

# Modeling Data

## Exercises



## Contents

<b>1. Module .....</b>	<b>3</b>
1.1 Objectives.....	3
1.2 Overview.....	3
<b>2. About Data Modeling .....</b>	<b>4</b>
2.1 Introduction.....	4
2.1.1 Business Objects.....	4
2.1.2 XML-Objects.....	4
2.1.3 XML Object Definitions (XSD) .....	5
2.1.4 Transformations.....	5
2.2 References .....	6
<b>3. Exercises .....</b>	<b>7</b>
3.1 Prerequisites.....	7
3.2 Configuring Design and Deployment structure .....	7
3.2.1 Creating Design Folder Structure .....	7
3.2.2 Creating Deployment Structure .....	8
3.3 XML Schema Definitions .....	8
3.3.1 Creating an XML Schema .....	8
3.3.2 Exploring Existent XML Schemas.....	13
3.4 Data Transformations .....	14
3.4.1 Creating Data Transformations.....	14
3.4.2 Custom Data Transformation Web Service .....	18

3.4.3 Generic Data Transformation Web Service .....	21
3.5 Using Transformations in Business Process .....	24
3.5.1 Prerequisites.....	24
3.5.2 Analyzing Situation .....	24
3.5.3 Creating the Business Process.....	24
3.6 Transformations in User Interfaces.....	29
3.7 Make Changes Available to SCM.....	30
<b>4. Learning Report .....</b>	<b>31</b>

# 1. Module

---

## 1.1 Objectives

After completing this course module, you will be able to:

- Understand the concept of business objects
- Understand the concept of business object abstraction
- Create canonical data objects
- Create data transformation models
- Use data transformation in a business process

## 1.2 Overview

Typically, when running multiple applications within a company, each of the applications uses their own type of data. For example, the definition of a customer (and the corresponding customer data being stored) in a CRM application will be different than the customer definition of the ERP application. By modeling data, you can create an abstraction layer between the business and the application by applying a generic business object customer, and additionally using data transformations to convert the generic object into the specific customer object of the application and/or vice versa.

## 2. About Data Modeling

---

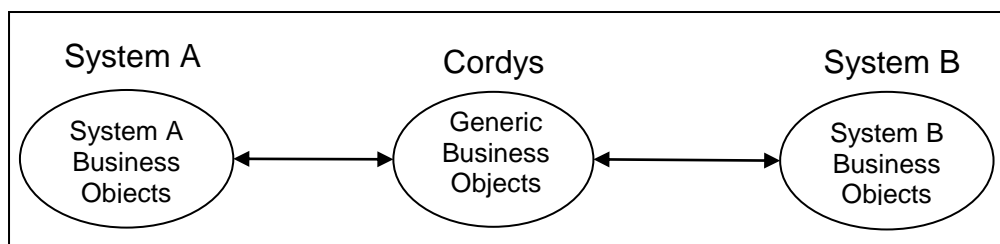
### 2.1 Introduction

#### 2.1.1 Business Objects

Business Objects are the core of any application. Generally, an application has a set of business objects interacting with each other within business processes or user interfaces. Typically, these business objects are stored somewhere for reference and future use.

Examples of applicable business objects are: the (sales) quote, the sales order, the customer, the supplier, the product, the inventory, the employee, and the expense report.

Any implementation has its own private definition of business objects, even for the same type of application. For example the customer definition in ERP applications like SAP, PeopleSoft, BAAN, Axapta, all have their own definition. The Cordys platform offers different techniques to overcome these differences and to facilitate collaboration between multiple applications, systems and even organizations.



A generic format or canonical data model is needed to abstract the backend application from the process implementation. This way the processes, user interfaces, etc., are not directly connected to the backends and do need to be changed when you switch to other applications.

#### 2.1.2 XML-Objects

XML stands for Extensible **M**arkup **L**anguage. In Cordys, business objects are defined as *XML objects* or *XML documents*.

In XML, business objects can be represented as follows:

```
<cars>
  <car>
    <brand>Mercedes</brand>
    <type>Roadster</type>
    <color colortype="P">red</color>
  </car>
  <car>
    <brand>Mercedes</brand>
    <type>Roadster</type>
    <color colortype="M">black</color>
  </car>
</cars>
```

### 2.1.3 XML Object Definitions (XSD)

XSD stands for **XML Schema Definition**. It is also referred to as XML Schema. The XSD contains the implementation details for an XML object, like data type, restrictions. You need the XSD along with the XML object to understand the data.

In XSD the business object cars can be represented as follows:

```
<element name="cars">
  <complexType>
    <sequence>
      <element name="car">
        <complexType>
          <sequence>
            <element name="brand">
              <simpleType>
                <restriction base="string">
                  <totalDigits value="20"/>
                </restriction>
              </simpleType>
            </element>
            <element name="type"/>
            <element name="color">
              <complexType mixed="true">
                <attribute name="colortype"/>
              </complexType>
            </element>
          </sequence>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>
```

### 2.1.4 Transformations

Transformations are used to convert one XML object into another XML object. The transformation model describes how the XML structure needs to be transformed using XSLT functions.

## 2.2 References

More information about this subject is available:

- Cordys Online Documentation  
Working with Business Models → Modeling Data
- <http://community.cordys.com>
- <http://www.w3schools.com>

## 3. Exercises

---

### 3.1 Prerequisites

Before you start with this module please note the following prerequisites, the exercises are written based on their successful completion.

**You must have completed the following modules**

- Application Management
- Web Service and the SOA Grid
- Developing Web Services
- (Developing Process for some exercise)

**You must have ONLY the following roles assigned to yourself**

- Administrator
- Developer
- Cordys Fundamentals Trainee

### 3.2 Configuring Design and Deployment structure

In this exercise you will setup the folder structure for the design time and deployment time components for modeling data.

For detailed information see the module Application Management.

#### 3.2.1 Creating Design Folder Structure

Here you will setup the design folder structure for modeling data according to the standard for this training (see Application Management).

