and then create a new XML Schema (`.xsd`) document.

Starting XMLSpy
To start XMLSpy, double-click the XMLSpy icon on your desktop or use the **Start | All Programs** menu to access the XMLSpy program. XMLSpy is started with no documents open in the interface.

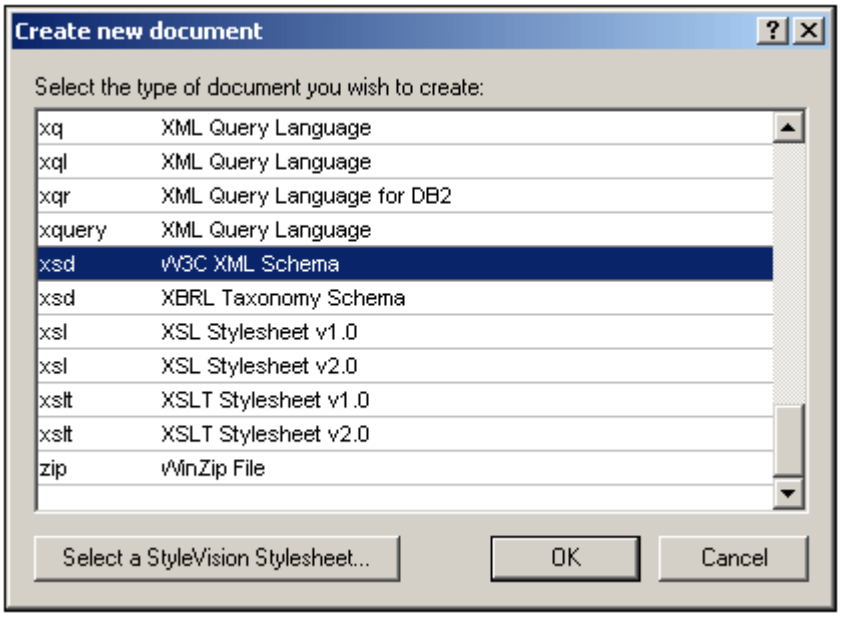


Note the four main parts of the interface: (i) the Project and Info Windows on the left; (ii) the Main Window in the middle; (iii) the Entry Helpers on the right; and (iv) the Output Windows at the bottom.

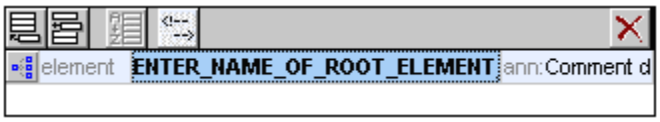
 © 2012 Altova GmbH

Creating a new XML Schema file
To create a new XML Schema file:

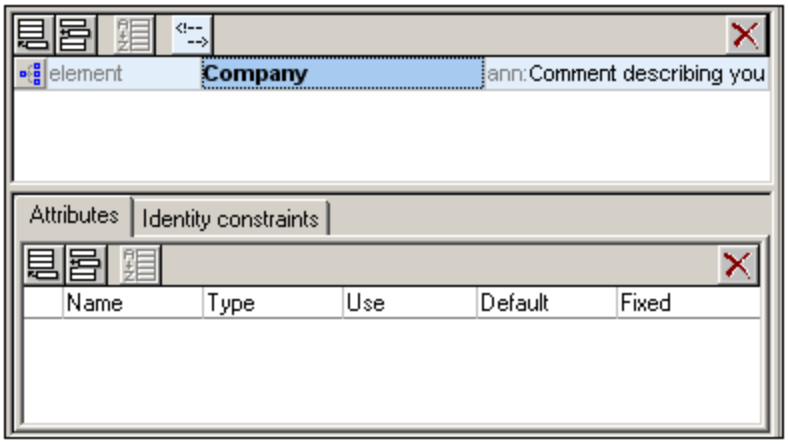
1. Select the menu option **File | New**. The Create new document dialog opens.



2. In the dialog, select the `xsd` entry (the document description and the list in the window might vary from that in the screenshot) and confirm with **OK**. An empty schema file appears in the Main Window in Schema View. You are prompted to enter the name of the root element.



3. Double-click in the highlighted field and enter `Company`. Confirm with **Enter**. `Company` is now the root element of this schema and is created as a global element. The view you see in the Main Window *(screenshot below)* is called the Schema Overview. It provides an overview of the schema by displaying a list of all the global components in the top pane of the Main Window; the bottom pane displays the attributes and identity constraints of the selected global component. (You can view and edit the content model of individual global components by clicking the Display Diagram icon to the left of that global component.)



4. In the Annotations field (`ann`) of the `Company` element, enter the description of the element, in this case, `Root element`.
5. Click the menu option **File | Save**, and save your XML Schema with any name you like (`AddressFirst.xsd`, for example).

© 2012 Altova GmbH

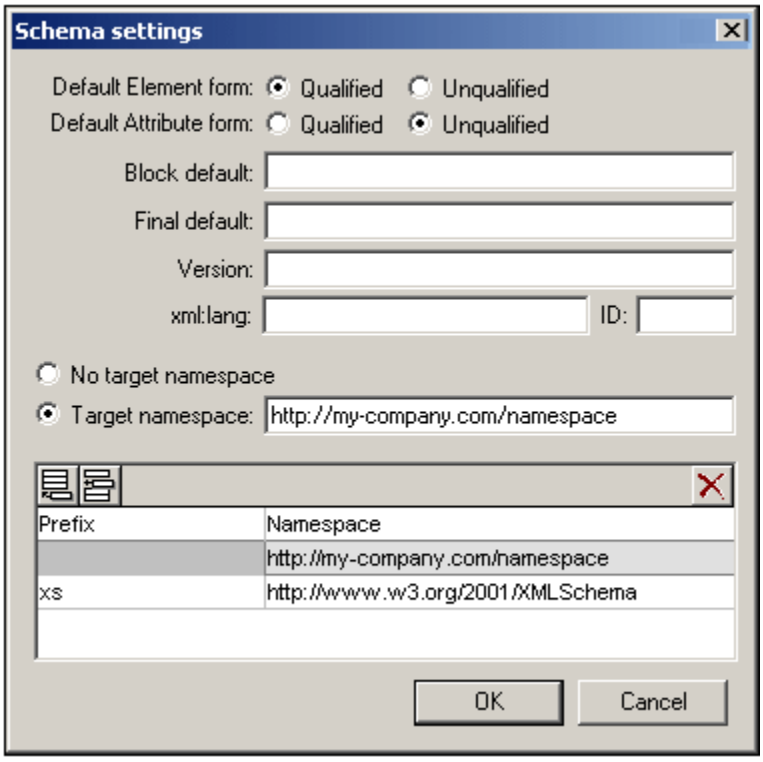
2.2 Defining Namespaces

XML namespaces are an important issue in XML Schemas and XML documents. An XML Schema document must reference the XML Schema namespace and, optionally, it can define a target namespace for the XML document instance. As the schema designer, you must decide how to define both these namespaces (essentially, with what prefixes.)

In the XML Schema you are creating, you will define a target namespace for XML document instances. (The required reference to the XML Schema namespace is created automatically by XMLSpy when you create a new XML Schema document.)

To create a target namespace:

1. Select the menu option **Schema Design | Schema settings**. This opens the Schema Settings dialog.



2. Click the Target Namespace radio button, and enter `http://my-company.com/namespace`. In XMLSpy, the namespace you enter as the target namespace is created as the default namespace of the XML Schema document and displayed in the list of namespaces in the bottom pane of the dialog.
3. Confirm with the **OK** button.

Please note:

- The XML Schema namespace is automatically created by XMLSpy and given a prefix of `xs:`.
- When the XML document instance is created, it must have the target namespace defined in the XML Schema for the XML document to be valid.

2.3 Defining a Content Model

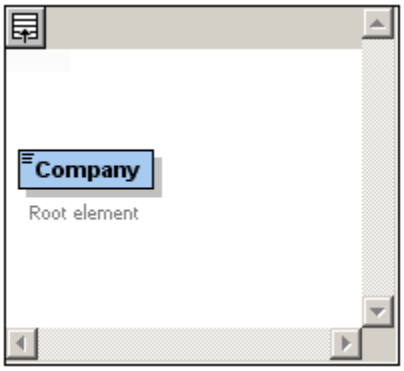
In the Schema Overview, you have already created a global element called `Company`. This element is to contain one `Address` element and an unlimited number of `Person` elements—its content model. Global components that can have content models are elements, complexTypes, and element groups.

In XMLSpy, the content model of a global component is displayed in the Content Model View of Schema View (*screenshot below*). To view and edit the content model of a global component, click the Display Diagram icon located to the left of the global component.

In this section, you will create the content model of the `Company` element.

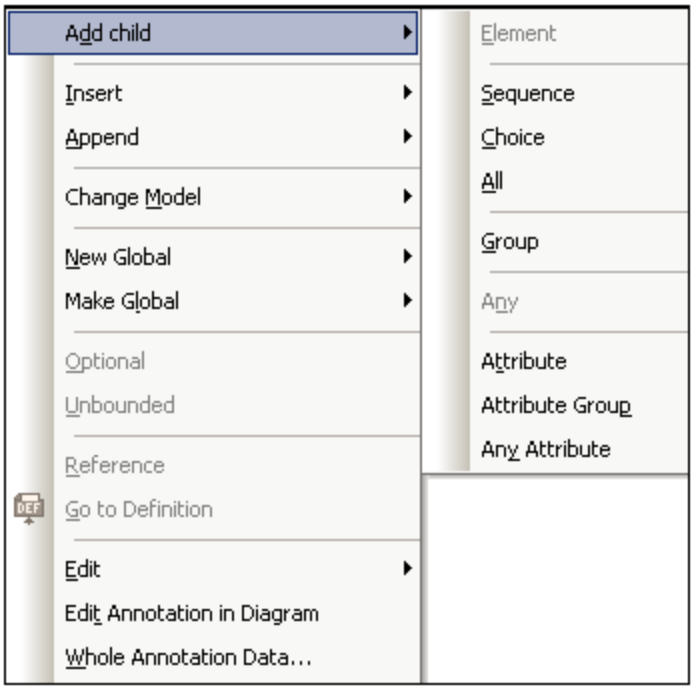
Creating a basic content model
To create the content model of the `Company` element:

1. In the Schema Overview, click the Display Diagram icon of the `Company` element. This displays the content model of the `Company` element (*screenshot below*), which is currently empty. Alternatively, you can double-click the `Company` entry in the Components entry helper to display its content model.

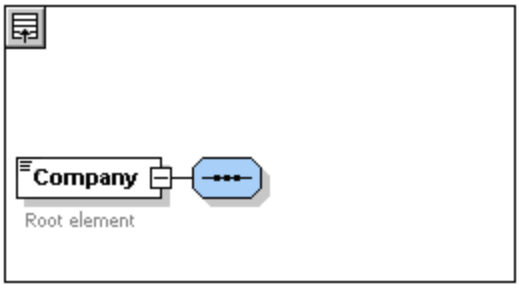


2. A content model consists of **compositors** and **components**. The compositors specify the relationship between two components. At this point of the `Company` content model, you must add a child compositor to the `Company` element in order to add a child element. To add a compositor, right-click the `Company` element. From the context menu that appears, select **Add Child | Sequence**. (Sequence, Choice, and All are the three compositors that can be used in a content model.)

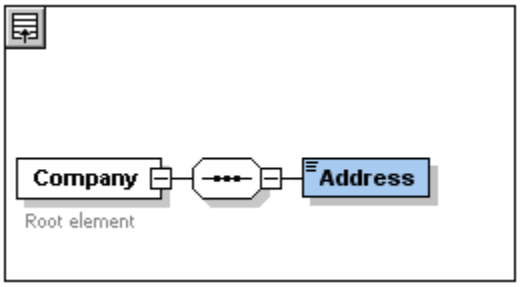
 © 2012 Altova GmbH



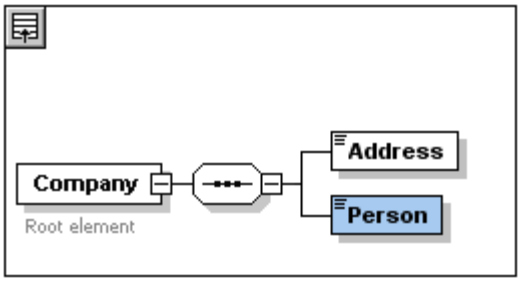
This inserts the Sequence compositor, which defines that the components that follow must appear in the specified sequence.



3. Right-click the Sequence compositor and select **Add Child | Element**. An unnamed element component is added.
4. Enter `Address` as the name of the element, and confirm with **Enter**.

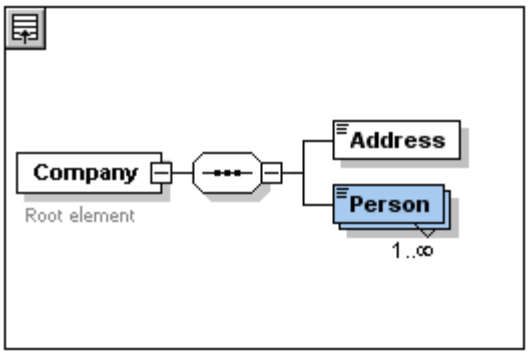


5. Right-click the Sequence compositor again, select **Add Child | Element**. Name the newly created element component `Person`.



You have so far defined a schema which allows for one address and one person per company. We need to increase the number of `Person` elements.

6. Right-click the `Person` element, and select **Unbounded** from the context menu. The `Person` element in the diagram now shows the number of allowed occurrences: 1 to infinity.



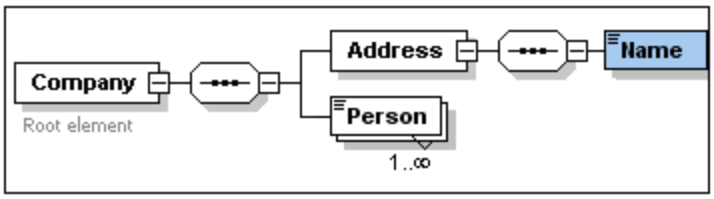
Alternatively, in the Details Entry Helper, you can edit the `minOcc` and `maxOcc` fields to specify the allowed number of occurrences, in this case 1 and unbounded, respectively.

Adding additional levels to the content model structure

The basic content model you have created so far contains one level: a child level for the `company` element which contains the `Address` and `Person` elements. Now we will define the content of the `Address` element so it contains `Name`, `Street`, and `City` elements. This is a second level. Again we need to add a child compositor to the `Address` element, and then the element components themselves.

Do this as follows:

1. Right-click the `Address` element to open the context menu, and select **Add Child | Sequence**. This adds the Sequence compositor.
2. Right-click the Sequence compositor, and select **Add Child | Element**. Name the newly created element component `Name`.



Complex types, simple types, and XML Schema data types

Till this point, we have not explicitly defined any element type. Click the **Text** tab to display the

 © 2012 Altova GmbH

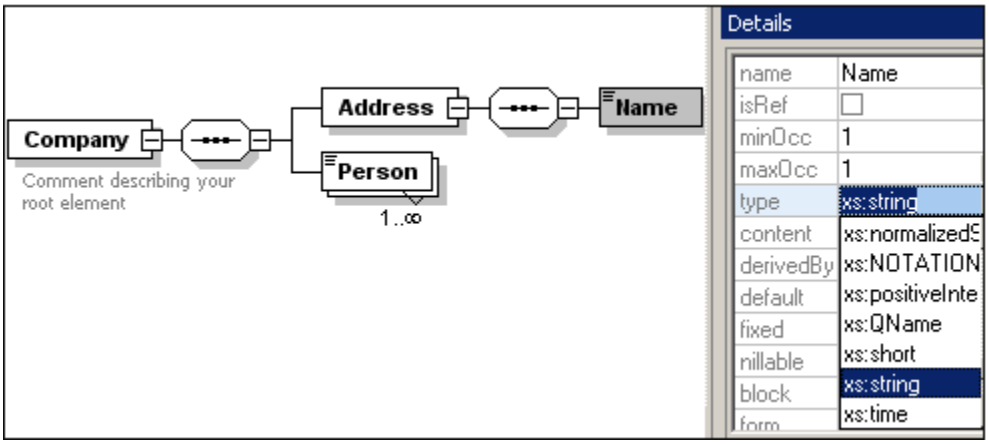
Text View of your schema (*listing below*). You will notice that whenever a Sequence compositor was inserted, the `xs:sequence` element was inserted within the `xs:complexType` element. In short, the `Company` and `Address` elements, because they contain child elements, are complex types. A complex type element is one which contains attributes or elements.

```
<xs:element name="Company">
  <xs:annotation>
    <xs:documentation>Root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Address">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Name"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="Person"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Simple type elements, on the other hand, contain only text and have no attributes. Text can be strings, dates, numbers, etc. We want the `Name` child of `Address` to contain only text. It is a simple type, the text content of which we want to restrict to a string. We can do this using the XML Schema data type `xs:string`.

To define the `Name` element to be of this datatype:

1. Click the **Schema** tab to return to Schema View.
2. Click the `Name` element to select it.
3. In the Details Entry Helper, from the dropdown menu of the `type` combo box, select the `xs:string` entry.



Note that both `minOcc` and `maxOcc` have a value of 1, showing that this element occurs only once.

The text representation of the `Name` element is as follows:

```
<xs:element name="Name" type="xs:string"/>
```

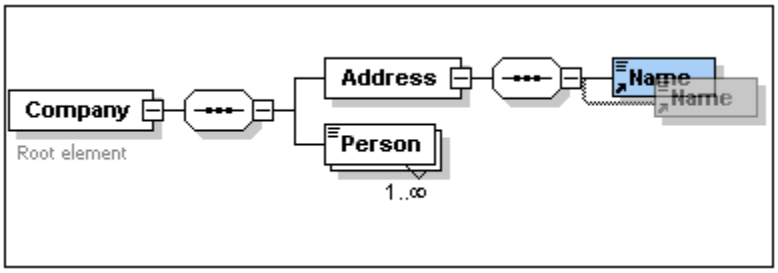
Please note: A simple type element can have any one of several XML Schema data types. In all these cases, the icon indicating text-content appears in the element box.

2.4 Adding Elements with Drag-and-Drop

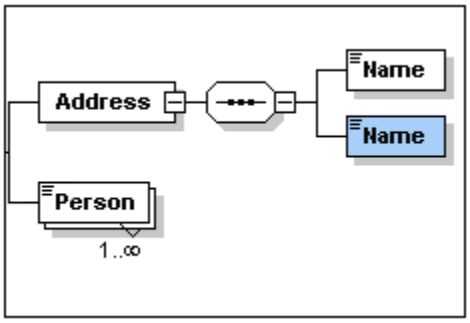
You have added elements using the context menu that appears when you right-click an element or compositor. You can also create elements using drag-and-drop, which is quicker than using menu commands. In this section, you will add more elements to the definition of the `Address` element using drag-and-drop, thus completing this definition.

To complete the definition of the `Address` element using drag-and-drop:

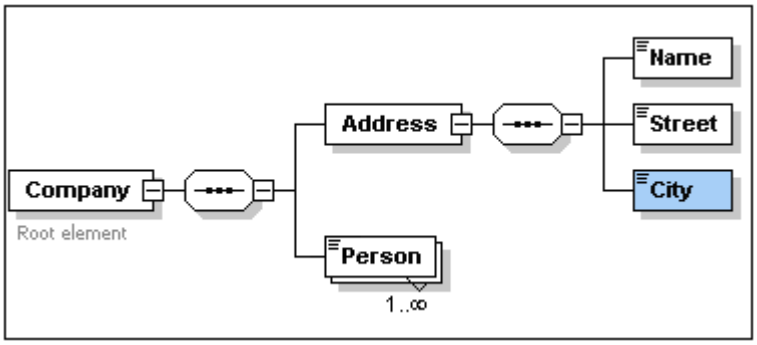
1. Click the `Name` element of the `Address` element, hold down the **Ctrl** key, and drag the element box with the mouse. A small "plus" icon appears in the element box, indicating that you are about to copy the element. A copy of the element together with a connector line also appears, showing where the element will be created.



2. Release the mouse button to create the new element in the `Address` sequence. If the new element appears at an incorrect location, drag it to a location below the `Name` element.



3. Double-click in the element box, and type in `Street` to change the element name.
4. Use the same method to create a third element called `City`. The content model should now look like this:



The `Address` element has a sequence of a `Name`, a `Street`, and a `City` element, in that order.

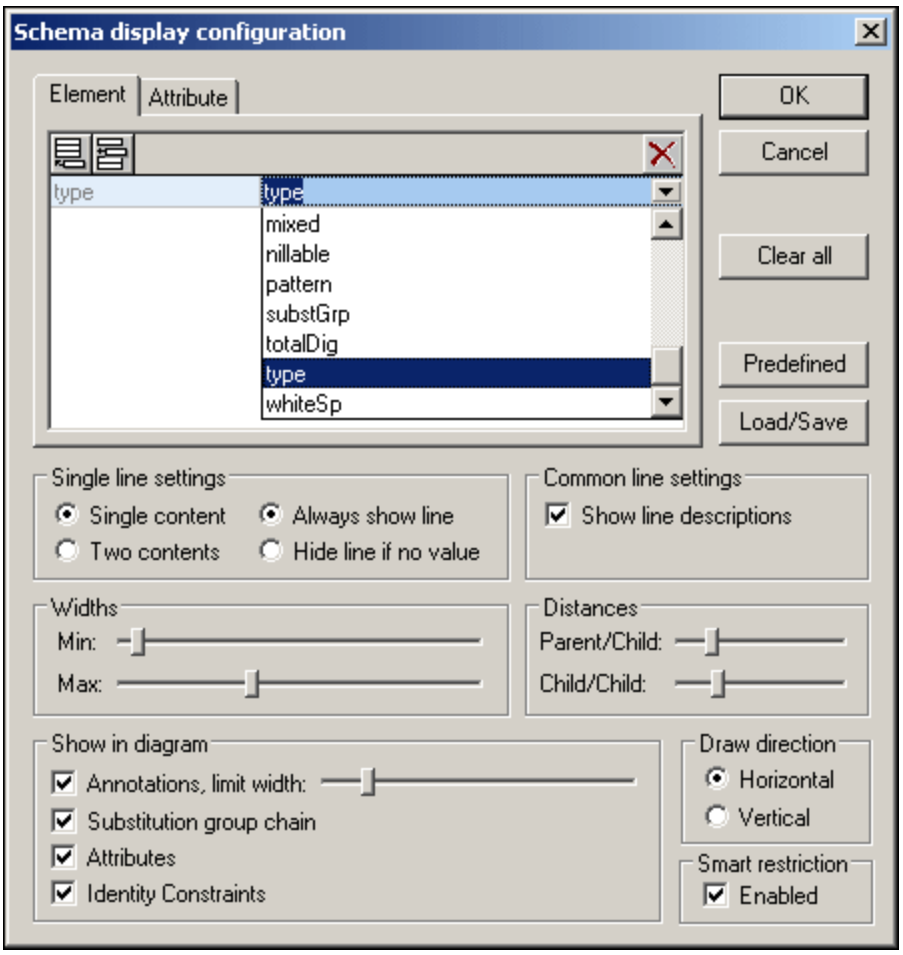
 © 2012 Altova GmbH

2.5 Configuring the Content Model View

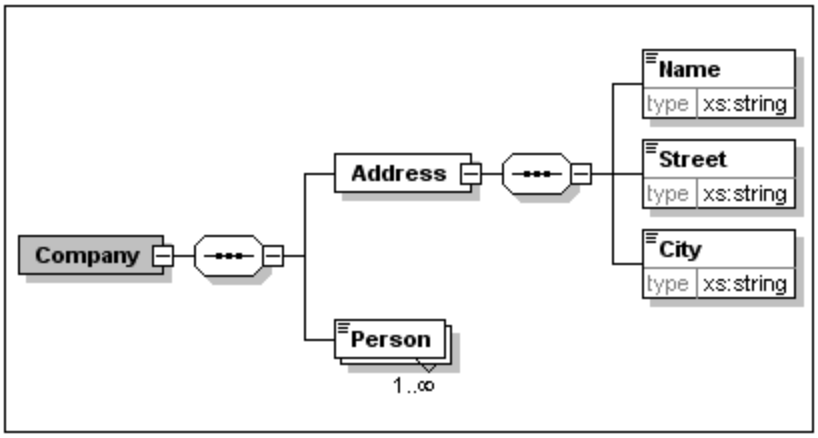
This is a good time to configure the Content Model View. We will configure the Content Model View such that the `type` of the element is displayed for each element.

To configure the Content Model View:

1. Select the Content Model View (click the Content Model View icon) of a component in order to enable the Configure view command.
2. Select the menu option **Schema Design | Configure view**. The Schema Display Configuration dialog appears.

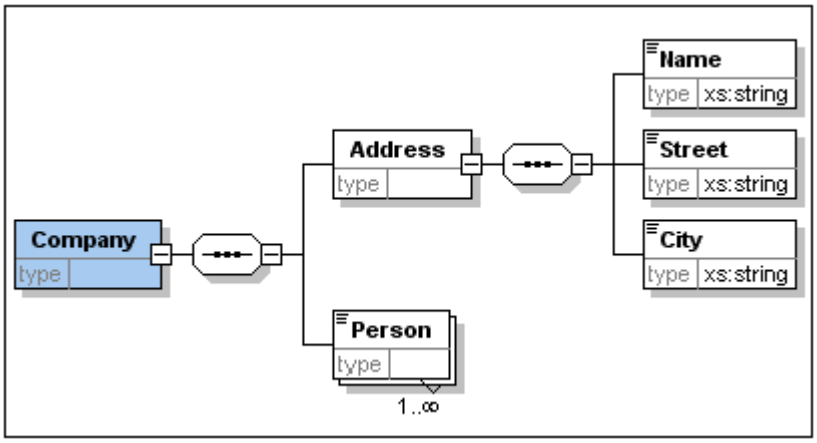


3. Click the **Append** icon (in the **Element** tab) to add a property descriptor line for each element box.
4. From the dropdown menu, select `type` (or double-click in the line and enter "`type`"). This will cause the data type of each element to be displayed in the Content Model View.
5. In the Single Line Settings pane, select Hide Line If No Value. This hides the description of the datatype in the element box if the element does not have a datatype (for example, if the element is a complex type).



Notice that the type descriptor line appears for the `Name`, `Street`, and `City` elements, which are simple types of type `xs:string`, but not for the complex type elements. This is because the Hide Line If No Value toggle is selected.

6. In the Single Line Settings group, select the Always Show Line radio button.
7. Click **OK** to confirm the changes.



Notice that the descriptor line for the data type is always shown—even in element boxes of complex types, where they appear without any value.

Please note:

- The property descriptor lines are editable, so values you enter in them become part of the element definition.
- The settings you define in the Schema display configuration dialog apply to the schema documentation output as well as the printer output.

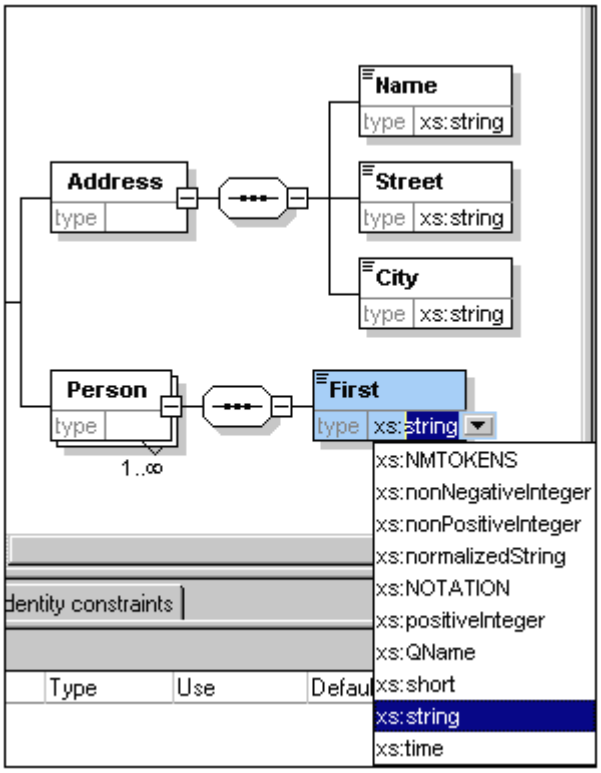
 © 2012 Altova GmbH

2.6 Completing the Basic Schema

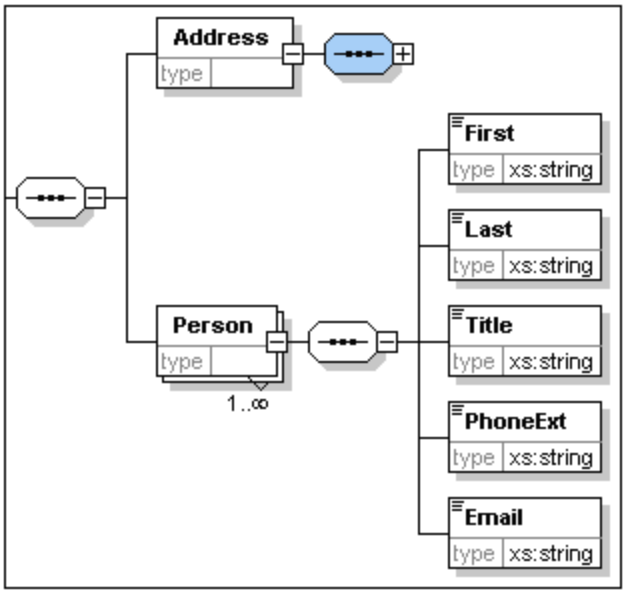
You have defined the content of the `Address` element. Now you need to define the content of the `Person` element. The `Person` element is to contain the following child elements, all of which are simple types: `First`, `Last`, `Title`, `PhoneExt`, and `Email`. All these elements are mandatory except `Title` (which is optional), and they must occur in the order just specified. All should be of datatype `xs:string` except `PhoneExt`, which must be of datatype `xs:integer` and limited to 2 digits.

To create the content model for `Person`:

1. Right-click the `Person` element to open the context menu, and select **Add Child | Sequence**. This inserts the Sequence compositor.
2. Right-click the Sequence compositor, and select **Add Child | Element**.
3. Enter `First` as the name of the element, and press the **Tab** key. This automatically places the cursor in the `type` field.



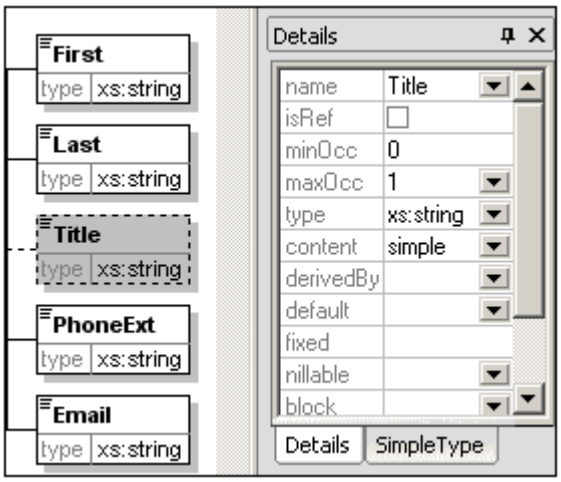
4. Select the `xs:string` entry from the dropdown list or enter it into the `type` value field.
5. Use the drag-and-drop method to create four more elements. Name them `Last`, `Title`, `PhoneExt`, and `Email`, respectively.



Please note: You can select multiple elements by holding down the **Ctrl** key and clicking each of the required elements. This makes it possible to, e.g., copy several elements at once.

Making an element optional

Right-click the `Title` element and select **Optional** from the context menu. The frame of the element box changes from solid to dashed; this is a visual indication that an element is optional.



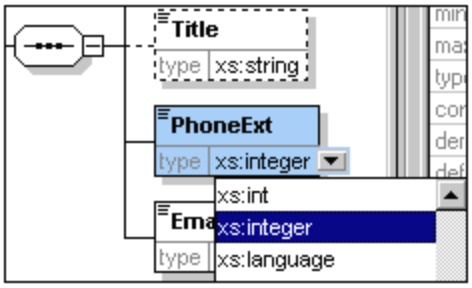
In the Details Entry Helper, you will see that `minOcc=0` and `maxOcc=1`, indicating that the element is optional. Alternatively to using the context menu to make an element optional, you can set `minOcc=0` in order to make the element optional.

Limiting the content of an element

To define the `PhoneExt` element to be of type `xs:integer` and have a maximum of two digits:

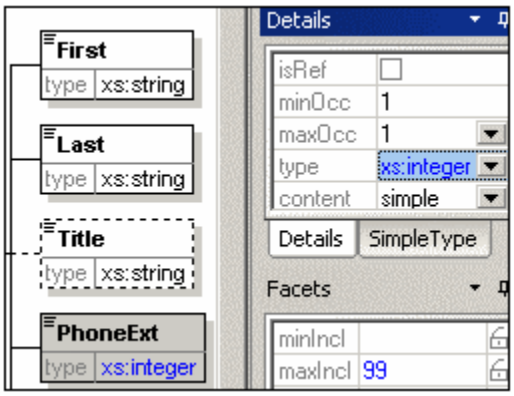
1. Double-click in the `type` field of the `PhoneExt` element, and select (or enter) the `xs:integer` entry from the dropdown list.

 © 2012 Altova GmbH



The items in the Facets Entry Helper change at this point.

2. In the Facets Entry Helper, double-click in the `maxIncl` field and enter `99`. Confirm with **Enter**.



This defines that all phone extensions up to, and including 99, are valid.

3. Select the menu option **File | Save** to save the changes to the schema.

Please note:

- Selecting an XML Schema datatype that is a simple type (for example, `xs:string` or `xs:date`), automatically changes the content model to simple in the Details Entry Helper (`content = simple`).
- Adding a compositor to an element (`sequence`, `choice`, or `all`), automatically changes the content model to complex in the Details Entry Helper (`content = complex`).
- The schema described above is available as `AddressFirst.xsd` in the `C:\Documents and Settings\<username>\My Documents\Altova\XMLSpy2013\\Examples\Tutorial` folder of your XMLSpy application folder.

3 XML Schemas: Advanced

Now that you have created a basic schema, we can move forward to a few advanced aspects of schema development.

Objective
In this section, you will learn how to:

- Work with complex types and simple types, which can then be used as the types of schema elements.
- Create global elements and reference them from other elements.
- Create attributes and their properties, including enumerated values.

You will start this section with the basic `AddressFirst.xsd` schema you created in the first part of this tutorial.

Commands used in this section
In this section of the tutorial, you will use Schema View exclusively. The following commands are used:

Display Diagram (or Display Content Model View). This icon is located to the left of all global components in Schema Overview. Clicking the icon causes the content model of the associated global component to be displayed.

Display All Globals. This icon is located at the top left-hand corner of the Content Model View. Clicking the icon switches the view to Schema Overview, which displays all global components.

Append. The Append icon is located at the top left-hand corner of the Schema Overview. Clicking the icon enables you to add a global component.

 © 2012 Altova GmbH

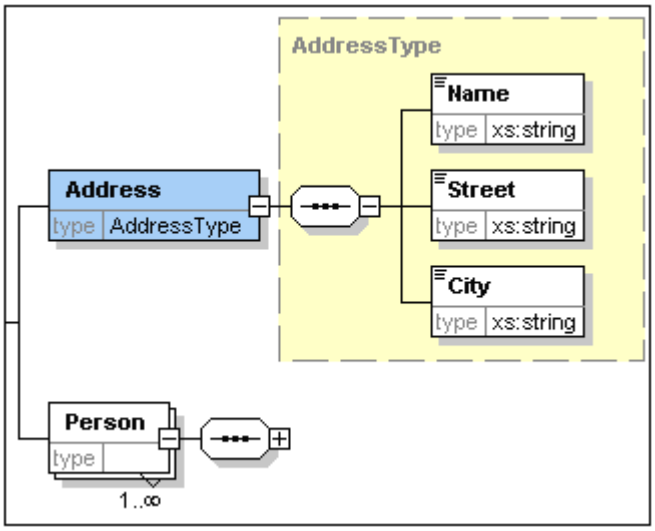
3.1 Working with Complex Types and Simple Types

Having defined the content model of an element, you may decide you want to reuse it elsewhere in your schema. The way to do this is by creating that element definition as a global complex type or as a global element. In this section, you will work with global complex types. You will first create a complex type at the global level and then extend it for use in a content model. You will learn about global elements later in this tutorial.

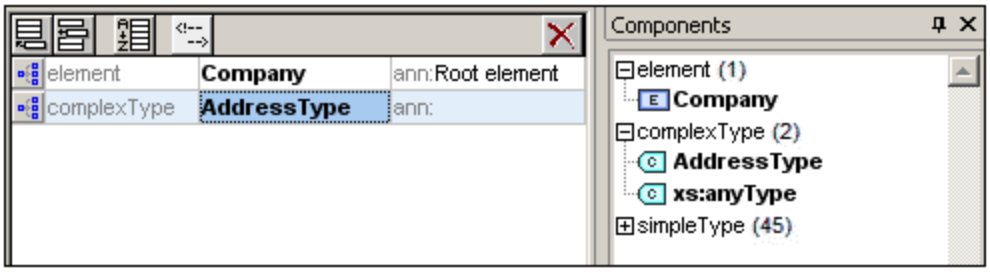
Creating a global complex type
The basic `Address` element that we defined (containing `Name`, `Street`, and `City` elements) can be reused in various address formats. So let us create this element definition as a complex type, which can be reused.

To create a global complex type:

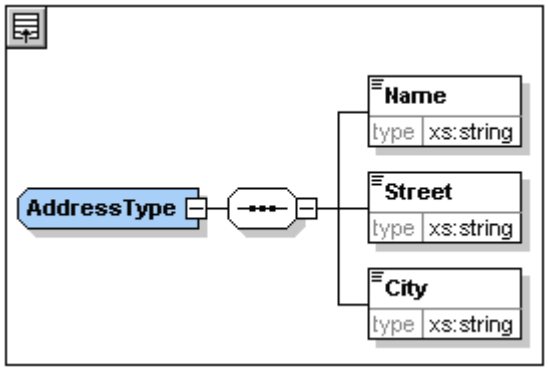
1. In the Content Model View, right-click the `Address` element.
2. In the context menu that now appears, select **Make Global | Complex type**. A global complex type called `AddressType` is created, and the `Address` element in the `Company` content model is assigned this type. The content of the `Address` element is the content model of `AddressType`, which is displayed in a yellow box. Notice that the datatype of the `Address` element is now `AddressType`.



3. Click the Display All Globals icon. This takes you to the Schema Overview, in which you can view all the global components of the schema.
4. Click the expand icons for the **element** and **complexType** entries in the Components entry helper, to see the respective schema constructs.
   The Schema Overview now displays two global components: the `Company` element and the complex type `AddressType`. The Components Entry Helper also displays the `AddressType` complex type.



5. Click on the Content Model View icon of `AddressType` to see its content model (*screenshot below*). Notice the shape of the complex type container.



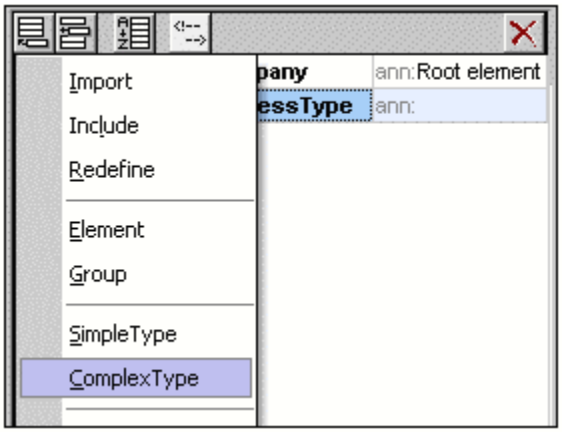
6. Click the Display All Globals icon to return to the Schema Overview.

Extending a complex type definition

We now want to use the global `AddressType` component to create two kinds of country-specific addresses. For this purpose we will define a new complex type based on the basic `AddressType` component, and then extend that definition.

Do this as follows:

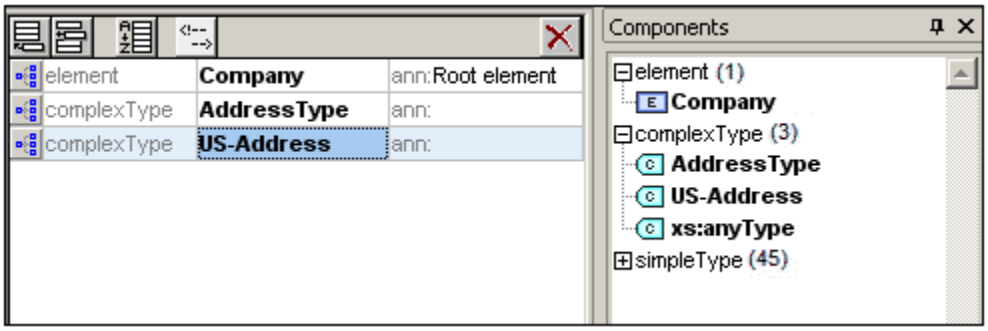
1. Switch to Schema Overview. (If you are in Content Model View, click the Display All Globals icon .)
2. Click the Append icon at the top left of the component window. The following menu opens:



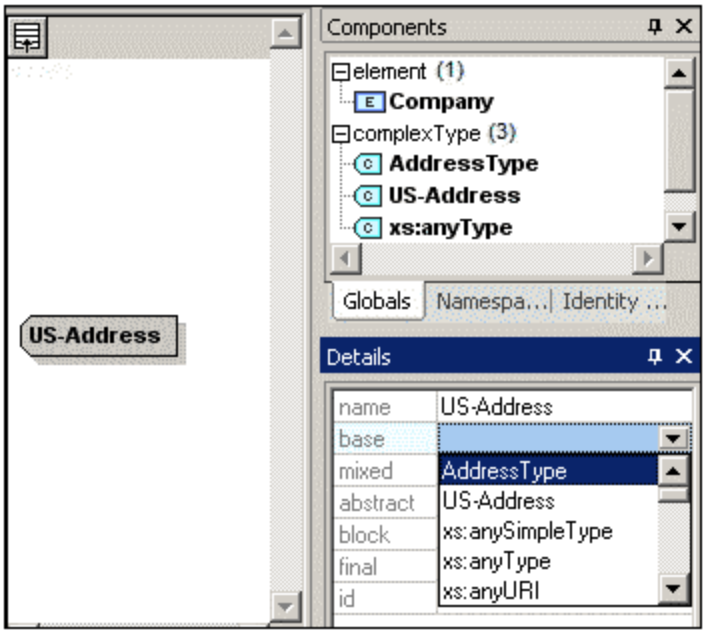
3. Select **ComplexType** from the menu. A new line appears in the component list, and the

cursor is set for you to enter the component name.

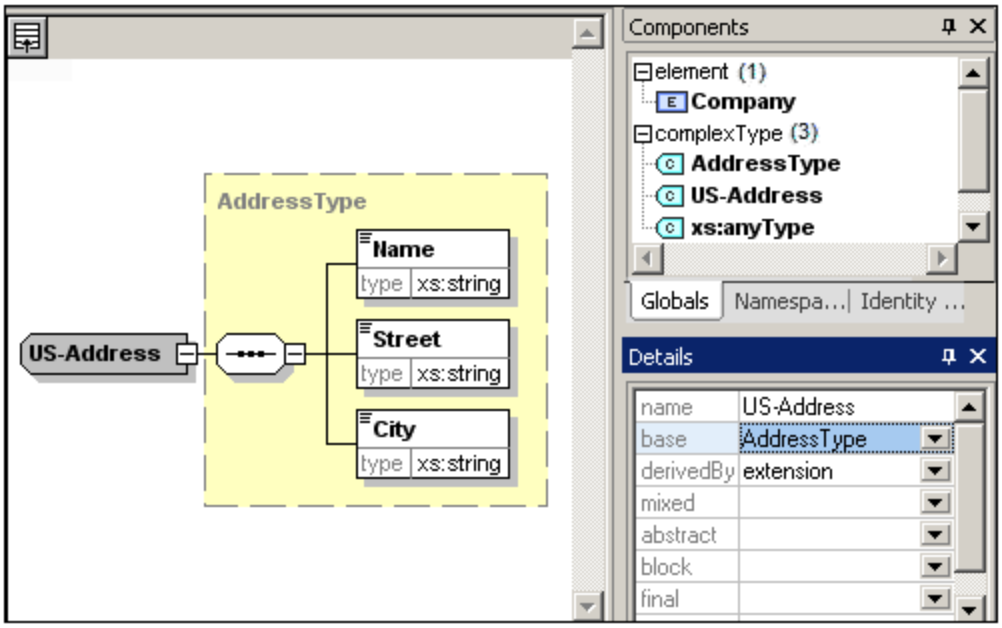
4. Enter `US-Address` and confirm with **Enter**. (If you forget to enter the hyphen character "-" and enter a space, the element name will appear in red, signalling an invalid character.)



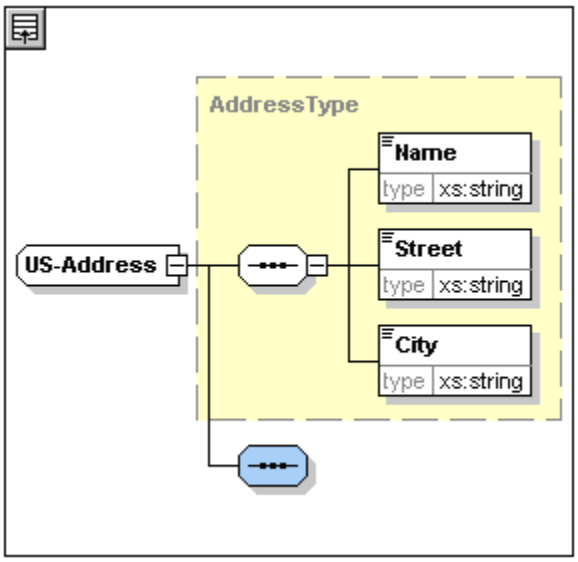
5. Click the Content Model View icon of `US-Address` to see the content model of the new complex type. The content model is empty (*see screenshot below*).
6. In the Details entry helper, click the `base` combo box and select the `AddressType` entry.