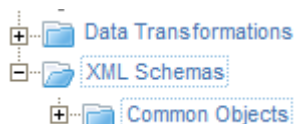
1. Open the *Workspace Documents*.
2. Open the *Fundamentals training* workspace.

#### Data Transformations

1. Right click the *My Application Project* and select *New → Folder*.
2. Enter the name: **Data Transformations**.

#### XML Schema

1. Right click the *My Application Project* and select *New → Folder*.
2. Enter the name: **XML Schemas**.
3. Add a subfolder named **Common Objects**.



In the XML Schema folder, you will additionally create subfolders for every backend. This way you can have the same schema name for different backends that you support and use with your data transformations. The common Objects are objects not created for a specific backend but are treated as the company's standard structure.

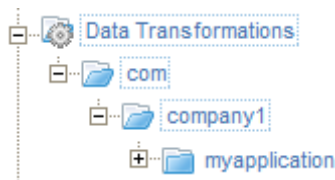
### 3.2.2 Creating Deployment Structure

Here you will create the deployment structure.

#### Data Transformations

Data transformations are at runtime identified by their path and model name.

1. Right click *Data Transformations* and select *Set Start Point of Qualified name*.
2. Add a subfolder **com**.
3. Add a subfolder **companyX**.
4. Add a subfolder **myapplication**.



#### XML Schema

XML Schemas are design time only objects that are used in other documents, so there is no deployment configuration required as they are not published separately. For example, when you use an XML Schema in a web service operation, the XSD will be published along with the web service and the deployment structure of the web service is applied.

## 3.3 XML Schema Definitions

In this exercise you will create an XML schema to reflect the business object Employee structure. This is the company employee structure (canonical data) for the employee and is not related to any application. XML schemas can be used for data transformations, business processes, case models decision cases, etc.

### 3.3.1 Creating an XML Schema

1. If closed, open the *Workspace Documents*.
2. Open the *Fundamentals training* workspace.
3. Navigate to *XML Schemas* → *Common Objects*.
4. Right click the *Common Objects* folder and create a new document of type XML

Schema (  **XML Schema**  
Create XML schema ).



**5.** Provide the following values:

Field	Value
Name	Employee.xsd
Description	Employee common object
Paste Schema OR Instance here	

Employee.xsd - XML Schema\*

Name  
Employee.xsd

Description  
Employee common object

Paste Schema OR Instance here.

< Previous   Next >   Finish   Cancel

**NOTE**

When you have the employee structure already available, you can immediately past it here. For now, you will study the creation of the schema step by step.

**6.** Click **Finish**.

## 7. Change the following values:

Field	Value
Target Namespace URI	http://schemas.companyX.com/myapplication/employee
Namespace prefix	emp

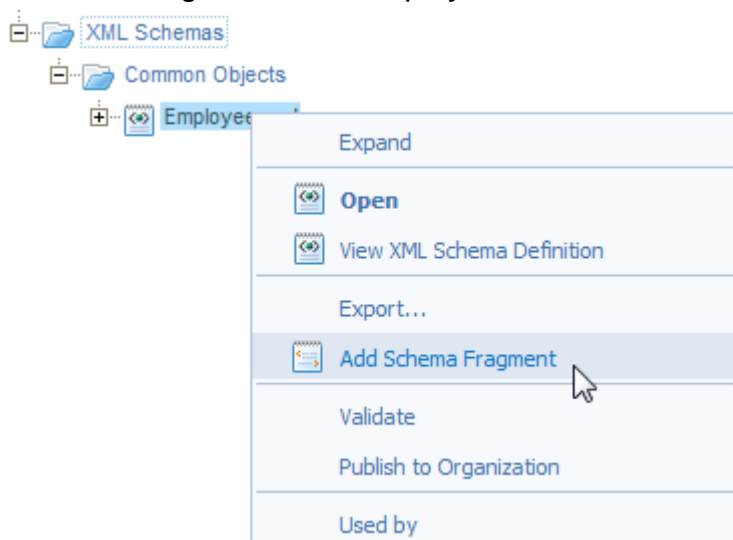
The screenshot shows the 'Employee.xsd - XML Schema' editor. The 'Workspace' pane on the left contains 'Default Case Management Event Types' and 'My Application Project'. The main editor area shows the following fields:

- Name: Employee.xsd
- Description: Employee common object
- Target Namespace URI: http://schemas.company1.com/myapplication/employee
- Namespace Prefix inside this schema: emp

Below these fields is a text area for 'The base XML Schema node' containing the following code:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:emp="http://schemas.company1.com/myapplication/employee"
targetNamespace="http://schemas.company1.com/myapplication/employee"
elementFormDefault="qualified" />
```

At the bottom, there is a table with two columns: 'Prefix' and 'XSD Reference'.

8. Save your *Employee.xsd* XML schema and close the screen.9. In the tree, right click the *Employee.xsd* and select *Add Schema Fragment*.

When looking at the WSDL of a web service, you will notice that it contains a schema for the input and for the output message. An XSD can also contain multiple schemas. For example, when defining a customer schema, it can contain multiple different addresses, e.g. postal address, visiting address, business address, etc. Whereas, the relevant address fields will be the same for these addresses. In addition to the customer schema, you will have an address schema as well. And from the customer schema, you will add a visit and a postal address while referencing the common address schema.

10. Provide the following values:

Field	Value
Name	Employee
Description	Employee schema
Open Document	Checked

11. Click Finish.

The Schema Fragment editor opens, with the root element Employee added.

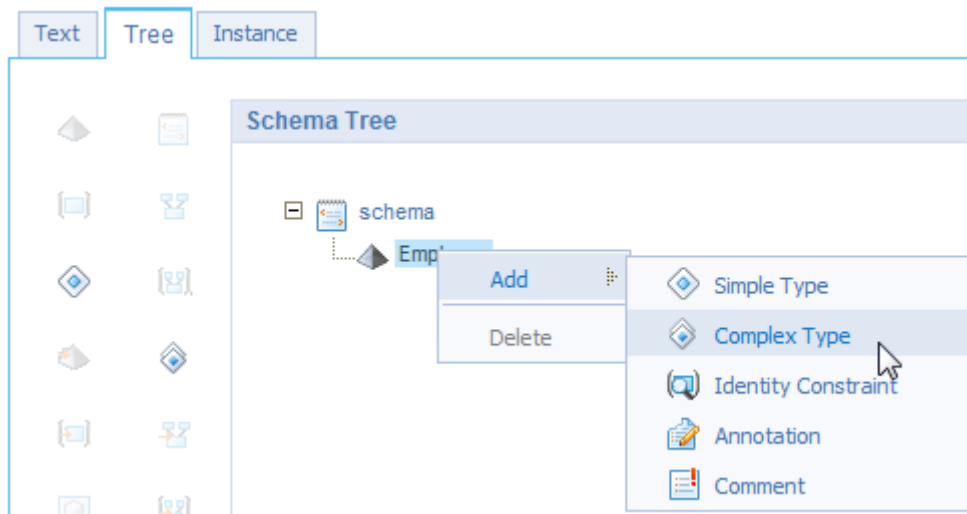
You create/view the schema fragments via one of the available tabs:

- Text**      You provide (copy or type) XSD
- Tree**      Graphically create the fragment structure
- Instance**    Provide an XML schema object (Read only)

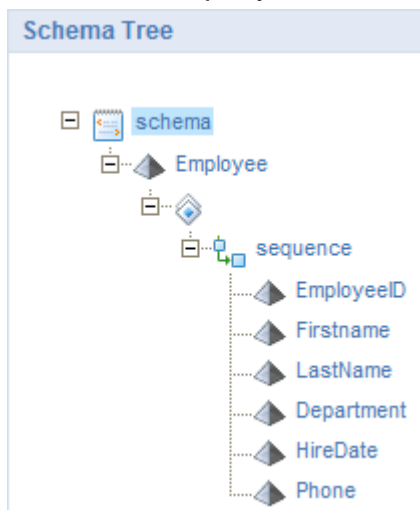
You will use the *Tree* tab to graphically create the relevant structure.

You can add components to the schema tree using the following two methods:

- Select a component in the tree and click on the component in the toolbox on the left
- Right click the element, e.g. *Employee* in the tree and select Add → .....



12. In the next steps, you will model the following fragment structure:



13. Add to the *Employee* (Employee) element a Complex Type (Complex Type icon).
14. Add a *Model Group* (Model Group icon) to the complex type.
15. Change the *Modifier* to *sequence* (properties at the right)
16. Add the following *Elements* (Element icon) to the Model Group: (**names are case sensitive**)