The Content Model View now displays the `AddressType` content model as the content model of US-Address (*screenshot below*).

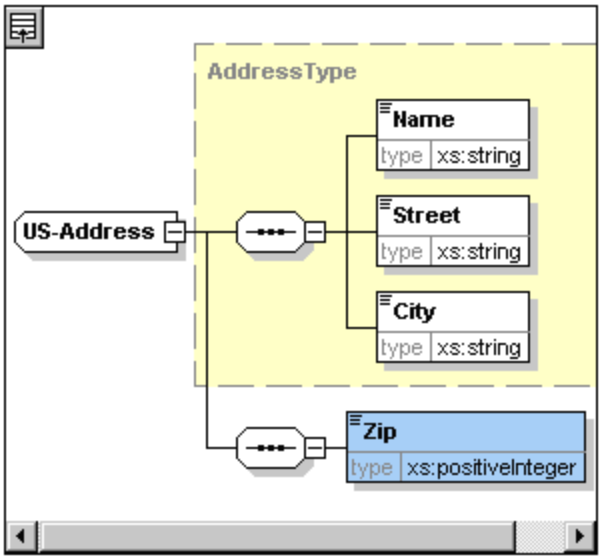


7. Now we can extend the content model of the `US-Address` complex type to take a ZIP Code element. To do this, right-click the `US-Address` component, and, from the context menu that appears, select **Add Child | Sequence**. A new sequence compositor is displayed outside the `AddressType` box (*screenshot below*). This is a visual indication that this is an extension to the element.



8. Right-click the new sequence compositor and select **Add Child | Element**.
9. Name the newly created element `Zip`, and then press the **Tab** key. This places the cursor in the value field of the type descriptor line.
10. Select `xs:positiveInteger` from the dropdown menu that appears, and confirm with **Enter**.

 © 2012 Altova GmbH



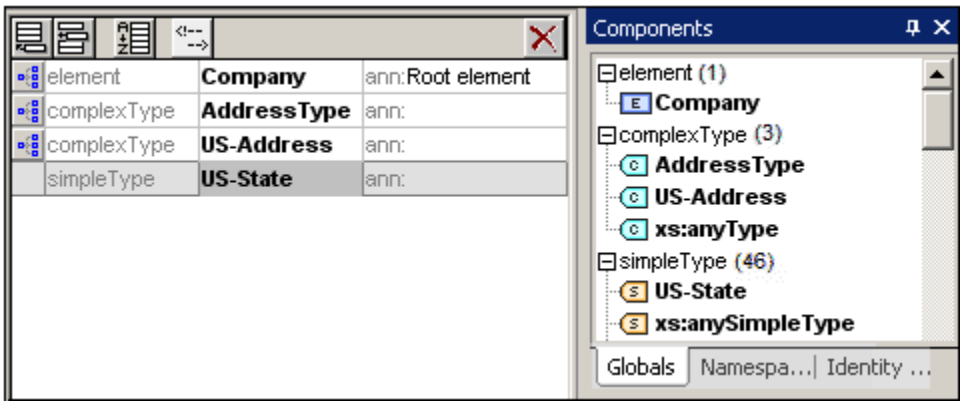
You now have a complex type called `US-Address`, which is based on the complex type `AddressType` and extends it to contain a ZIP code.

Global simple types
Just as the complex type `US-Address` is based on the complex type `AddressType`, an element can also be based on a simple type. The advantage is the same as for global complex types: the simple type can be reused. In order to reuse a simple type, the simple type must be defined globally. In this tutorial, you will define a content model for US states as a simple type. This simple type will be used as the basis for another element.

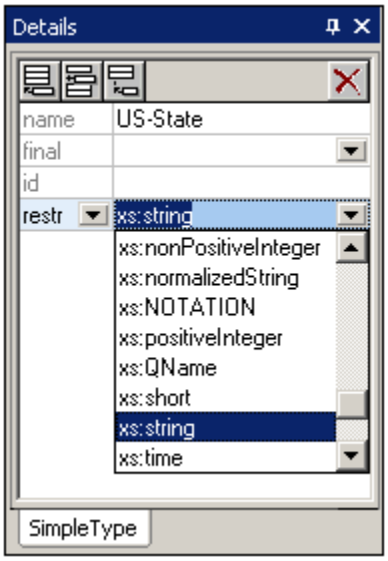
Creating a global simple type
Creating a global simple type consists of appending a new simple type to the list of global components, naming it, and defining its datatype.

To create a global simple type:

1. Switch to Schema Overview. (If you are in Content Model View, click the Display All Globals icon .)
2. Click the Append icon, and in the context menu that appears, select **SimpleType**.
3. Enter `US-State` as the name of the newly created simpleType.
4. Press **Enter** to confirm. The simple type `US-State` is created and appears in the list of simple types in the Components Entry Helper (Click the expand icon of the simpleType entry to see it).



5. In the Details Entry Helper (*screenshot below*), place the cursor in the value field of `restr` and enter `xs:string`, or select `xs:string` from the dropdown menu in the `restr` value field.



This creates a simple type called `US-State`, which is of datatype `xs:string`. This global component can now be used in the content model of `US-Address`.

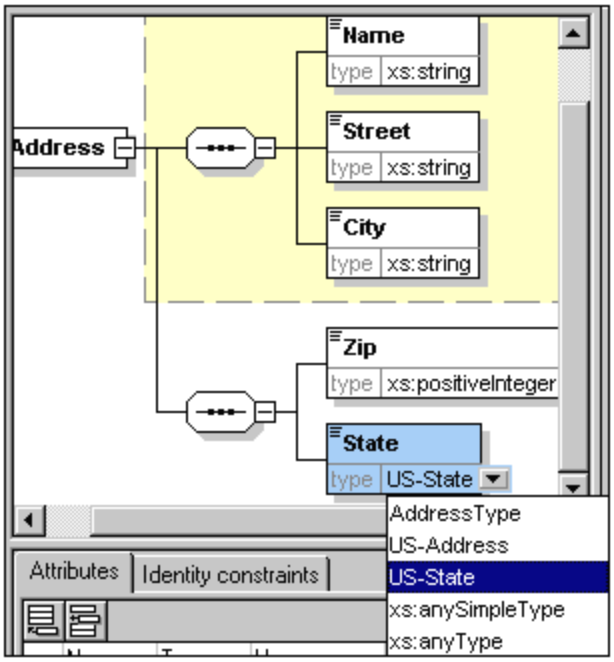
Using a global simple type in a content model

A global simple type can be used in a content model to define the type of a component. We will use `US-State` to define an element called `State` in the content model of `US-Address`.

Do the following:

1. In Schema Overview, click the Component Model View icon of `US-Address`.
2. Right-click the lower sequence compositor and select **Add Child | Element**.
3. Enter `State` for the element name.
4. Press the **Tab** key to place the cursor in the value field of the type descriptor line.
5. From the drop-down menu of this combo box, select `US-State`.

 © 2012 Altova GmbH



The `State` element is now based on the `US-State` simple type.

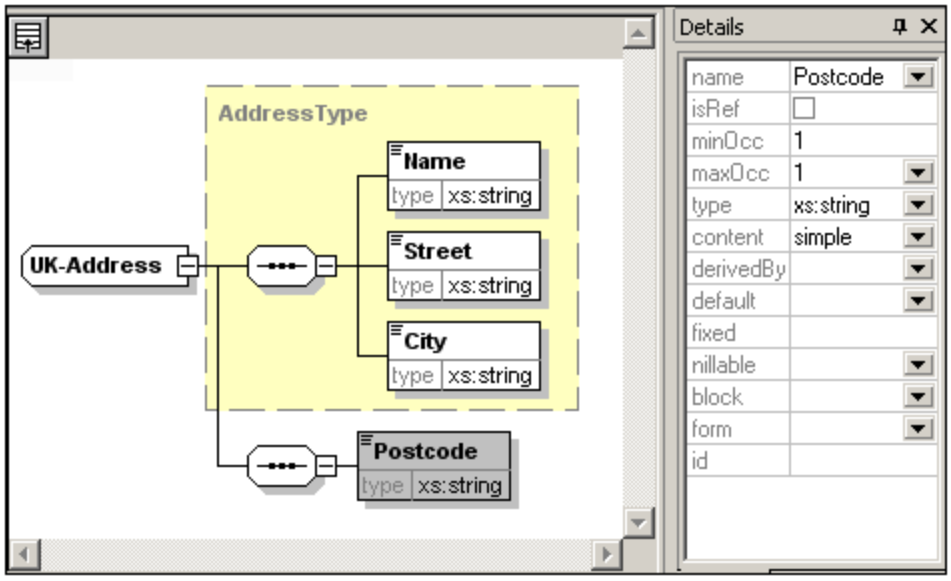
Creating a second complex type based on `AddressType`

We will now create a global complex type to hold UK addresses. The complex type is based on `AddressType`, and is extended to match the UK address format.

Do the following:

1. In Schema Overview, create a global complex type called `UK-Address`, and base it on `AddressType` (`base=AddressType`).
2. In the Content Model View of `UK-Address`, add a `Postcode` element and give it a type of `xs:string`.

Your `UK-Address` content model should look like this:



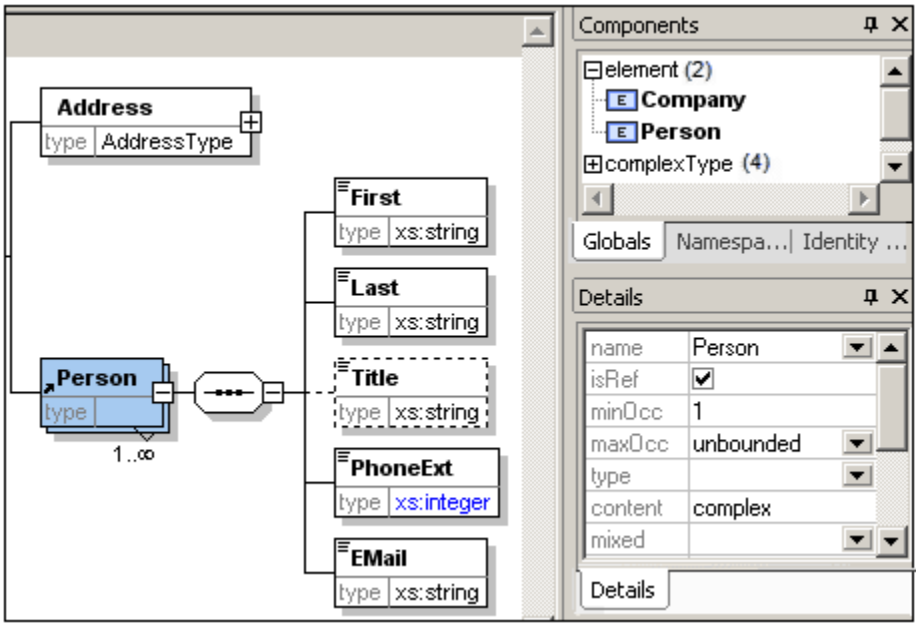
Please note: In this section you created global simple and complex types, which you then used in content model definitions. The advantage of global types is that they can be reused in multiple definitions.

© 2012 Altova GmbH

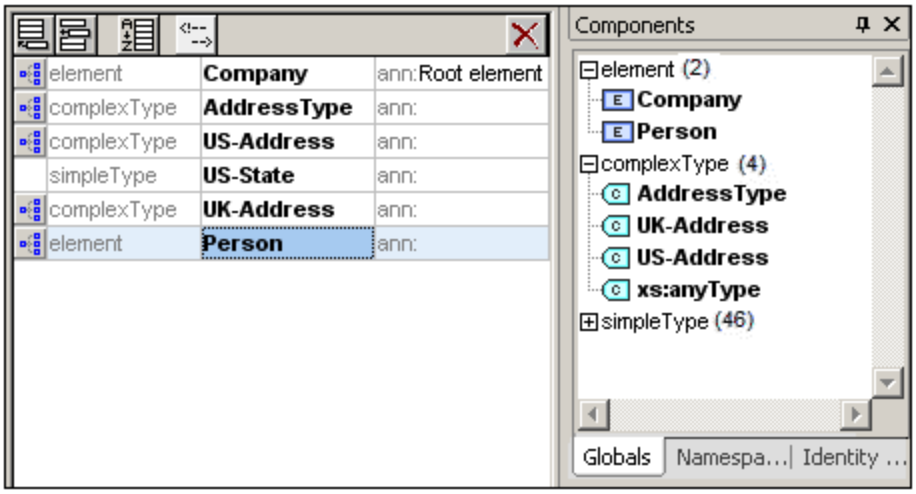
3.2 Referencing Global Elements

In this section, we will convert the locally defined `Person` element to a global element and reference that global element from within the `Company` element.

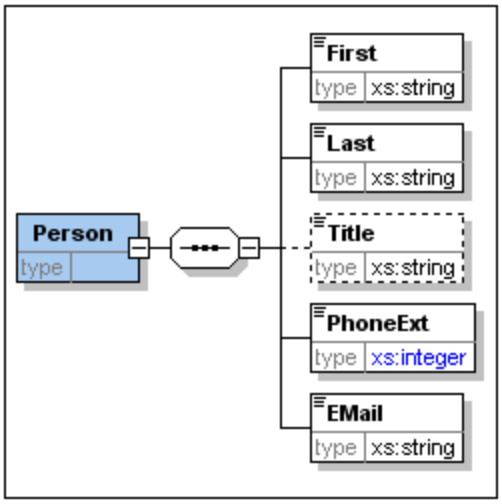
1. Click (Display All Globals) to switch to Schema Overview.
2. Click the Display Diagram icon of the `Company` element.
3. Right-click the `Person` element, and select **Make Global | Element**. A small link arrow icon appears in the `Person` element, showing that this element now references the globally declared `Person` element. In the Details Entry Helper, the `isRef` check box is now activated.



4. Click the Display All Globals icon to return to Schema Overview. The `Person` element is now listed as a global element. It is also listed in the Components Entry Helper.



5. In the Components Entry Helper, double-click the `Person` element to see the content model of the global `Person` element.



Notice that the global element box does **not** have a link arrow icon. This is because it is the referenced element, not the referencing element. It is the referencing element that has the link arrow icon.

Please note:

- An element that references a global element must have the same name as the global element it references.
- A global declaration does not describe where a component is to be used in an XML document. It only describes a content model. It is only when a global declaration is referenced from within another component that its location in the XML document is specified.

  A globally declared element can be reused at multiple locations. It differs from a globally declared complex type in that its content model cannot be modified without also modifying the global element itself. If you change the content model of an element that references a global element, then the content model of the global element will also be changed, and, with it, the content model of all other elements that reference that global element.

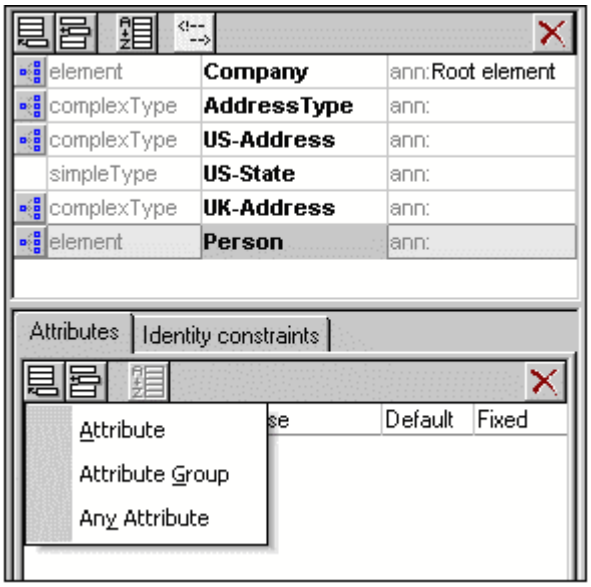
 © 2012 Altova GmbH

3.3 Attributes and Attribute Enumerations

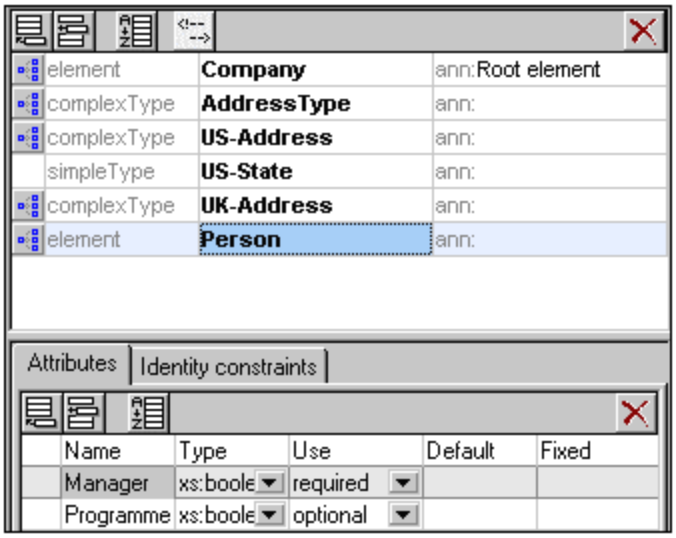
In this section, you will learn how to create attributes and enumerations for attributes.

Defining element attributes

1. In the Schema Overview, click the `Person` element to make it active.
2. Click the Append icon , in the top left of the Attributes/Identity Constraints tab group (in the lower part of the Schema Overview window), and select the Attribute entry.



3. Enter `Manager` as the attribute name in the Name field.
4. Use the `Type` combo box to select `xs:boolean`.
5. Use the `Use` combo box to select `required`.

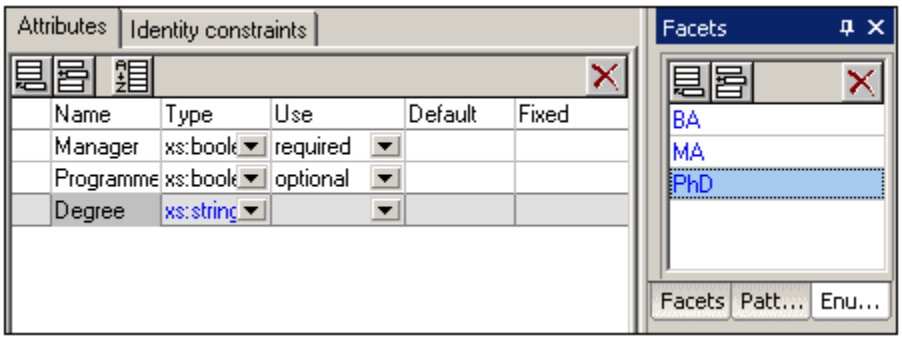


6. Use the same procedure to create a `Programmer` attribute with `Type=xs:boolean` and `Use=optional`.

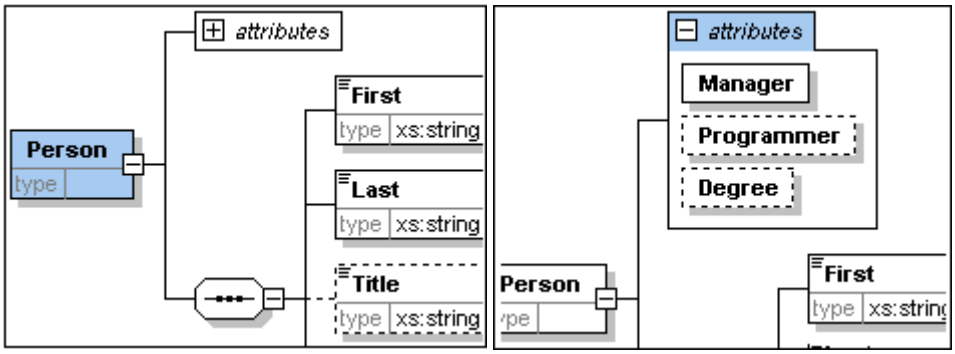
Defining enumerations for attributes
Enumerations are values allowed for a given attribute. If the value of the attribute in the XML instance document is not one of the enumerations specified in the XML Schema, then the document is invalid. We will create enumerations for the `Degree` attribute of the `Person` element.

Do the following:

1. In the Schema Overview, click the `Person` element to make it active.
2. Click the Append icon in the top left of the Attributes window, and select the **Attribute** entry.
3. Enter `Degree` as the attribute name, and select `xs:string` as its type.
4. With the `Degree` attribute selected, in the Facets Entry Helper, click the **Enumerations** tab (*see screenshot*).



5. In the **Enumerations** tab, click the Append icon .
6. Enter `BA`, and confirm with **Enter**.
7. Use the same procedure to add two more enumerations: `MA` and `PhD`.
8. Click on the Content Model View icon of `Person`.



The previously defined attributes are visible in the Content Model View. Clicking the expand icon displays all the attributes defined for that element. This display mode and the Attributes tab can be toggled by selecting the menu option **Schema Design | Configure view**, and checking and unchecking the **Attributes** check box in the **Show in diagram** pane.

9. Click the Display all Globals icon to return to the Schema Overview.

Saving the completed XML Schema

Please note: Before saving your schema file, rename the `AddressLast.xsd` file that is delivered with XMLSpy to something else (such as `AddressLast_original.xsd`), so as not to

 © 2012 Altova GmbH

overwrite it.

Save the completed schema with any name you like (**File | Save as**). We recommend you save it with the name `AddressLast.xsd` since the XML file you create in the next part of the tutorial will be based on the `AddressLast.xsd` schema.

4 XML Schemas: XMLSpy Features

After having completed the XML Schema, we suggest you become familiar with a few navigation shortcuts and learn about the schema documentation that you can generate from within XMLSpy. These are described in the subsections of this section.

Commands used in this section
In this section of the tutorial, you will use Schema View exclusively. The following commands are used:

Display Diagram (or Display Content Model View). This icon is located to the left of all global components in Schema Overview. Clicking the icon causes the content model of the associated global component to be displayed.

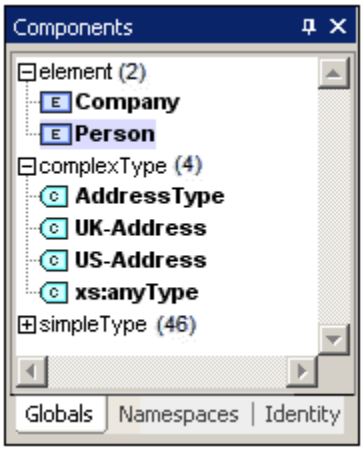
© 2012 Altova GmbH

4.1 Schema Navigation

This section shows you how to navigate Schema View efficiently. We suggest that you try out these navigation mechanisms to become familiar with them.

Displaying the content model of a global component
Global components that can have content models are complex types, elements, and element groups. The Content Model View of these components can be opened in the following ways:

- In Schema Overview, click the Display Diagram icon to the left of the component name.
- In either Schema Overview or Content Model View, double-click the element, complex type, or element group in the Components Entry Helper (*screenshot below*). This displays the content model of that component.

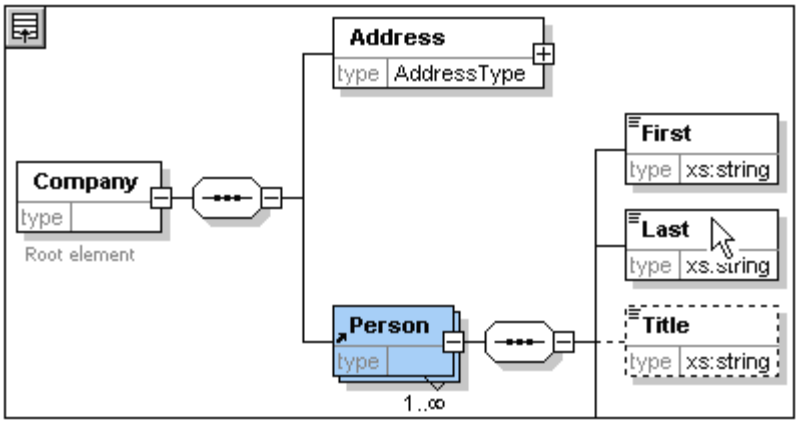


If you double-click any of the other global components (simple type, attribute, attribute group) in the Components Entry Helper, that component will be highlighted in Schema Overview (since such a component would not have a content model).

In the Components Entry Helper, the double-clicking mechanism works in both the Globals and Namespaces tabs.

Going to the definition of a global element from a referencing element
If a content model contains an element that references a global element, you can go directly to the content model of that global element or to any of its contained components by holding down **Ctrl** and double-clicking the required element.

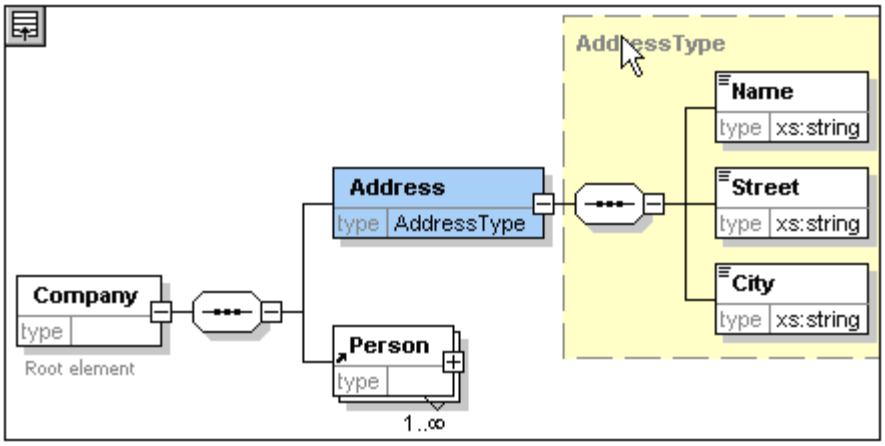
For example, while viewing the `Company` content model, holding down **Ctrl** while double-clicking `Last` opens the `Person` content model and highlights the `Last` element in it.



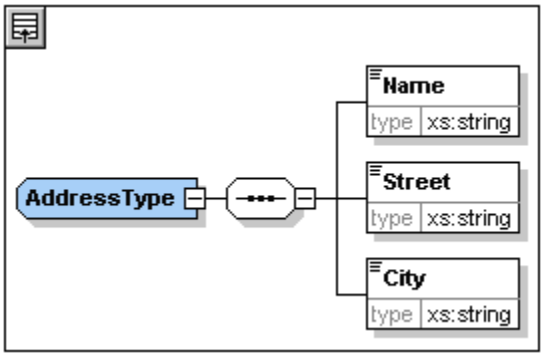
When the `Last` element is highlighted, all its properties are immediately displayed in the relevant entry helpers and information window.

Going to the definition of a complex type