Name	Type
EmployeeID	integer
FirstName	string
LastName	string
Department	string
HireDate	date
Phone	string

17. Save the XML schema fragment.

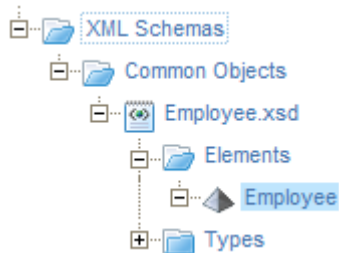
Why is it important to fill in the properties (data type, max occur etc) as accurate as possible?

---



---

18. Look at the *Text* tab for the actual XSD you have graphically created.
19. Look at the *Instance* tab for an example of an XML object/instance.
20. Close the XML Schema Fragment editor.



21. Right click the *Employee.xsd*, select *View XML Schema Definition* to verify the entered schema definition fragment.

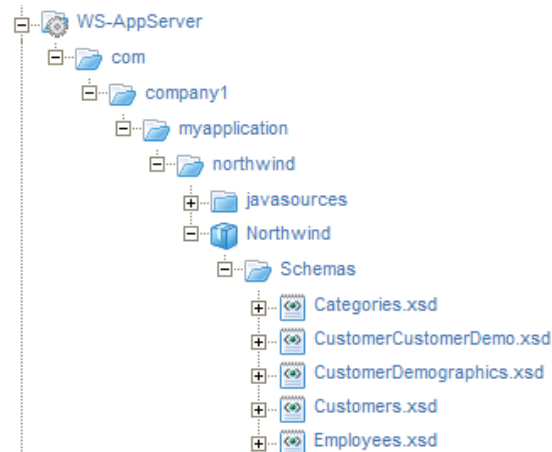
When you (re)open the *Employee* schema fragment, note that:

- The name of the complex type will be empty, whereas while creating the fragment, it displayed complex Type.
- The model group icon (M), is changed to the sequence icon

### 3.3.2 Exploring Existing XML Schemas

When you generate web services, XML schemas are automatically created. In this part you will explore the *Employee.xsd* schema created when you generated the web services for the Northwind backend in module Developing Web Services. This is the second XML schema that you will use for the data transformation later.

1. If closed, open the *Workspace Documents*.
2. Navigate to *WS-AppServer* → *com* → *companyX* → *myapplication* → *northwind* → *Northwind* (M) → *Schemas*.



3. Right click *Employees.xsd* and select *View XML Schema Definition*.

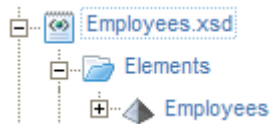
What is the difference with the employee schema you created?

---



---

4. Close the *XML Schema Definition Viewer*.
5. In the tree expand the *Employees.xsd*.
6. Open the *Employees* fragment:



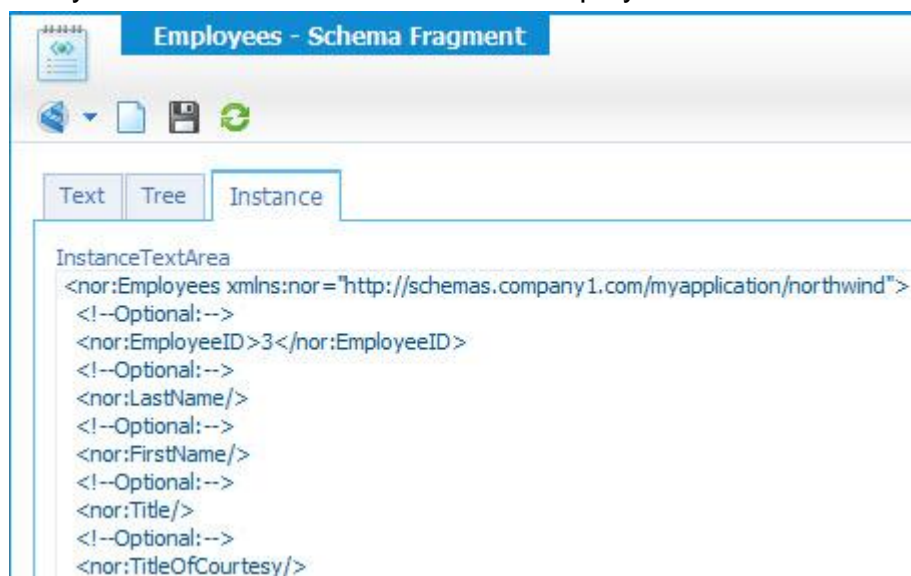
What is the type for the element *Employees*?

---



---

7. Go to the *Instance* tab.
8. Verify that the linked model fields are displayed.



9. Close the schema fragment editor.

## 3.4 Data Transformations

In this exercise you will create data transformations to convert the common employee structure to a backend employee's structure. Additionally, you will generate a web service executing the defined transformations and use this as a service in a business process.

### 3.4.1 Creating Data Transformations

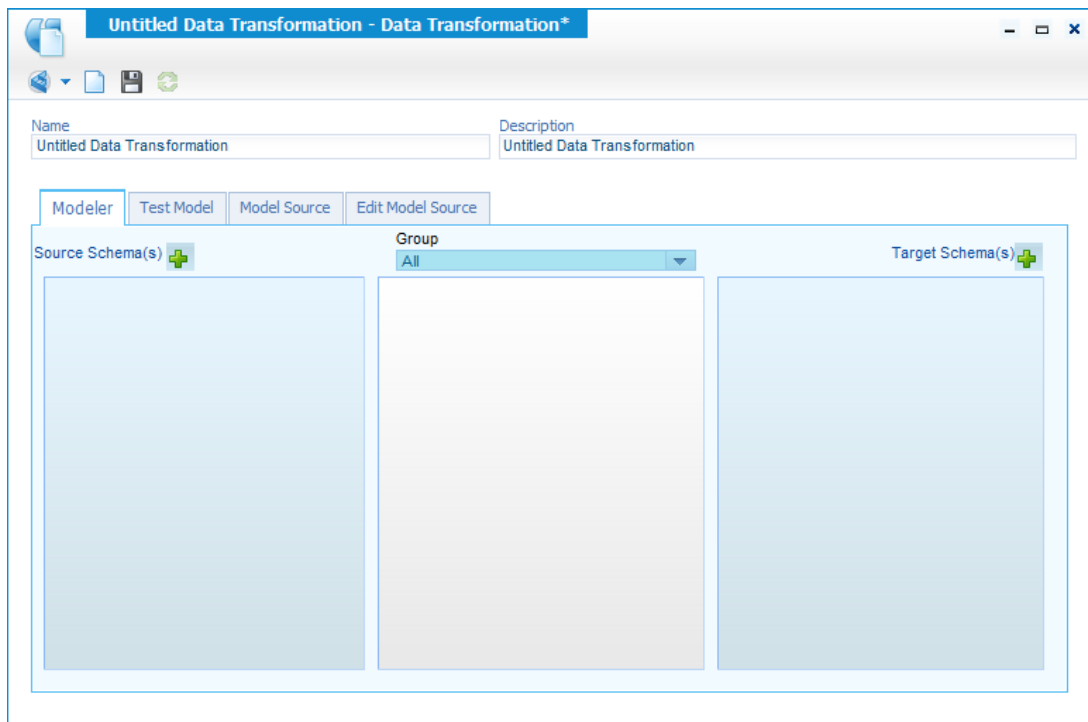
1. If closed, open the *Workspace Documents*.
2. Navigate to *Data Transformations* → *com* → *companyX* → *myapplication*.

**Data Transformation**

Define and test transformations between source and target data model

3. Add a new document of type *Data Transformation* ( ).

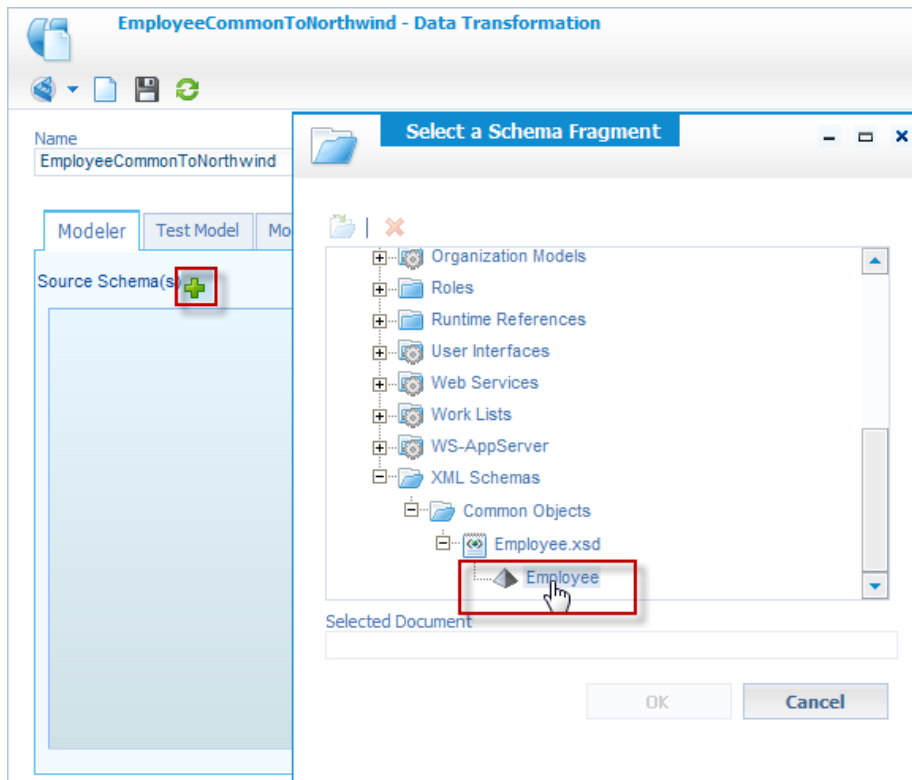
The Data Transformation editor is started.



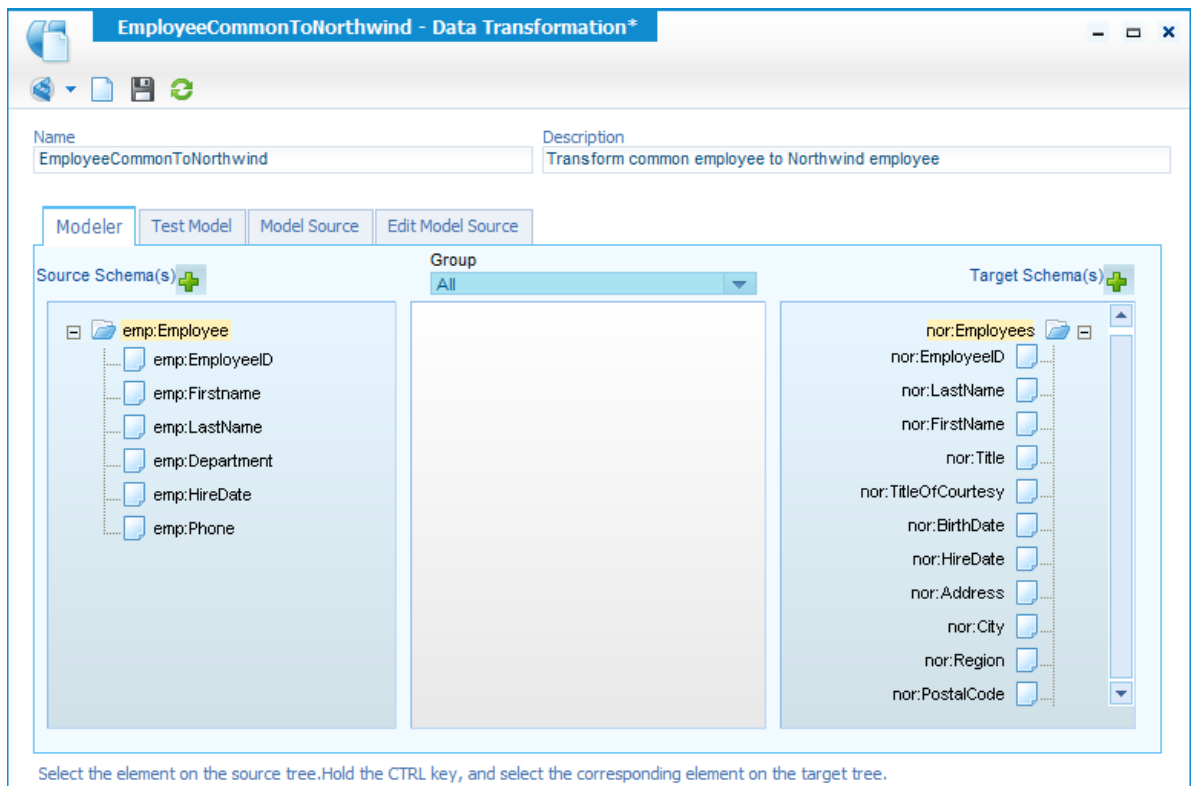
4. Save the data transformation model with the following details.

Field	Value
Name	EmployeeCommonToNorthwind
Description	Transform common employee to Northwind employee
Location	Prefilled with: My Application Project/Data Transformations/com/companyX/myapplication

5. Click the **Insert Source Schema** (+) icon to add the *Employee* schema fragment you created:



6. Add the *Target Schema* in the same way by selecting the WS-AppServer *Employees* schema fragment:

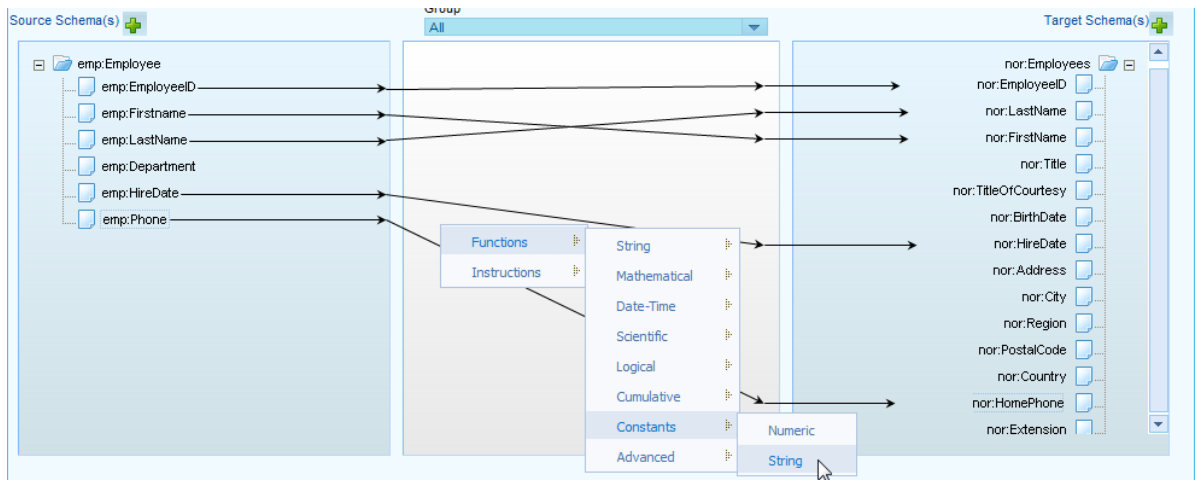


7. Connect all the related elements by clicking the source element and then CTRL + click the target element.

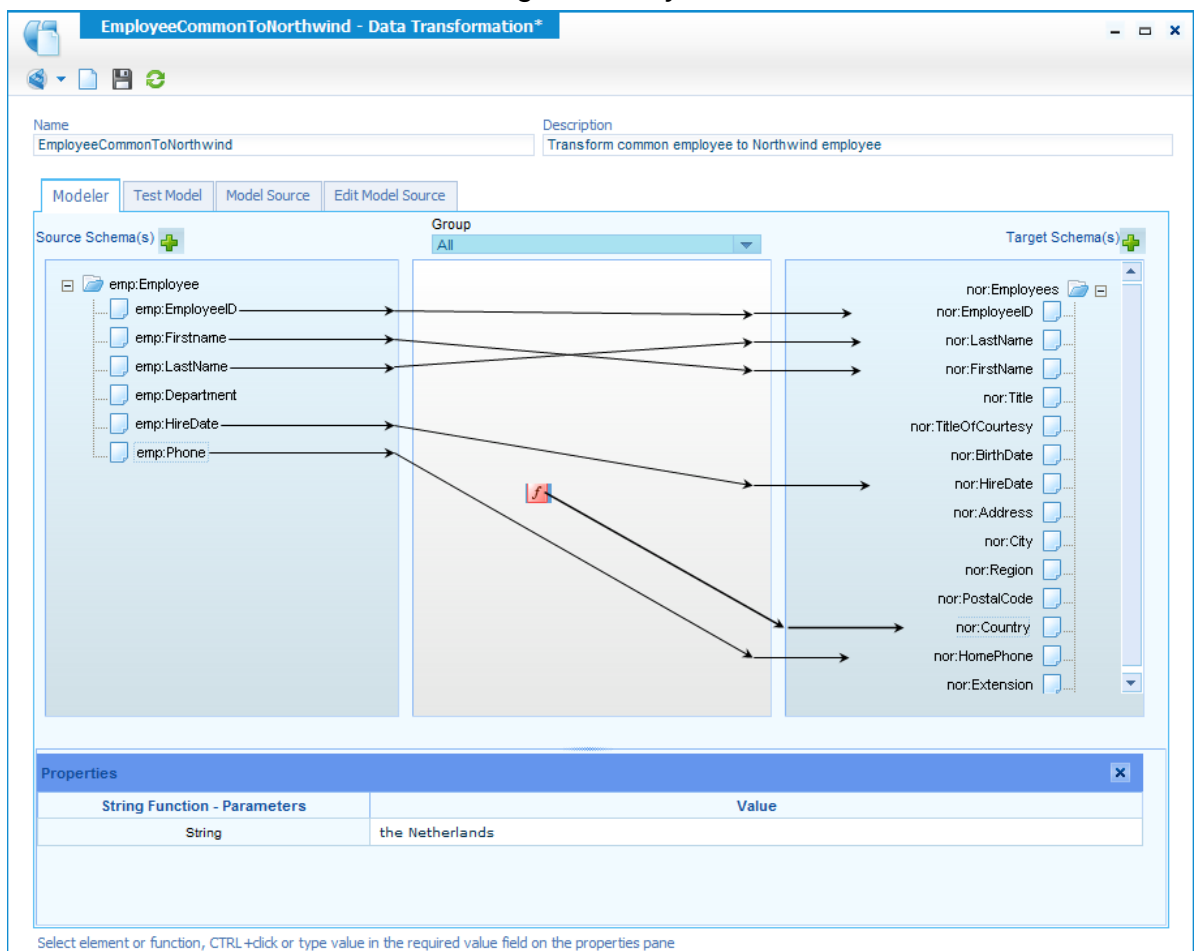


The Source object does not contain the *Country* field, you will use an (XSLT) function to provide a fixed value.

8. Right click in the box in the middle and select **Functions** → **Constants** → **String**.



9. In the Properties box, opened in the lower part of the editor, type the name of your country e.g. **the Netherlands, Germany, USA** (Max 15 characters).
10. Connect the function block to the target *Country* element:



11. Save your data transformation model.

### Test the design model

1. Go to the *Test Model* tab.

**NOTE**

The month names are filled in by default so you can immediately perform a transformation test.

2. Provide values in the *Source XML*, e.g. you can enter your personal details.
3. Click **Transform**.
4. Check whether the data transformations are performed successfully.
5. Close the editor.
6. Publish your data transformation model.

### 3.4.2 Custom Data Transformation Web Service

In this part you will create a custom web service operation.

After you created a data transformation model, you can use it in two ways:

- You can use the generic web service, which can use any transformation model. Hence is flexible to use but does not have a specific input and output message.