Complex types are often used as the type of some element within a content model. To go directly to the definition of a complex type from within a content model, double-click the **name** of the complex type in the yellow box (*see mouse pointer in screenshot below*).



This takes you to the Content Model View of the complex type.



Please note: Just as with referenced global elements, you can go directly to an element within the complex type definition by holding down **Ctrl** and double-clicking the required element in the content model that contains the complex type.

 © 2012 Altova GmbH

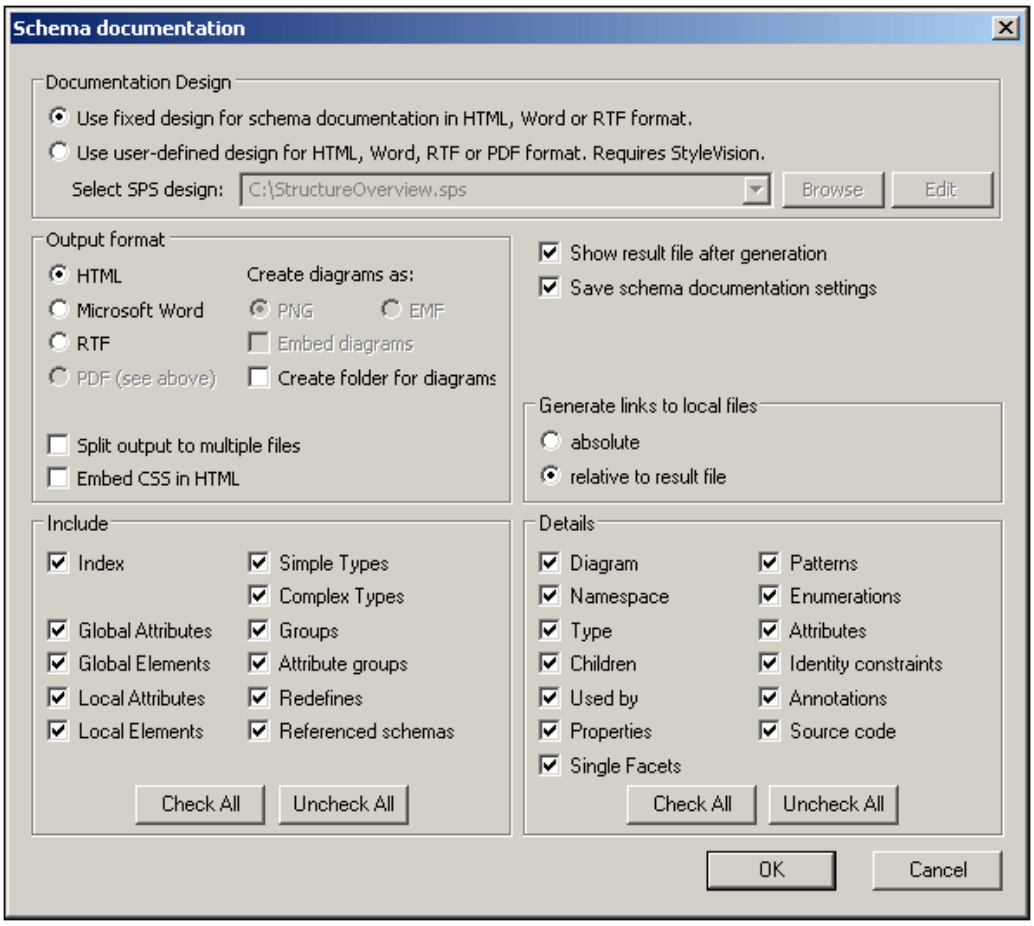
4.2 Schema Documentation

XMLSpy provides detailed documentation of XML Schemas in HTML and Microsoft Word (MS Word) formats. You can select the components and the level of detail you want documented. Related components are hyperlinked in both HTML and MS Word documents. In order to generate MS Word documentation, you must have MS Word installed on your computer (or network).

In this section, we will generate documentation for the `AddressLast.xsd` XML Schema.

Do the following:

1. Select the menu option **Schema design | Generate documentation**. This opens the Schema Documentation dialog.



2. For the Output Format option, select HTML, and click **OK**.
3. In the Save As dialog, select the location where the file is to be saved and give the file a suitable name (say `AddressLast.html`). Then click the **Save** button.

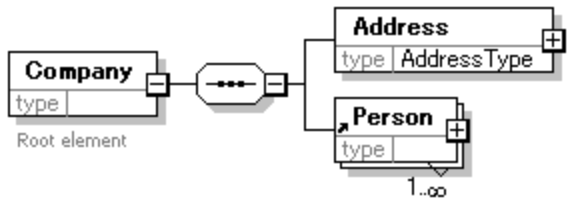
   The HTML document appears in the Browser View of XMLSpy. Click on a link to go to the corresponding linked component.

Schema **AddressLast.xsd**

schema location:	**C:¥Program Files¥Altova¥XMLSpy2010 ¥Examples¥Tutorial¥AddressLast.xsd**
attribute form default:	**unqualified**
element form default:	**qualified**
targetNamespace:	**http://my-company.com/namespace**

Elements	Complex types	Simple types
Company	**AddressType**	**US-State**
Person	**UK-Address**	
	US-Address	

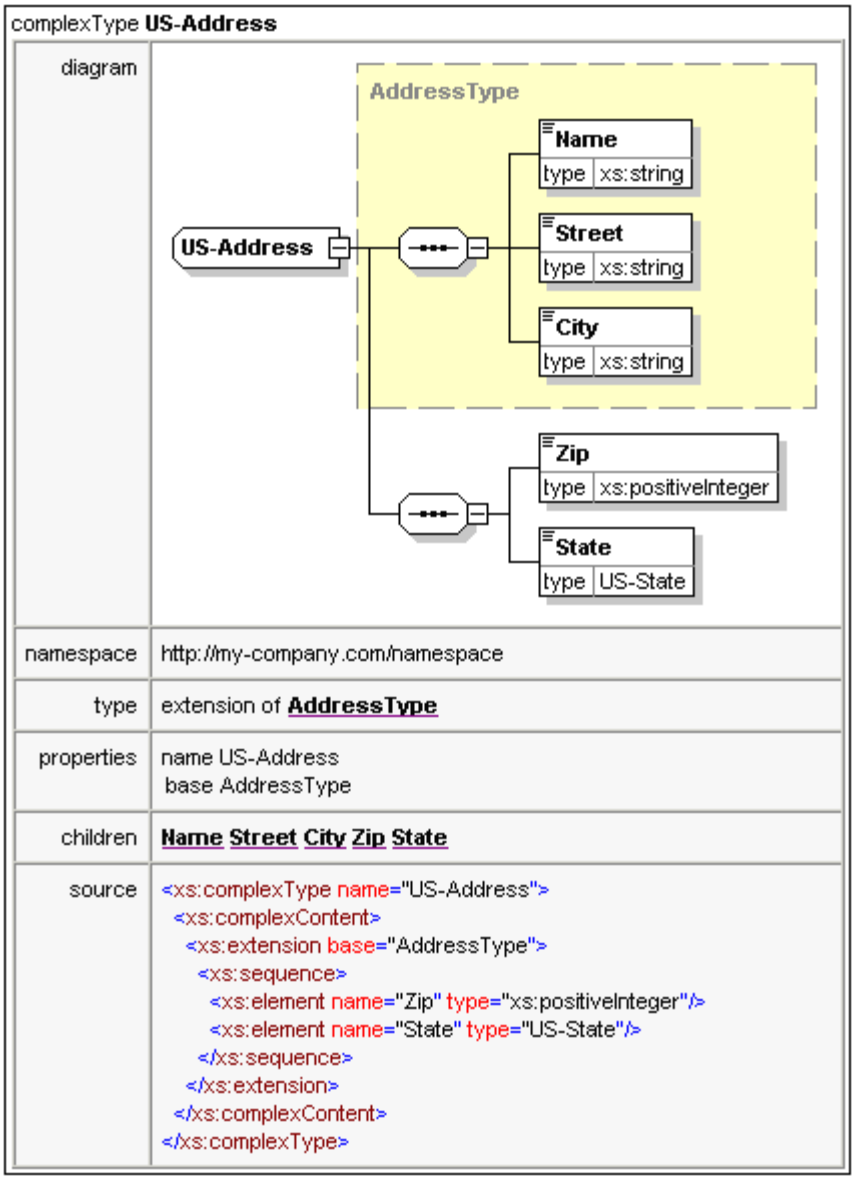
element **Company**

diagram	
namespace	http://my-company.com/namespace
properties	content complex
children	**Address** **Person**
annotation	documentation Root element

source	`<xs:element name="Company">`
	`  <xs:annotation>`
	`  <xs:documentation>Root element</xs:documentation>`
	`  </xs:annotation>`
	`  <xs:complexType>`
	`  <xs:sequence>`
	`  <xs:element name="Address" type="AddressType"/>`
	`  <xs:element ref="Person" maxOccurs="unbounded"/>`
	`  </xs:sequence>`
	`  </xs:complexType>`
	`</xs:element>`

The diagram above shows the **first page** of the schema documentation in HTML form. If components from other schemas have been included, then those schemas are also documented.

 © 2012 Altova GmbH



The diagram above shows how complex types are documented.

element **US-Address/Zip**

diagram	**Zip** type xs:positiveInteger
namespace	http://my-company.com/namespace
type	**xs:positiveInteger**
properties	name Zip isRef 0 content simple
source	<xs:element name="Zip" type="xs:positiveInteger"/>

element **US-Address/State**

diagram	**State** type US-State
namespace	http://my-company.com/namespace
type	**US-State**
properties	name State isRef 0 content simple
source	<xs:element name="State" type="US-State"/>

simpleType **US-State**

namespace	http://my-company.com/namespace
type	**xs:string**
properties	name US-State
used by	element **US-Address/State**
source	<xs:simpleType name="US-State"> <xs:restriction base="xs:string"/> </xs:simpleType>

The diagram above shows how elements and simple types are documented.

You can now try out the MS Word output option. The Word document will open in MS Word. To use hyperlinks in the MS Word document, hold down **Ctrl** while clicking the link.

 © 2012 Altova GmbH

5 XML Documents

In this section you will learn how to create and work with XML documents in XMLSpy. You will also learn how to use the various intelligent editing features of XMLSpy.

Objective
The objectives of this section are to learn how to do the following:

- Create a new XML document based on the `AddressLast.xsd` schema.
- Specify the type of an element so as to make an extended content model for that element available to the element during validation.
- Insert elements and attributes and enter content for them in Grid View and Text View using intelligent entry helpers.
- Copy XML data from XMLSpy to Microsoft Excel; add new data in MS Excel; and copy the modified data from MS Excel back to XMLSpy. This functionality is available in the Database/Table View of Grid View.
- Sort XML elements using the sort functionality of Database/Table View.
- Validate the XML document.
- Modify the schema to allow for three-digit phone extensions.

Commands used in this section
In this section of the tutorial, you will mostly use the Grid View and Text View, and in one section Schema View. The following commands are used:

File | New. Creates a new type of XML file.

View | Text View. Switches to Text View.

View | Grid View. Switches to Enhanced Grid View.

XML | Table | Display as Table. Displays multiple occurrences of a single element type at a single hierarchic level as a table. This view of the element is called the Database/Table View (or simply Table View). The icon is used to switch between the Table View and regular Grid View.

F7. Checks for well-formedness.

F8. Validates the XML document against the associated DTD or Schema.

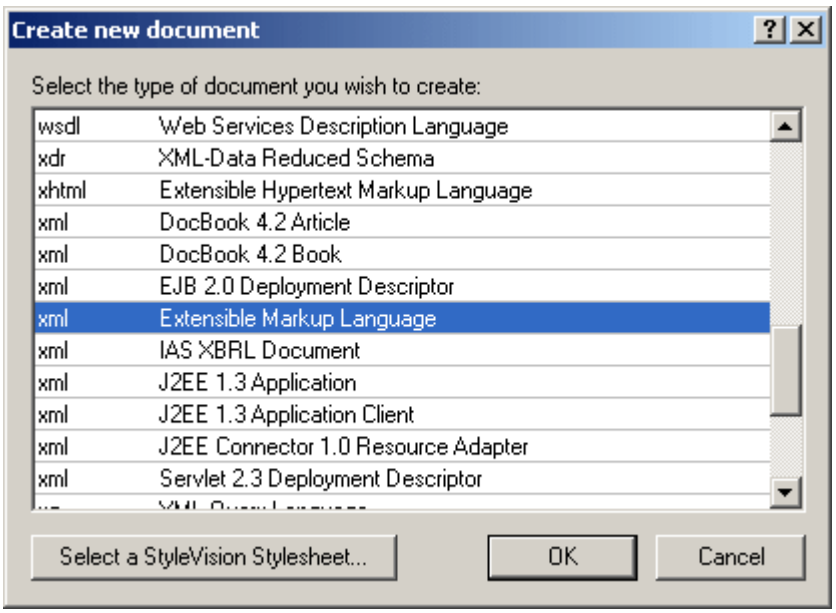
Opens the associated DTD or XML Schema file.

5.1 Creating a New XML File

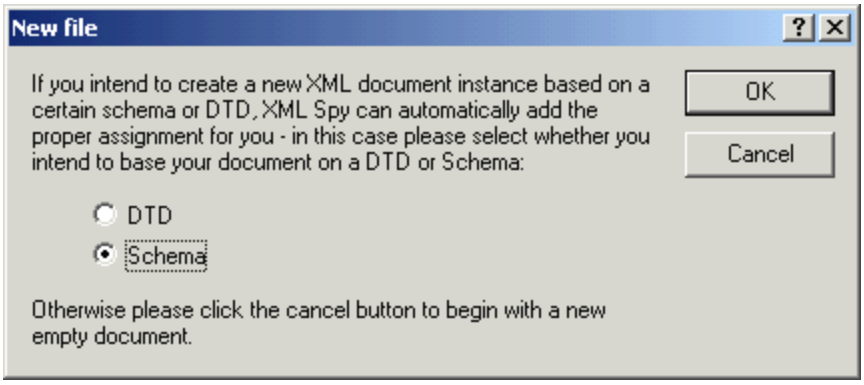
When you create a new XML file in XMLSpy, you are given the option of basing it on a schema (DTD or XML Schema) or not. In this section you will create a new file that is based on the `AddressLast.xsd` schema you created earlier in the tutorial.

To create the new XML file:

1. Select the menu option **File | New**. The Create new document dialog opens.

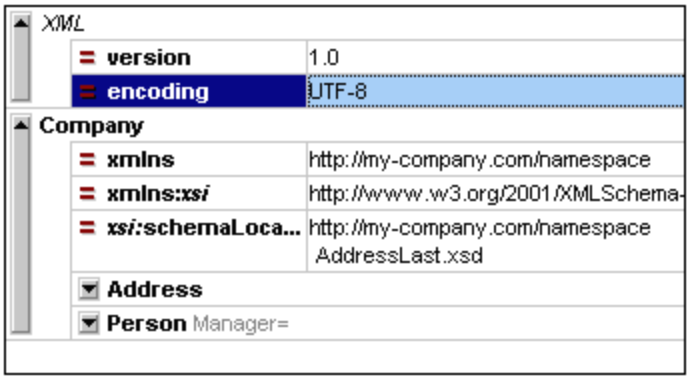


2. Select the `Extensible Markup Language` entry (or generic XML document entry) from the dialog, and confirm with **OK**. A prompt appears, asking if you want to base the XML document on a DTD or Schema.

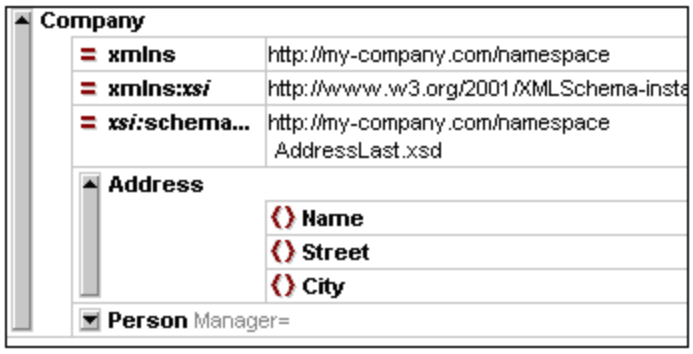


3. Click the Schema radio button, and confirm with **OK**. A further dialog appears, asking you to select the schema file your XML document is to be based on.
4. Use the Browse or Window buttons to find the schema file. The Window button lists all files open in XMLSpy and projects. Select `AddressLast.xsd` (see Tutorial introduction for location), and confirm with **OK**. An XML document containing the main elements defined by the schema opens in the main window.
5. Click the **Grid** tab to select Grid View.
6. In Grid View, notice the structure of the document. Click on any element to reduce selection to that element. Your document should look something like this:

 © 2012 Altova GmbH

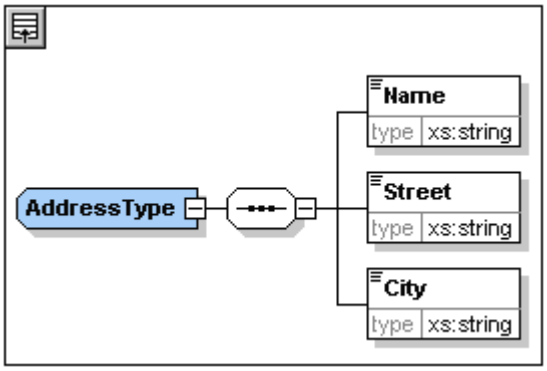


7. Click on the icon next to `Address`, to view the child elements of `Address`. Your document should look like this:

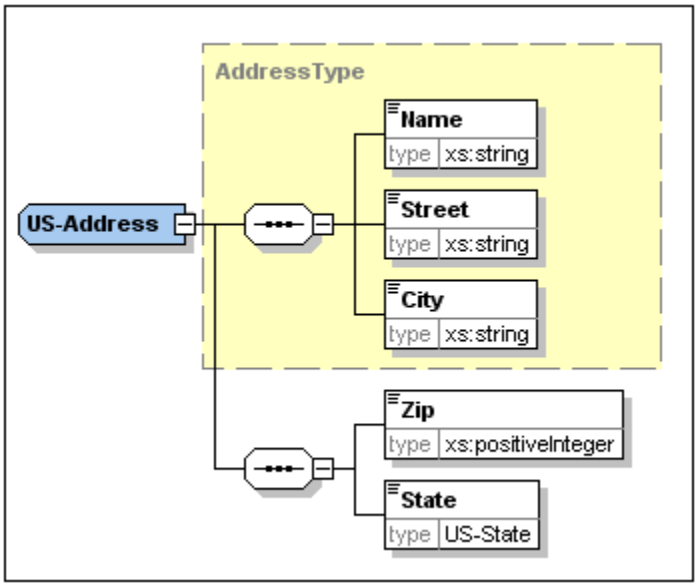


5.2 Specifying the Type of an Element

The child elements of `Address` are those defined for the global complex type `AddressType` (*the content model of which is defined in the XML Schema* `AddressLast.xsd` *shown in the Schema View screenshot below*).



We would, however, like to use a specific US or UK address type rather than the generic address type. You will recall that, in the `AddressLast.xsd` schema, we created global complex types for `US-Address` and `UK-Address` by extending the `AddressType` complex type. The content model of `US-Address` is shown below.



In the XML document, in order to specify that the `Address` element must conform to one of the extended `Address` types (`US-Address` or `UK-Address`) rather than the generic `AddressType`, we must specify the required extended complex type as an attribute of the `Address` element.

Do the following:

1. In the XML document, right-click the `Name` element, and select **Insert | Attribute** from the context menu.

An attribute field is added to the `Address` element.

2. Ensure that `xsi:type` is entered as the name of the attribute (*screenshot below*).
3. Press **Tab** to move into the next (value) field.

4. Enter `US-Address` as the value of the attribute.

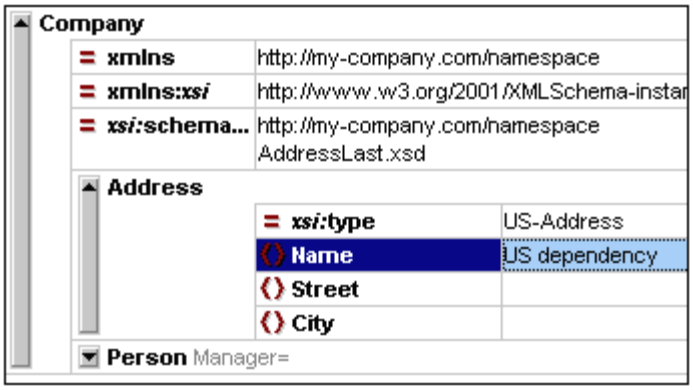
Please note: The `xsi` prefix allows you to use special XML Schema related commands in your XML document instance. Notice that the the namespace for the `xsi` prefix was automatically added to the document element when you assigned a schema to your XML file. In the above case, you have specified a type for the `Address` element. See the XML Schema specification for more information.

5.3 Entering Data in Grid View

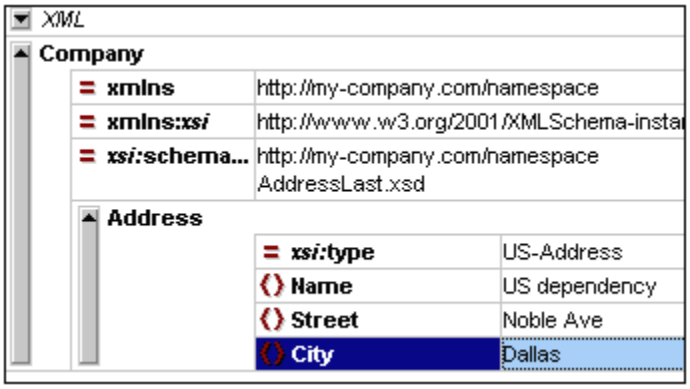
You can now enter data into your XML document.

Do the following:

1. Double-click in the `Name` value field (or use the arrow keys) and enter `US dependency`. Confirm with **Enter**.



2. Use the same method to enter a `Street` and `City` name (for example, `Noble Ave` and `Dallas`).
3. Click the `Person` element and press **Delete** to delete the `Person` element. (We will add it back in the next section of the tutorial.) After you do this, the entire `Address` element is highlighted.
4. Click on any child element of the `Address` element to deselect all the child elements of `Address` except the selected element. Your XML document should look like this:

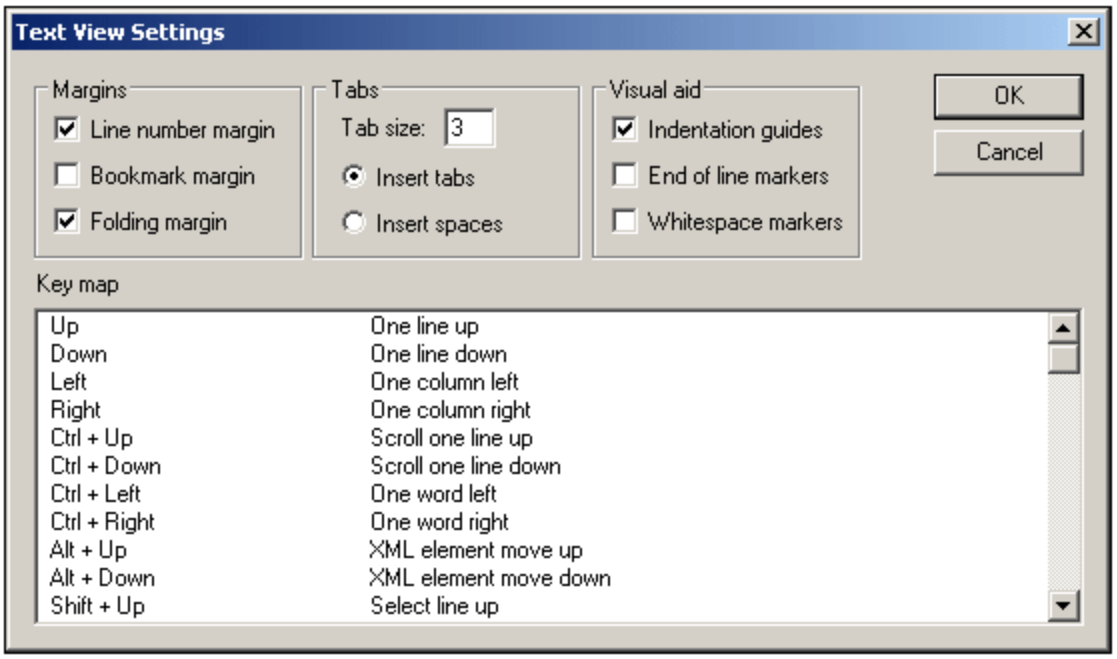


 © 2012 Altova GmbH

5.4 Entering Data in Text View

Text View is ideal for editing the actual data and markup of XML files because of its DTD/XML Schema-related intelligent editing features.

Structural layout features
In addition, Text View has a number of viewing and structural editing features that make editing large sections of text easy. These features can be switched on and off in the Text View Settings dialog (**View | Text View Settings**, *screenshot below*).



The following margins in Text View can be switched on and off:

- Line number margins
- Bookmark margins, in which individual lines can be highlighted with a marker
- Source folding margins, which contain icons to expand and collapse the display of elements

Additionally, visual aids such as indentation guides, end-of-line markers, and whitespace markers can be switched on and off, by checking and unchecking, respectively, their check boxes in the *Visual Aid* pane of the Text View Settings dialog (*see screenshot above*).

The bookmark feature is useful for setting up markers in your document. To insert a bookmark, use the command **Edit | Insert/Remove Bookmark**. Once bookmarks have been inserted you can navigate these bookmarks using commands in the **Edit** menu.

The screenshot below shows the current XML file in Text View with all structural editing features enabled. For the sake of clarity, none of the line numbers, indentation guides, etc, will be shown in Text View in rest of this tutorial. Please see the User Manual for more information on Text View.

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <Company xmlns="http://my-company.com/nam
3   C:\PROGRA~1\Altova\XMLSPY2004\Examples\
4     <Address xsi:type="US-Address">
5       <Name>US dependency</Name>
6       <Street>Noble Ave</Street>
7       <City>Dallas</City>
8     </Address>
```