- You generate a custom web service operation for one data transformation model. Hence it contains proper input and output message but cannot be used for another data transformation model.

#### Create the Web Service

1. In the *Workspace Document* toolbar, click **New**.
2. Select document type *Web Service*.

3. Provide the following values:

Field	Value
Source	Data Transformation
Name	DataTransformations
Description	Data Transformations
Namespace	http://schemas.companyX.com/myapplication/transformations
Location	My Application Project/Web Services

**DataTransformations - Web Service\***

Select the source  
Data Transformation

Name  
DataTransformations

Description  
Data Transformations

Namespace  
http://schemas.company1.com/myapplication/transformations

Location  
Web Services

< Previous   Next >   Finish   Cancel

4. Click **Next**.

5. Provide the following values to define the web service:

Field	Value
Data Transformation	EmployeeCommonToNorthwind
Web Service Interface Name	Common
Operation Name	EmployeeToNorthwind

**DataTransformations - Web Service\***

Data Transformation  
EmployeeCommonToNorthwind

Select Web Service Operation(s)

Web Service Interface Name  
Common

Web Service Operation Name  
EmployeeToNorthwind

< Previous   Next >   Finish   Cancel

6. Click **Finish**.

The logic behind the created web service structure is:

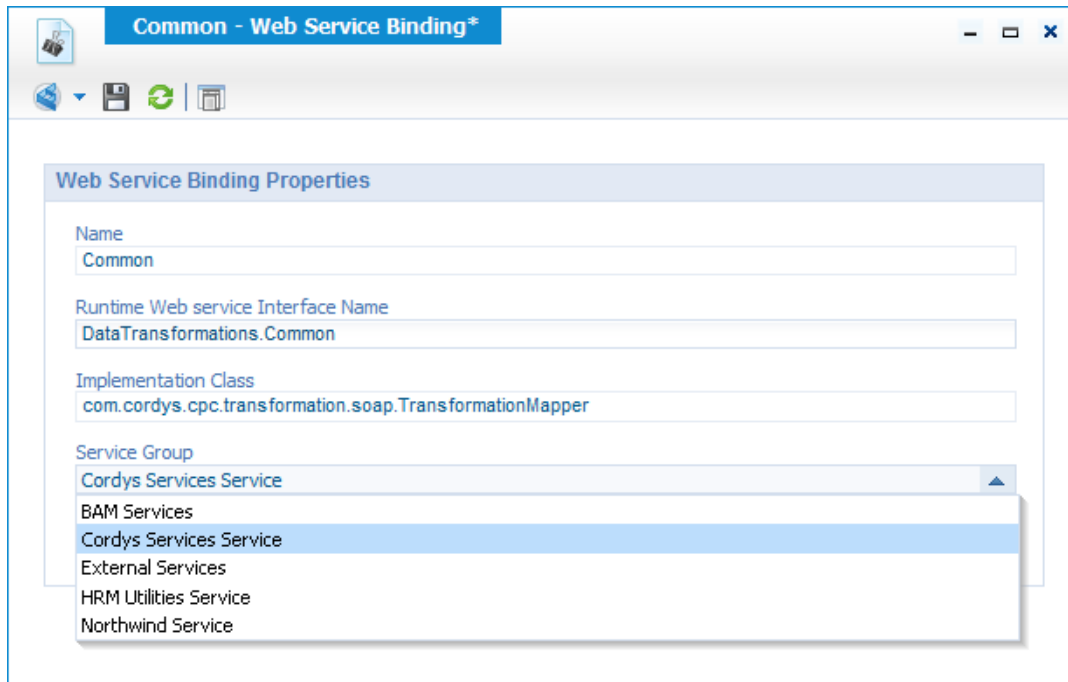
- You combine all data transformation services for your application/project together in one Web Services definition called Data Transformations.
- Per backend type, you will use a separate web service interface with the associated name of the backend type or common to group them together, e.g. Common, Northwind.
- The name of the web service operation is made short (no repetition of the interface common name) to quickly identify the operation and identify them by name.

## Using the Web Service

1. Navigate to *Web Services* → *DataTransformations*.



2. Open the Web Service Interface *Common*.
3. Select the *Service Group*: *Cordys Services Service*.

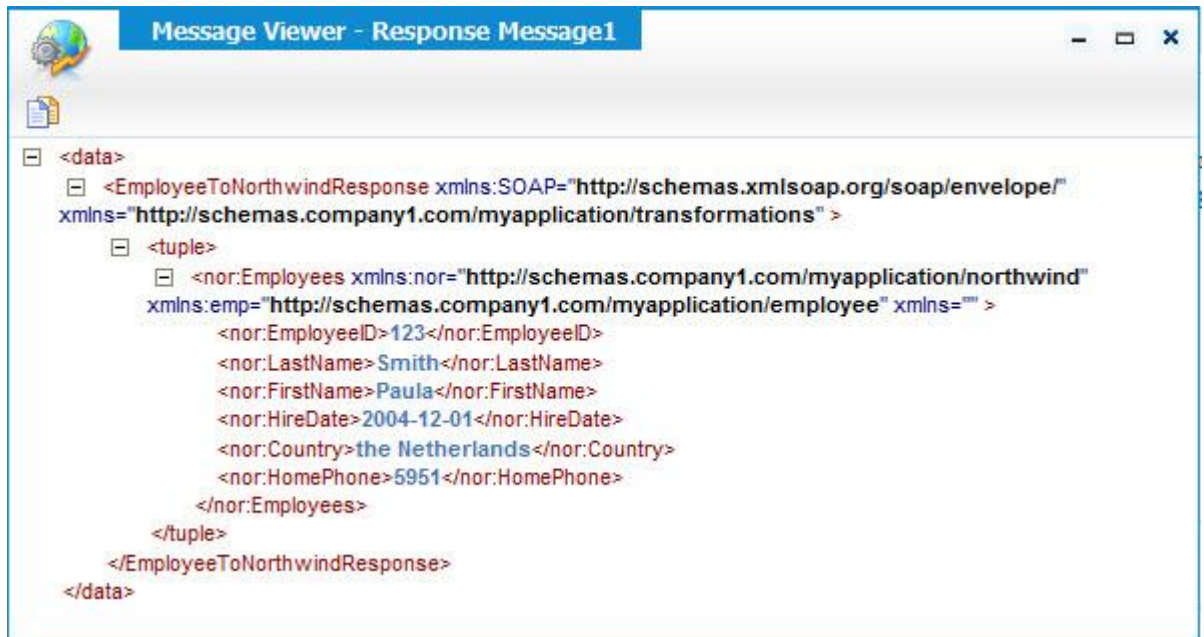


## NOTE

On the training server the Cordys Services combines multiple application connectors like the Data Transformation connector, which is required in your own organization for executing custom web services.