Editing in Text View
In this section, you will enter and edit data in Text View in order to become familiar with the features of Text View.

Do the following:

1. Select the menu item **View | Text view**, or click on the **Text** tab. You now see the XML document in its text form, with syntax coloring.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.0.1 U
(http://www.xmlspy.com) by Alexander Pilz (private) -->
<Company xmlns="http://my-company.com/namespace"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://my-
company.com/namespace
AddressLast.xsd">
    <Address xsi:type="US-Address">
        <Name>US dependency</Name>
        <Street>Noble Ave.</Street>
        <City>Dallas</City>
```

Text | Grid | Schema/... | Authentic | Browser

Untitled1.xml

2. Place the text cursor after the end tag of the `Address` element, and press **Enter** to add a new line.
3. Enter the less-than angular bracket **<** at this position. A dropdown list of all elements allowed at that point (according to the schema) is displayed. Since only the `Person` element is allowed at this point, it will be the only element displayed in the list.

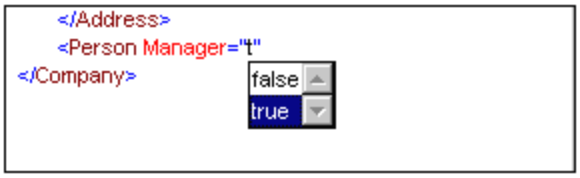
```
<Address xsi:type="US-Address">
  <Name>US dependency</Name>
  <Street>Noble Ave</Street>
  <City>Dallas</City>
</Address>
<
</Person
```

4. Select the `Person` entry. The `Person` element, as well as its attribute `Manager`, are inserted, with the cursor inside the value-field of the `Manager` attribute.

```
    </Address>
    <Person Manager="|"
</Company>
```

5. From the dropdown list for the `Manager` attribute, select `true`.

 © 2012 Altova GmbH

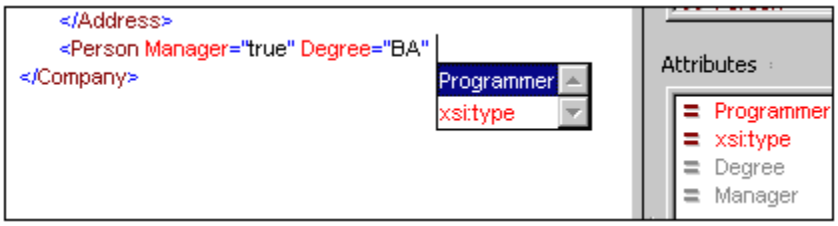


Press **Enter** to insert the value `true` at the cursor position.

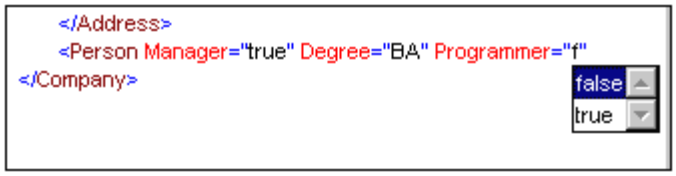
6. Move the cursor to the end of the line (using the **End** key if you like), and press the space bar. This opens a dropdown list, this time containing a list of attributes allowed at that point. Also, in the Attributes Entry Helper, the available attributes are listed in red. The `Manager` attribute is grayed out because it has already been used.

7. Select `Degree` with the Down arrow key, and press **Enter**. This opens another list box, from which you can select one of the predefined enumerations (`BA`, `MA`, or `PhD`).

8. Select `BA` with the Down arrow key and confirm with **Enter**. Then move the cursor to the end of the line (with the **End** key), and press the space bar. `Manager` and `Degree` are now grayed out in the Attributes Entry Helper.



9. Select `Programmer` with the Down arrow key and press **Enter**.



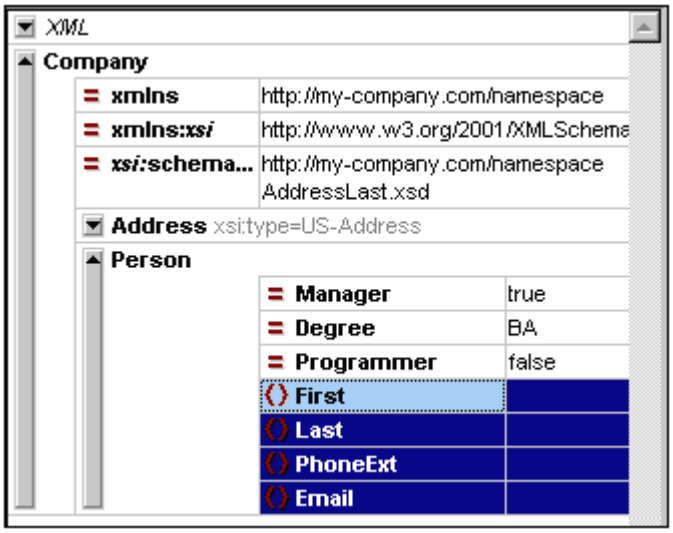
10. Enter the letter `"f"` and press **Enter**.
11. Move the cursor to the end of the line (with the **End** key), and enter the greater-than angular bracket **>**. XMLSpy automatically inserts all the required child elements of `Person`. (Note that the optional `Title` element is not inserted.) Each element has start and end tags but no content.