4. **Save** the Web Service Interface.
5. *Publish* the Web Service.
6. *Test* the *EmployeeToNorthwind* web service operation, using your own details.

7. The response should look similar to this:

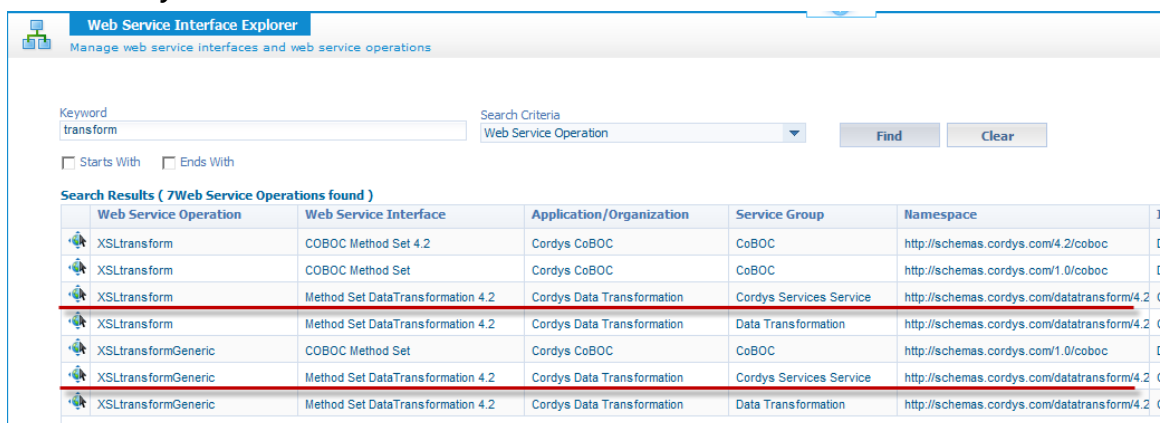


8. Copy the complete **input** message structure for the data transformations operation:  
`<Employee xmlns="http://.....">` to `</Employee>`  
 from the soap request and place it in a text editor for later use.

### 3.4.3 Generic Data Transformation Web Service

In this exercise you will use a generic transformation service which can be used to execute any data transformation model.

1. Open the *Web Service Interface Explorer* (  )
2. Provide **Keyword transform** and click **Find**.



There are two generic services to perform a data transformation:

- |                            |  |
|----------------------------|--|
| <b>XSLtransform</b>        | Requires an XML object and the name of a data transformation model to transform the given object.  |
| <b>XSLtransformGeneric</b> | Requires an XML object and the actual XSLT to transform the object. In this case, you do not need to create the data transformation model in Cordys as you will directly use (an external) XSLT. |

3. Right click the operation *XSLtransform* of application interface *Cordys Data Transformation* attached to Service Group *Cordys Services Service* and select *Test*.
4. In the **Next Steps** you will model the following Soap Request:

SOAP Request

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <XSLtransform xmlns="http://schemas.cordys.com/datatransform/4.2">
      <SourceDoc>
        <ns0:Employee xmlns:ns0="http://schemas.company1.com/myapplication/employee">
          <ns0:EmployeeID>123</ns0:EmployeeID>
          <ns0:FirstName>Paula</ns0:FirstName>
          <ns0:LastName>Smith</ns0:LastName>
          <ns0:Department>Sales</ns0:Department>
          <ns0:HireDate>2004-12-01</ns0:HireDate>
          <ns0:Phone>5951</ns0:Phone>
        </ns0:Employee>
      </SourceDoc>
      <StyleSheetLocator>
        <MapName>com/company1/myapplication/EmployeeCommonToNorthwind</MapName>
      </StyleSheetLocator>
    </XSLtransform>
  </SOAP:Body>
</SOAP:Envelope>
```