```
<?xml version="1.0" encoding="UTF-8"?>
<Company xmlns="http://my-company.com/namespace"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://my-company.com/namespace
AddressLast.xsd">
    <Address xsi:type="US-Address">
        <Name>US dependency</Name>
        <Street>Noble Ave</Street>
        <City>Dallas</City>
    </Address>
    <Person Manager="true" Degree="BA" Programmer="false">
        <First></First>
        <Last></Last>
        <PhoneExt></PhoneExt>
        <Email></Email>
    </Person>
</Company>
```

You could now enter the `Person` data in Text View, but let's move to Grid View to see the flexibility of moving between views when editing a document.

Switching to Grid View

To switch to Grid View, select the menu item **View | Grid View,** or click the **Grid** tab. The newly added child elements of `Person` are highlighted.



Now let us validate the document and correct any errors that the validation finds.

© 2012 Altova GmbH

5.5 Validating the Document

XMLSpy provides two evaluations of the XML document:

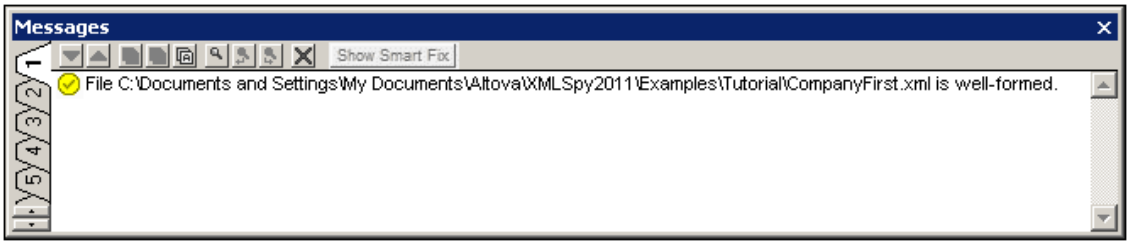
- A well-formedness check
- A validation check

If either of these checks fails, we will have to modify the document appropriately.

Checking well-formedness
An XML document is well-formed if starting tags match closing tags, elements are nested correctly, there are no misplaced or missing characters (such as an entity without its semi-colon delimiter), etc.

You can do a well-formedness check in any editing view. Let us select Text View. To do a well-formedness check, select the menu option **XML | Check well-formedness**, or press the **F7** key, or click . A message appears in the Messages window at the bottom of the Main Window saying the document is well-formed.

Notice that the output of the Messages window has 9 tabs. The validation output is always displayed in the active tab. Therefore, you can check well-formedness in Tab1 for one schema file and keep the result by switching to Tab2 before validating the next schema document (otherwise Tab1 is overwritten with the validation result).

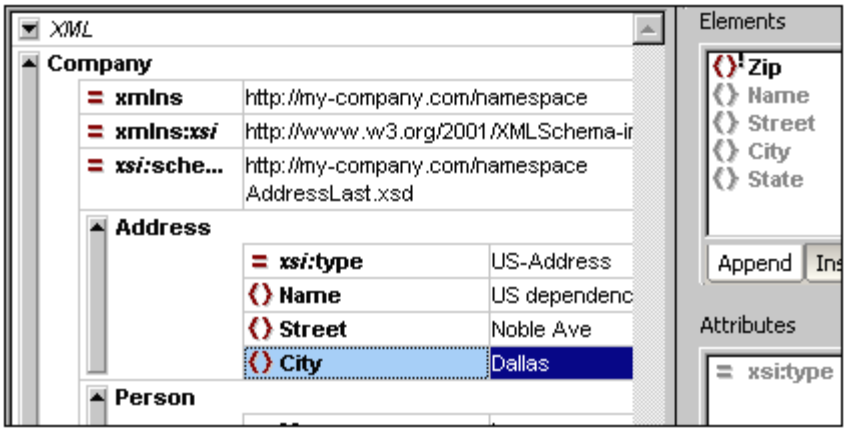


Please note: This check does not check the structure of the XML file for conformance with the schema. Schema conformance is evaluated in the validity check.

Checking validity
An XML document is valid according to a schema if it conforms to the structure and content specified in that schema.

To check the validity of your XML document, first select Grid View, then select the menu option **XML | Validate**, or press the **F8** key, or click . An error message appears in the Messages window saying the file is not valid. Mandatory elements are expected after the `City` element in `Address`. If you check your schema, you will see that the `US-Address` complex type (which you have set this `Address` element to be with its `xsi:type` attribute) has a content model in which the City element must be followed by a `Zip` element and a `State` element.

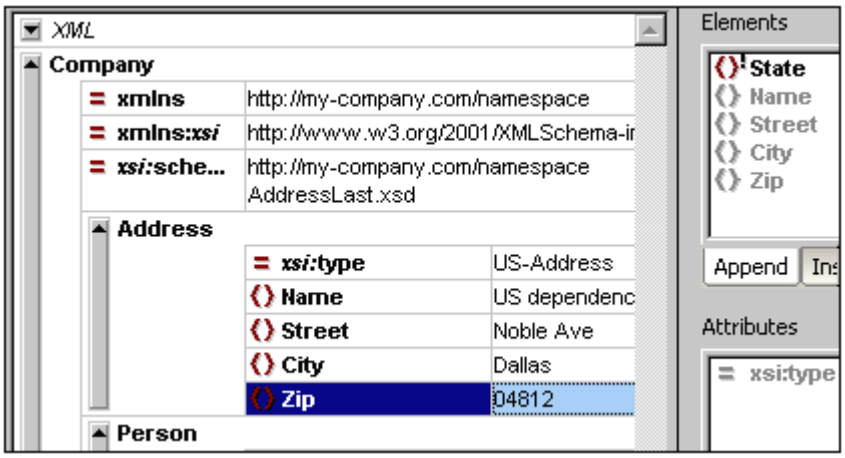
Fixing the invalid document
The point at which the document becomes invalid is highlighted, in this case the `City` element.



Now look at the Elements Entry Helper (at top right). Notice that the `Zip` element is prefixed with an exclamation mark, which indicates that the element is mandatory in the current context.

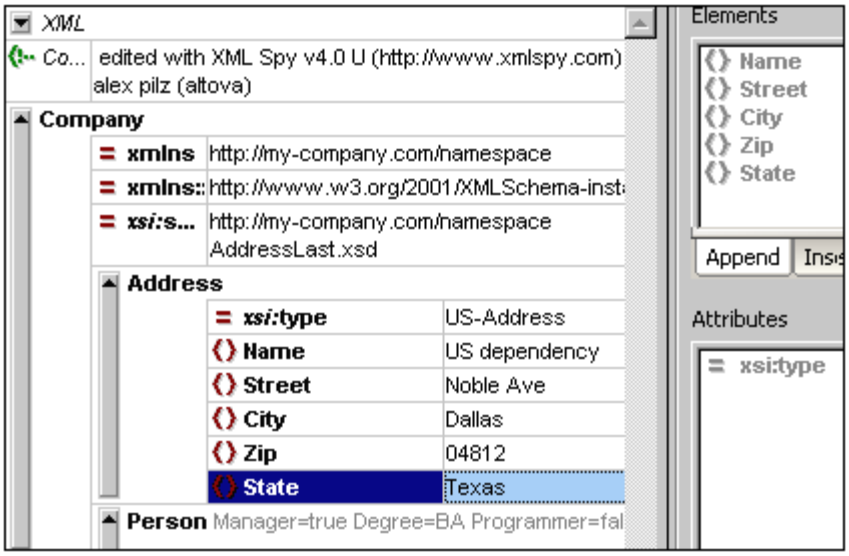
To fix the validation error:

1. In the Elements Entry Helper, double-click the `Zip` element. This inserts the `Zip` element after the `City` element (we were in the Append tab of the Elements Entry Helper).
2. Press the **Tab** key, and enter the Zip Code of the State (`04812`), then confirm with **Enter**. The Elements Entry Helper now shows that the `State` element is mandatory (it is prefixed with an exclamation mark). *See screenshot below*.



3. In the Elements Entry Helper, double-click the `State` element. Then press **Tab** and enter the name of the state (`Texas`). Confirm with **Enter**. The Elements Entry Helper now contains only grayed-out elements. This shows that there are no more required child elements of `Address`.

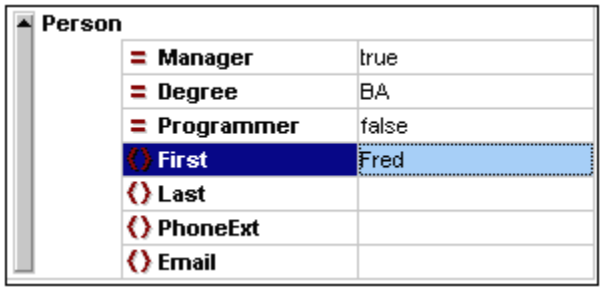
 © 2012 Altova GmbH



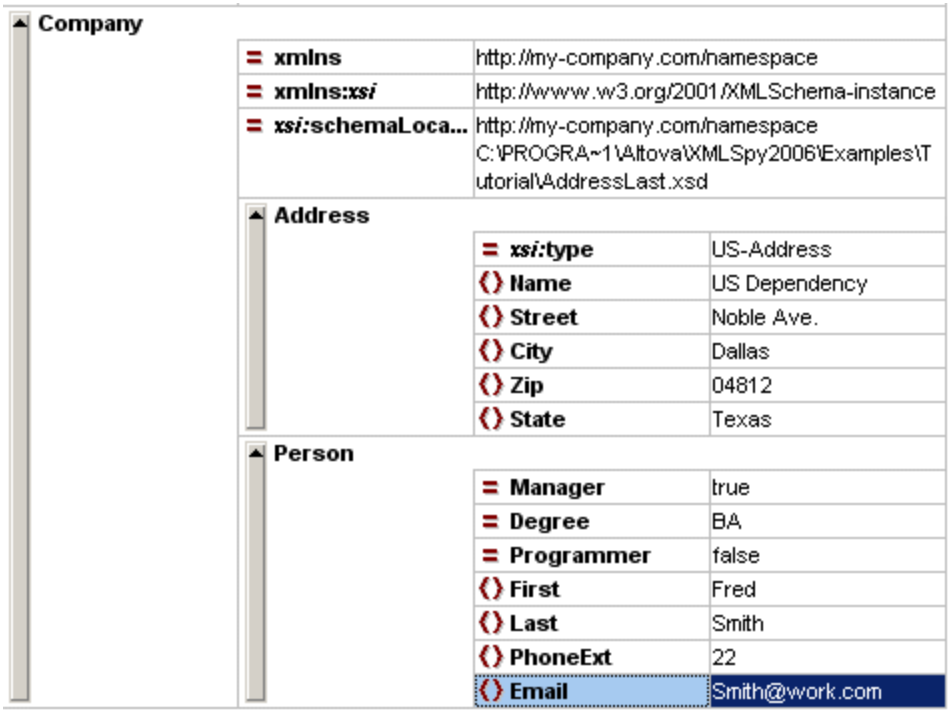
Completing the document and revalidating
Let us now complete the document (enter data for the `Person` element) before revalidating.

Do the following:

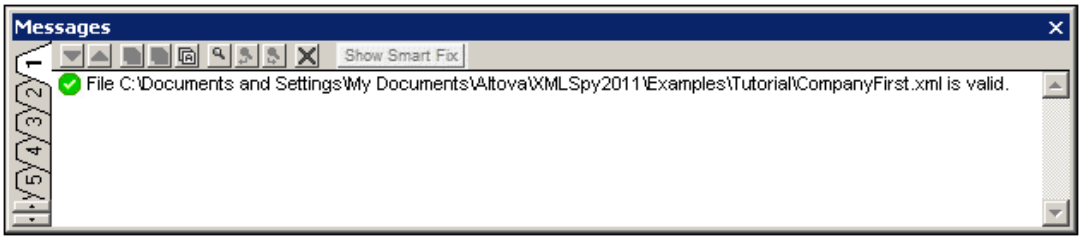
1. Click the value field of the element `First`, and enter a first name (say `Fred`). Then press **Enter**.



2. In the same way enter data for all the child elements of `Person`, that is, for `Last`, `PhoneExt`, and `Email`. Note that the value of `PhoneExt` must be an integer with a maximum value of 99 (since this is the range of allowed `PhoneExt` values you defined in your schema). Your XML document should then look something like this in Grid View:



3. Click again to check if the document is valid. A message appears in the Messages window stating that the file is valid. The XML document is now valid against its schema.



4. Select the menu option **File | Save** and give your XML document a suitable name (for example `CompanyFirst.xml`). Note that the finished tutorial file `CompanyFirst.xml` is in the `Tutorial` folder, so you may need to rename it before you give that name to the file you have created.

Please note: An XML document does not have to be valid in order to save it. Saving an invalid document causes a prompt to appear warning you that you are about to save an invalid document. You can select **Save anyway**, if you wish to save the document in its current invalid state.

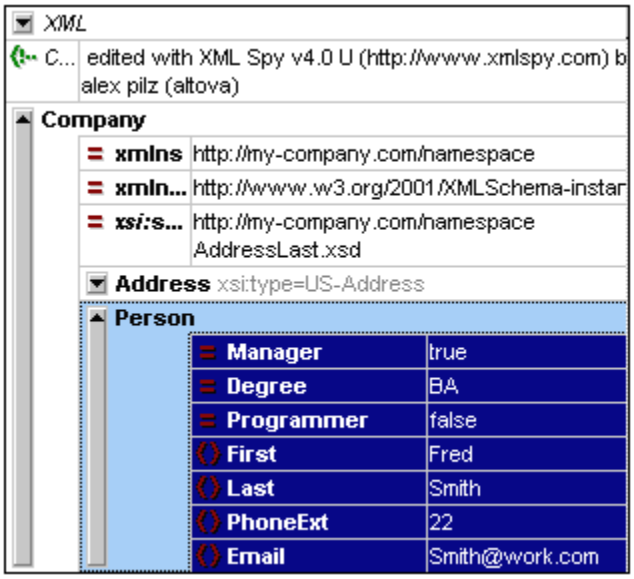
 © 2012 Altova GmbH

5.6 Adding Elements and Attributes

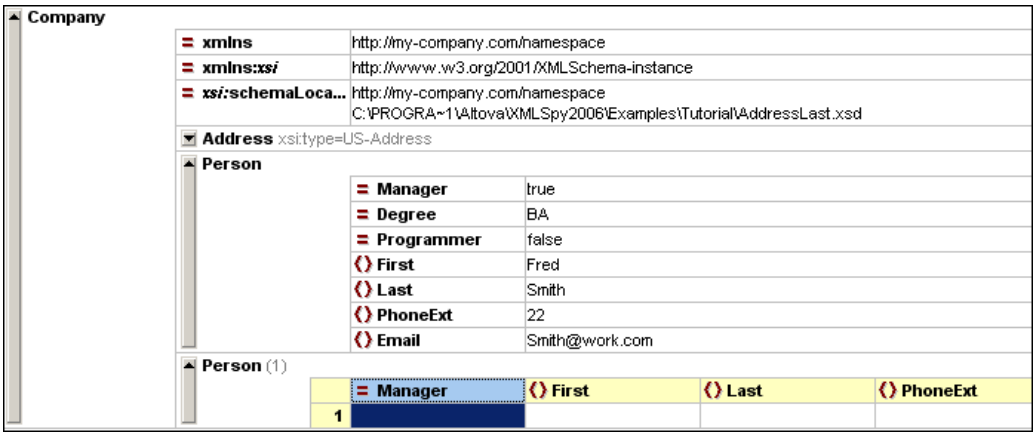
At this point, there is only one `Person` element in the document.

To add a new `Person` element:

1. Click the gray sidebar to the left of the `Address` element to collapse the `Address` element. This clears up some space in the view.
2. Select the entire `Person` element by clicking on or below the `Person` element text in Grid View. Notice that the `Person` element is now available in the **Append** tab of the Elements Entry Helper.



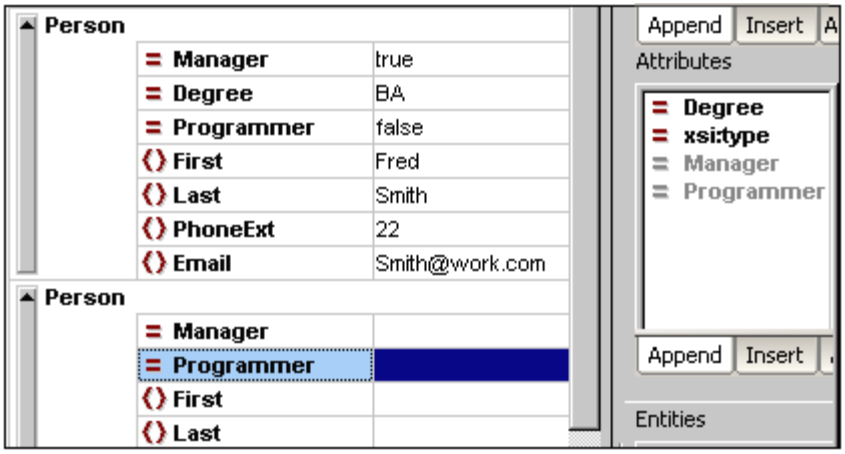
3. Double-click the `Person` element in the Elements Entry Helper. A new `Person` element with all mandatory child elements is appended (*screenshot below*). Notice that the optional `Title` child element of `Person` is not inserted.



4. Switch to Grid View and then press **F12** to switch the new `Person` element from Table View to normal Grid View.
5. Click on the `Manager` attribute of the new `Person` element. Take a look at the Attributes Entry Helper. The Manager entry is grayed out because it has already been entered. Also look at the Info Window, which now displays information about the `Manager` attribute. It is a required attribute and has therefore been added. The `Programmer`

attribute has not been added.

6. In the **Append** tab of the Attributes Entry Helper, double-click the `Programmer` entry. This inserts an empty `Programmer` attribute after the `Manager` attribute.



The `Programmer` attribute is now grayed out in the Attributes Entry Helper.

You could enter content for the `Person` element in this view, but let's switch to the Database/Table View of Grid View since it is more suited to editing a structure with multiple occurrences, such as `Person`.

 © 2012 Altova GmbH

5.7 Editing in Database/Table View

Grid View contains a special view called Database/Table View (hereafter called Table View), which is convenient for editing elements with multiple occurrences. Individual element types can be displayed as a table. When an element type is displayed as a table, its children (attributes and elements) are displayed as columns, and the occurrences themselves are displayed as rows.

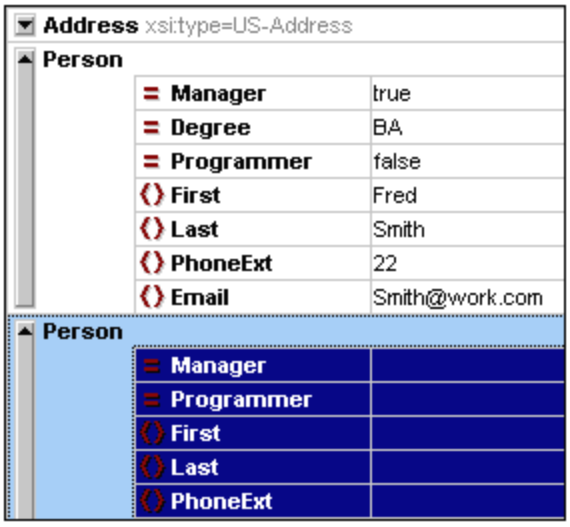
To display an element type as a table, you select any one of the element type occurrences and click the Display as Table icon in the toolbar (**XML | Table | Display as table**). This causes that element type to be displayed as a table. Descendant element types that have multiple occurrences are also displayed as tables. Table View is available in Enhanced Grid View, and can be used to edit any type of XML file (XML, XSD, XSL, etc.).

Advantages of Table View
Table View provides the following advantages:

- You can drag-and-drop column headers to reposition the columns relative to each other. This means that, in the actual XML document, the relative position of child elements or attributes is modified for all the element occurrences that correspond to the rows of the table.
- Tables can be sorted (in ascending or descending order) according to the contents of any column using **XML | Table | Ascending Sort** or **Descending Sort**.
- Additional rows (i.e., element occurrences) can be appended or inserted using **XML | Table | Insert Row**.
- You can copy-and-paste **structured data** to and from third party products
- The familiar intelligent editing feature is active in Table View also.

Displaying an element type as a Table
To display the `Person` element type as a table:

1. In Grid View, select either of the `Person` elements by clicking on or near the `Person` text.

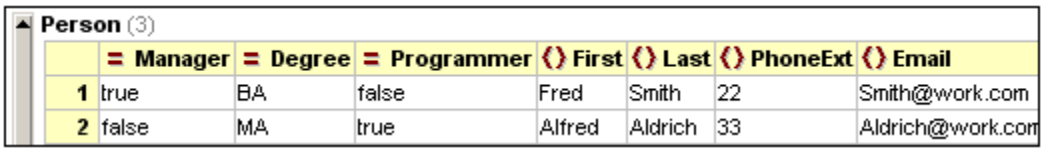


2. Select the menu option **XML | Table | Display as table**, or click the Display as Table icon. Both `Person` elements are combined into a single table. The element and attribute names are now the column headers, and the element occurrences are the rows of the table.

3. Select the menu option **View | Optimal widths**, or click the Optimal Widths icon, to optimize the column widths of the table.

Please note: Table View can be toggled off for individual element types in the document by selecting that table (click the element name in the table) and clicking the Display As Table icon. Note however that child elements which were displayed as tables will continue to be displayed as tables.

Entering content in Table View
To enter content for the second `Person` element, double-click in each of the table cells in the second row, and enter some data. Note, however, that `PhoneExt` must be an integer up to 99 in order for the file to be valid. The intelligent editing features are active also within cells of a table, so you can select options from dropdown lists where options are available (Boolean content and the enumerations for the `Degree` attribute).



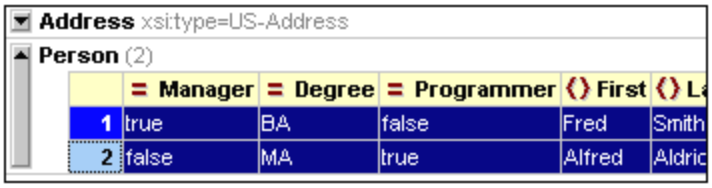
Please note: The Entry Helpers are active also for the elements and attributes displayed as a table. Double-clicking the `Person` entry in the Elements Entry Helper, for example, would add a new row to the table (i.e., a new occurrence of the `Person` element).

Copying XML data to and from third party products
You can copy spreadsheet-type data between third party products and XML documents in XMLSpy. This data can be used as XML data in XMLSpy and as data in the native format of the application copied to/from. In this section you will learn how to copy data to and from an Excel data sheet.

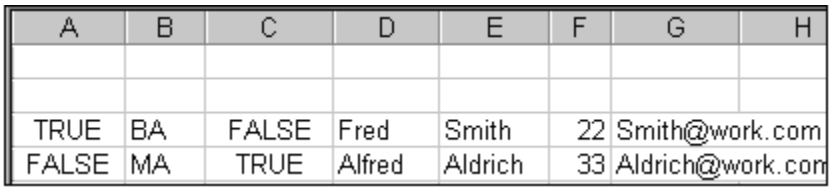
Do the following:

1. Click on the row label `1`, hold down the **Ctrl** key, and click on row label `2`. This selects both rows of the table.

 © 2012 Altova GmbH

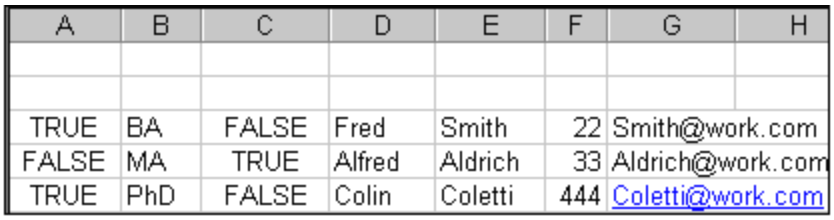


2. Select the menu option **Edit | Copy as Structured text**. This command copies elements to the clipboard as they appear on screen.
3. Switch to Excel and paste (**Ctrl+V**) the XML data in an Excel worksheet.



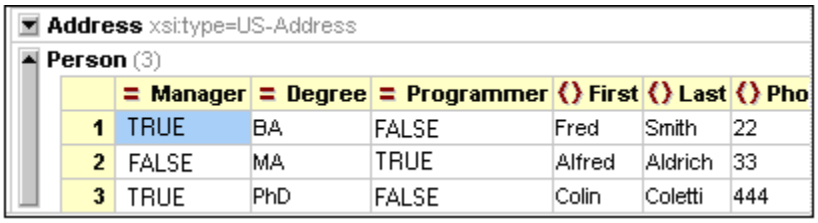
A	B	C	D	E	F	G	H
TRUE	BA	FALSE	Fred	Smith	22	Smith@work.com	
FALSE	MA	TRUE	Alfred	Aldrich	33	Aldrich@work.com	

4. Enter a new row of data in Excel. Make sure that you enter a three digit number for the `PhoneExt` element (say, `444`).



A	B	C	D	E	F	G	H
TRUE	BA	FALSE	Fred	Smith	22	Smith@work.com	
FALSE	MA	TRUE	Alfred	Aldrich	33	Aldrich@work.com	
TRUE	PhD	FALSE	Colin	Coletti	444	Coletti@work.com	

5. Mark the table data in Excel, and press **Ctrl+C** to copy the data to the clipboard.
6. Switch back to XMLSpy.
7. Click in the top left **data** cell of the table in XMLSpy, and select **Edit | Paste**.



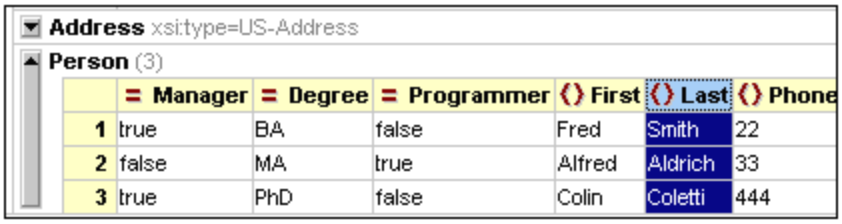
8. The updated table data is now visible in the table.
9. Change the uppercase boolean values `TRUE` and `FALSE` to lowercase `true` and `false`, respectively, using the menu option **Edit | Replace (Ctrl+H)**.

Sorting the table by the contents of a column

A table in Table View can be sorted in ascending or descending order by any of its columns. In this case, we want to sort the `Person` table by last names.

To sort a table by the contents of a column:

1. Select the `Last` column by clicking in its header.

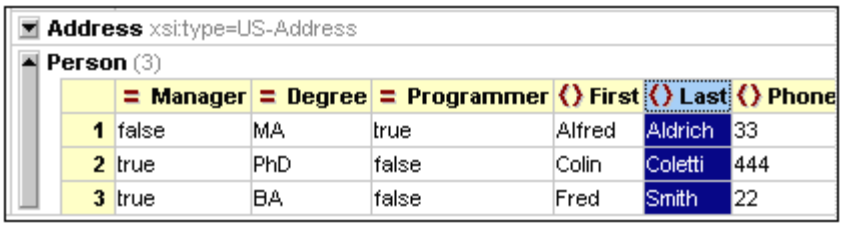


Address xsi:type=US-Address

Person (3)

	Manager	Degree	Programmer	First	Last	Phone
1	true	BA	false	Fred	Smith	22
2	false	MA	true	Alfred	Aldrich	33
3	true	PhD	false	Colin	Coletti	444

2. Select the menu option **XML | Table | Ascending sort** or click on the Ascending Sort icon . The column, and the **whole table** with it, are now sorted alphabetically. The column remains highlighted.



Address xsi:type=US-Address

Person (3)

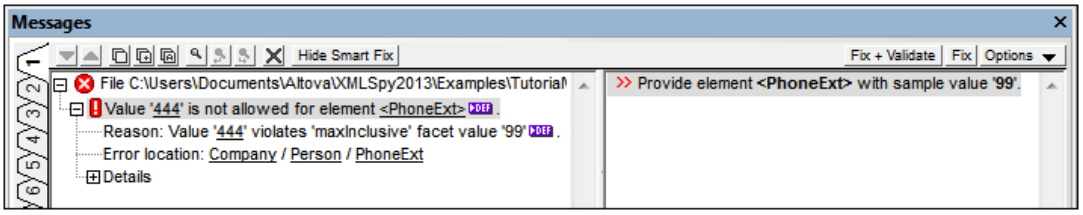
	Manager	Degree	Programmer	First	Last	Phone
1	false	MA	true	Alfred	Aldrich	33
2	true	PhD	false	Colin	Coletti	444
3	true	BA	false	Fred	Smith	22

The table is sorted not just in the display but also in the underlying XML document. That is, the order of the `Person` elements is changed so that they are now ordered alphabetically on the content of `Last`. (Click the Text tab if you wish to see the changes in Text View.)

3. Select the menu option **XML | Validate** or press **F8**. An error message appears indicating that the value `'444'` is not allowed for a `PhoneExt` element (*see screenshot*). The invalid `PhoneExt` element is highlighted .

   Expand "Details" to see that `PhoneExt` is not valid because it is not less than or equal to the maximum value of `99`.

   Please Note: You can click on the links in the error message to jump to the spot in the XML file where the error was found.



Since the value range we set for phone extension numbers does not cover this extension number, we have to modify the XML Schema so that this number is valid. You will do this in the next section.

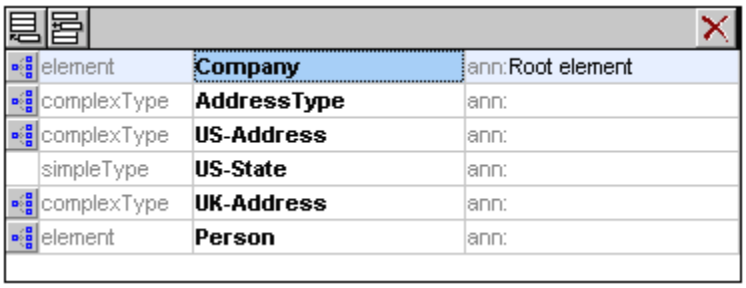
 © 2012 Altova GmbH

5.8 Modifying the Schema

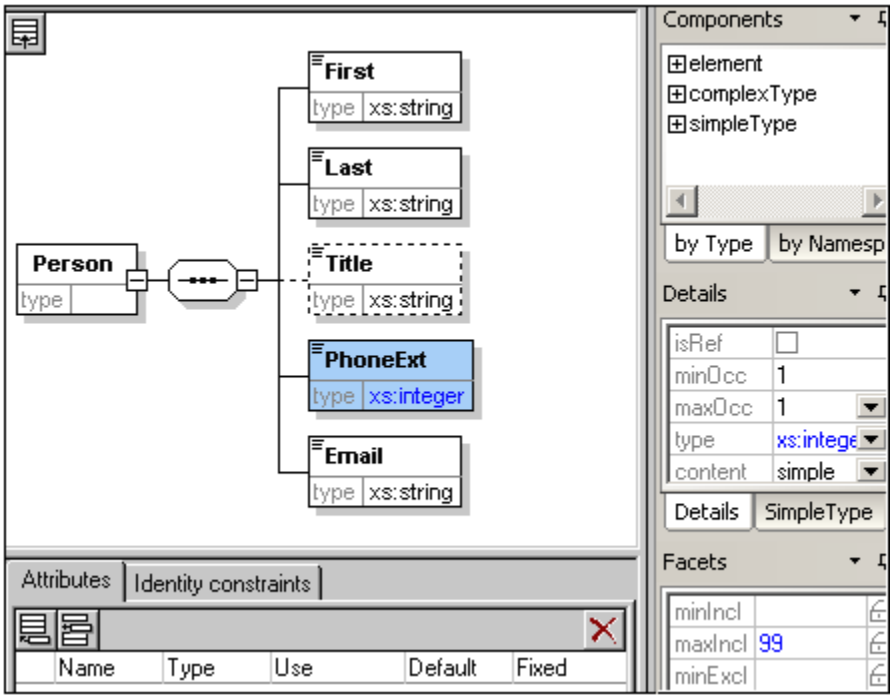
Since two-digit phone extension numbers do not cover all likely numbers, let's extend the range of valid values to cover three-digit numbers. We therefore need to modify the XML Schema. You can open and modify the XML Schema without having to close your XML document.

Do the following:

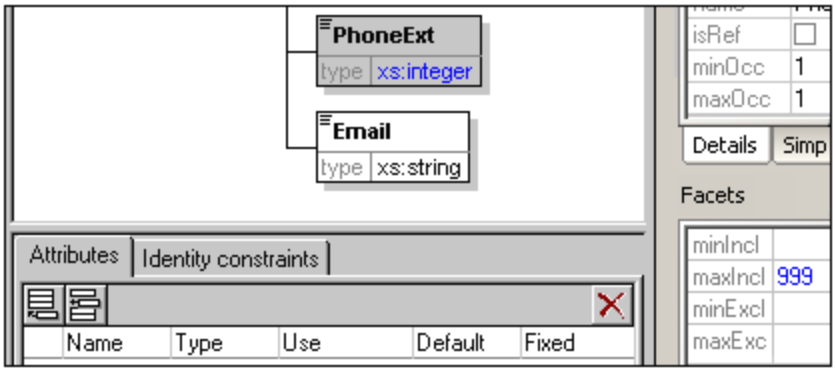
1. Select the menu option **DTD/Schema | Go to definition** or click the Go To Definition icon . The associated schema, in this case `AddressLast.xsd` is opened. Switch to Schema View (*screenshot below*) if you need to. (By default an XSD schema file will open in Schema View. The default view for every filetype, however, can be changed in the File Types tab of the Options dialog (**Tools | Options**).)



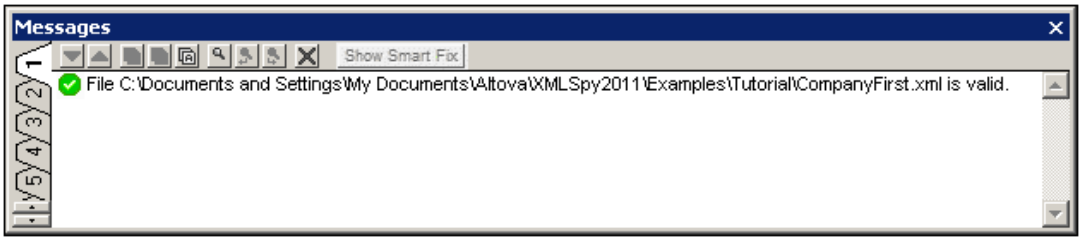
2. In Schema Overview, click the Display Diagram icon of the global `Person` element. In the Content Model View of the `Person` element, select the `PhoneExt` element. The facet data in the Facets tab is displayed.



3. In the Facets tab, double-click the `maxIncl` value field, change the value `99` to `999`, and confirm with **Enter**.



4. Save the schema document.
5. Press **Ctrl+Tab** to switch back to the XML document.
6. Click to revalidate the XML document.



A message that the file is valid appears in the Validation window. The XML document now conforms to the modified schema.

7. Select the menu option **File | Save As...** and save the file as `CompanyLast.xml`. (Remember to rename the original `CompanyLast.xml` file that is delivered with XMLSpy to something else, like `CompanyLast_orig.xml`).

Please note: `The CompanyLast.xml` file delivered with XMLSpy is in the in the `Tutorial` folder.

 © 2012 Altova GmbH

6 XSLT Transformations

Objective
To generate an HTML file from the XML file using an XSL stylesheet to transform the XML file. You should note that a "transformation" does not change the XML file into anything else; instead a new output file is generated. The word "transformation" is a convention.

Method
The method used to carry out the transformation is as follows:

- Assign a predefined XSL file, `Company.xsl`, to the XML document.
- Execute the transformation within the XMLSpy interface using one of the two built-in Altova XSLT engines. (*See note below.*)

Commands used in this section
The following XMLSpy commands are used in this section:

XSL/XQuery | Assign XSL, which assigns an XSL file to the active XML document.

XSL/XQuery | Go to XSL, opens the XSL file referenced by the active XML document.

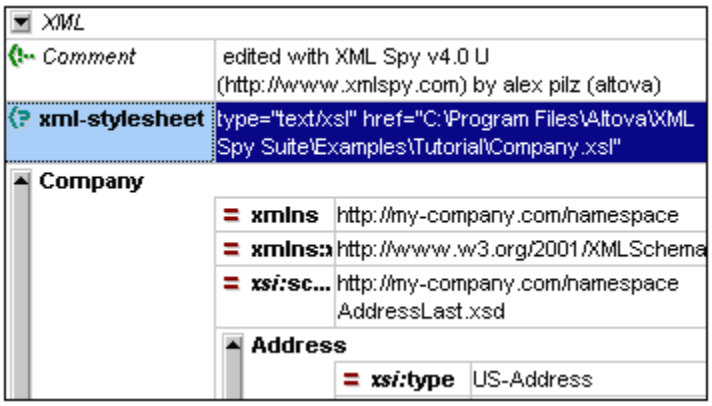
XSL/XQuery | XSL Transformation (F10), or the toolbar icon , transforms the active XML document using the XSL stylesheet assigned to the XML file. If an XSL file has not been assigned then you will be prompted for one when you select this command.

Please note: XMLSpy has two built-in XSLT engines, the Altova XSLT 1.0 Engine and Altova XSLT 2.0 Engine. The Altova XSLT 1.0 Engine is used to process XSLT 1.0 stylesheets. The Altova XSLT 2.0 Engine is used to process XSLT 2.0 stylesheets. The correct engine is automatically selected by XMLSpy on the basis of the version attribute in the `xsl:stylesheet` or `xsl:transform` element. In this tutorial transformation, we use XSLT 1.0 stylesheets. The Altova XSLT 1.0 Engine will automatically be selected for transformations with these stylesheets when the **XSL Transformation** command is invoked.

6.1 Assigning an XSLT File

To assign an XSLT file to the `CompanyLast.xml` file:

1. Click the `CompanyLast.xml` tab in the main window so that `CompanyLast.xml` becomes the active document, and switch to Text View.
2. Select the menu command **XSL/XQuery | Assign XSL**.
3. Click the **Browse** button, and select the `Company.xsl` file from the Tutorial folder. In the dialog, you can activate the option Make Path Relative to `CompanyLast.xml` if you wish to make the path to the XSL file (in the XML document) relative.4. Click **OK** to assign the XSL file to the XML document.
5. Switch to Grid View to see the assignment (*screenshot below*).



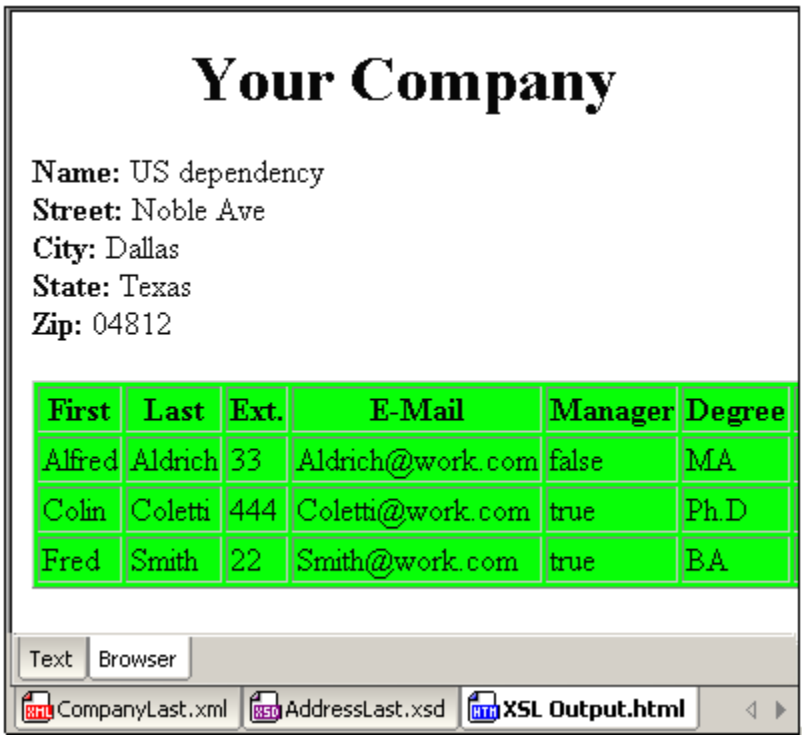
An `XML-stylesheet` processing instruction is inserted in the XML document that references the XSL file. If you activated the Make Path Relative to `CompanyLast.xml` check box, then the path is relative; otherwise absolute (as in the screenshot above).

 © 2012 Altova GmbH

6.2 Transforming the XML File

To transform the XML document using the XSL file you have assigned to it:

1. Ensure that the XML file is the active document.
2. Select the menu option **XSL/XQuery | XSL Transformation (F10)** or click the icon. This starts the transformation using the XSL stylesheet referenced in the XML document. (Since the `Company.xsl` file is an XSLT 1.0 document, the built-in Altova XSLT 1.0 Engine is automatically selected for the transformation.) The output document is displayed in Browser View; it has the name `XSL Output.html`. (If the HTML output file is not generated, ensure that, in the XSL tab of the Options dialog (**Tools | Options**), the default file extension of the output file has been set to `.html`.) The HTML document shows the Company data in one block down the left, and the Person data in tabular form below.



Please note: Should you only see a table header and no table data in the output file, make sure that you have defined the target namespace for your schema as detailed in Defining your own namespace at the beginning of the tutorial. The namespace must be **identical** in all three files (Schema, XML, and XSL).

6.3 Modifying the XSL File

You can change the output by modifying the XSL document. For example, let's change the background-color of the table in the HTML output from lime to yellow.

Do the following:

1. Click the `CompanyLast.xml` tab to make it the active document, and make sure you are in Grid View.
2. Select the menu option **XSL/XQuery | Go to XSL**.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xsi ="http://www.w3.org/2000/10/XMLSchema-instance"
xmlns:my="http://my-company.com/namespace">

<xsl:template match="/">
    <html>
        <head> <title>Your company</title></head>
            <body>
                <h1><center>Your Company</center></h1>
                <xsl:apply-templates select="//my:Address"/>
                    <table border="1" bgcolor="lime">
                        <thead align="center">
                            <td><strong>First</strong></td>
                            <td><strong>Last</strong></td>
                            <td><strong>Ext.</strong></td>
```

The command opens the `Company.xsl` file referenced in the XML document.

3. Find the line `<table border="1" bgcolor="lime">`, and change the entry `bgcolor="lime"` to `bgcolor="yellow"`.

```
<h1><center>Your Company</center></h1>
<xsl:apply-templates select="//my:Address"/>
    <table border="1" bgcolor="yellow">
        <thead align="center">
            <td><strong>First</strong></td>
            <td><strong>Last</strong></td>
```

4. Select the menu option **File | Save** to save the changes made to the XSL file.
5. Click the `CompanyLast.xml` tab to make the XML file active, and select **XSL/XQuery | XSL Transformation,** or press **F10**. A new `XSL Output.html` file appears in the XMLSpy GUI in Browser View. The background color of the table is yellow.

 © 2012 Altova GmbH

Your Company

Name: US dependency
Street: Noble Ave
City: Dallas
State: Texas
Zip: 04812

First	Last	Ext.	E-Mail	Manager	Degree
Alfred	Aldrich	33	Aldrich@work.com	false	MA
Colin	Coletti	444	Coletti@work.com	true	Ph.D
Fred	Smith	22	Smith@work.com	true	BA

6. Select the menu option **File | Save**, and save the document as `Company.html`.

7 Project Management

This section introduces you to the project management features of XMLSpy. After learning about the benefits of organizing your XML files into projects, you will organize the files you have just created into a simple project.

© 2012 Altova GmbH

7.1 Benefits of Projects

The benefits of organizing your XML files into projects are listed below.

- Files and URLs can be grouped into folders by common extension or any other criteria.
- Batch processing can be applied to specific folders or the project as a whole.
- A DTD or XML Schema can be assigned to specific folders, allowing validation of the files in that folder.
- XSLT files can be assigned to specific folders, allowing transformations of the XML files in that folder using the assigned XSLT.
- The destination folders of XSL transformation files can be specified for the folder as a whole.

All the above project settings can be defined using the menu option **Project | Project Properties...**. In the next section, you will create a project using the Project menu.

Additionally, the following advanced project features are available:

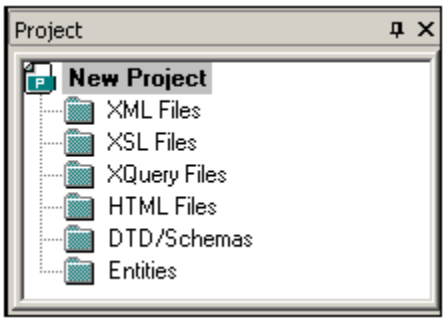
- XML files can be placed under source control using the menu option **Project | Source control | Add to source control...**. (Please see the Source Control section in the online help for more information.)
- Personal, network and web folders can be added to projects, allowing batch validation.

7.2 Building a Project

Having come to this point in the tutorial, you will have a number of tutorial-related files open in the Main Window. You can group these files into a tutorial project. First you create a new project and then you add the tutorial files into the appropriate sub-folders of the project.

Creating a basic project
To create a new project:

1. Select the menu option **Project | New Project**. A new project folder called `New Project` is created in the Project Window. The new project contains empty folders for typical categories of XML files in a project (*screenshot below*).



2. Click the `CompanyLast.xml` tab to make the `CompanyLast.xml` file the active file in the Main Window.
3. Select the menu option **Project | Add active and related files to project.** Two files are added to the project: `CompanyLast.xml` and `AddressLast.xsd`. Note that files referenced with Processing instructions (such as XSLT files) do not qualify as related files.
4. Select the menu option **Project | Save Project** and save the project under the name `Tutorial`.

Adding files to the project
You can add other files to the project as well. Do this as follows:

1. Click on any open XML file (with the `.xml` file extension) other than `CompanyLast.xml` to make that XML file the active file. (If no other XML file is open, open one or create a new XML file.)
2. Select the menu option **Project | Add active file to project**. The XML file is added to the XML Files folder on the basis of its `.xml` file type.
3. In the same way, add an HTML file and XSD file (say, the `Company.html` and `AddressFirst.xsd` files) to the project. These files will be added to the HTML Files folder and DTD/Schemas folder, respectively.
4. Save the project, either by selecting the menu option **Project | Save Project** or by selecting any file or folder in the Project Window and clicking the Save icon in the toolbar (or **File | Save**).

Please note: Alternatively, you can right-click a project folder and select **Add Active File** to add the active file to that specific folder.

Other useful commands
Here are some other commonly used project commands:

- To add a new folder to a project, select **Project | Add Project folder to Project**, and insert the name of the project folder.

 © 2012 Altova GmbH

- To delete a folder from a project, right-click the folder and select **Delete** from the context menu.
  To delete a file from a project, select the file and press the **Delete** key.

8 That's It

If you have come this far congratulations, and thank you!

We hope that this tutorial has been helpful in introducing you to the basics of XMLSpy. If you need more information please use the context-sensitive online help system, or print out the PDF version of the tutorial, which is available as `tutorial.pdf` in your XMLSpy application folder.

© 2012 Altova GmbH

Index

A

Attribute,
in schema definitions, 29
toggle in Content model view, 29

C

Complex type,
extending definition, 19
in schema definitions, 19
Component definition,
reusing, 19
Compositor,
for sequences, 8
Content Model,
creating a basic model, 8
toggle attributes, 29
Content Model View, 4

D

Database/Table View,
how to use, 55
Details Entry Helper, 8
Documentation,
for schema, 35

E

Element,
making optional, 15
restricting content, 15
element type,
specifying in XML document, 42
Enhanced Grid View,
see Grid View, 44
Entry Helper, 2
Details, 8
in Grid View, 53
Enumeration,
defining for attributes, 29
Example files,
tutorial, 1

G

Global element,
using in XML Schema, 27
Grid View,
and Table View, 55
appending elements and attributes, 53
data-entry in, 44
using Entry Helpers, 53

I

Identity constraint,
toggle in Content model view, 29
Info,
window, 2

M

Main window, 2

N

Namespace,
in schemas, 7
Navigation,
shortcuts in schema design, 33
New XML document,
creating, 40

© 2012 Altova GmbH

O

Occurrences,
number of, 8
Optional element,
making, 15
Overview, 2

P

Project,
window, 2
Project management in XMLSpy, 66
Projects in XMLSpy,
benefits of, 67
how to create, 68

S

Schema,
documentation, 35
see XML Schema, 3
Schema Overview, 4
Schema View,
configuring the view, 13
Sequence compositor,
using, 8
Simple type,
in schema definitions, 19

T

Table View,
how to use, 55
Template folder, 1
Text View,
editing in, 45
Tutorial,
example files, 1
goals, 1
type,
extension in XML document, 42

V

Validating,
XML documents, 49

W

Well-formedness check,
for XML document, 49
Windows,
overview, 2

X

XML document,
creating new, 40
editing in Text View, 45
XML document creation,
tutorial, 39
XML documents,
checking validity of, 49
XML Schema,
adding components, 8
adding elements with, 12
configuring the view, 13
creating a basic schema, 3
creating a new file, 4
defining namespaces in, 7
modifying while editing XML document, 59
navigation in design view, 33
tutorial, 3
XML schema definitions,
advanced, 18
xsi:type,
usage, 42
XSL transformation,
see XSLT, 61
XSLT,
modifying in XMLSpy, 64
XSLT transformation,

© 2012 Altova GmbH

XSLT transformation,
assigning XSLT file, 62
in XMLSpy, 63
tutorial, 61

© 2012 Altova GmbH