☐ Transactional
 ☐ No Reply

Invoke

5. Remove the elements *MapID* and *version*.
6. Change *<SourceDoc/>* to *<SourceDoc></SourceDoc>*
7. Paste the *Employee* structure you copied before in the *SourceDoc* element.

8. Specify *MapName*: (see below text and screenshot)  
**com/companyX/myapplication/EmployeeCommonToNorthwind**

Tip: You can find the MapName by copying the Qualified Name of the data transformation and change the dots “.” in slashes “/”, where the dot represents a directory separation.

**EmployeeCommonToNorthwind - Data Transformation**

General Annotations History

Name: EmployeeCommonToNorthwind

Description: Transform common employee to Northwind employee

Type: Data Transformation

Location: My Application Project/Data Transformations/com/company1/myapplication

Qualified Name: com.company1.myapplication.EmployeeCommonToNorthwind

With a Direct Access URL (REST) you can access this document directly. Click [here](#) to copy the Direct Access URL

OK Cancel

9. Click **Invoke**.

*What are the pros and cons of using the generic service operation or generate a specific web service operation to invoke a data transformation?*

---



---



## 3.5 Using Transformations in Business Process

In this exercise you will use your custom data transformation web service operation in a business process.

### 3.5.1 Prerequisites

This exercise assumes you have successfully completed the module Developing Business Processes. If you did not finish that module you can safely ignore this exercise.

### 3.5.2 Analyzing Situation

1. Read the analysis:

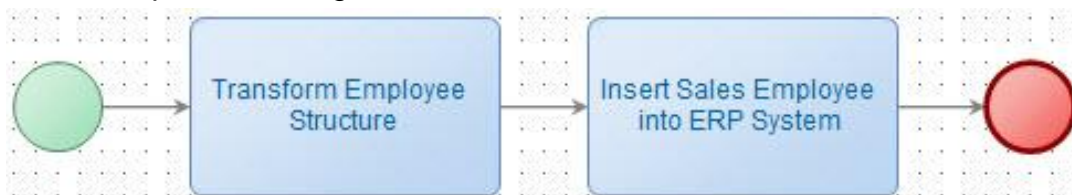
You will create a simple process that will use the common Employee structure as input message. The process will transform the data to the Northwind employee structure and subsequently insert the employee into the Northwind backend.

### 3.5.3 Creating the Business Process

1. Create a new *Business Process Model* with the following details:

Field	Value
Name	NewSalesEmployee
Description	Configure new sales employee setup
Location	My Application Project/Business Process Models/com/companyX/myapplication

2. Open the process properties.
3. Change the namespace to:  
**<http://schemas.companyX.com/myapplication/salesprocesses>**
4. Model the process design similar to:



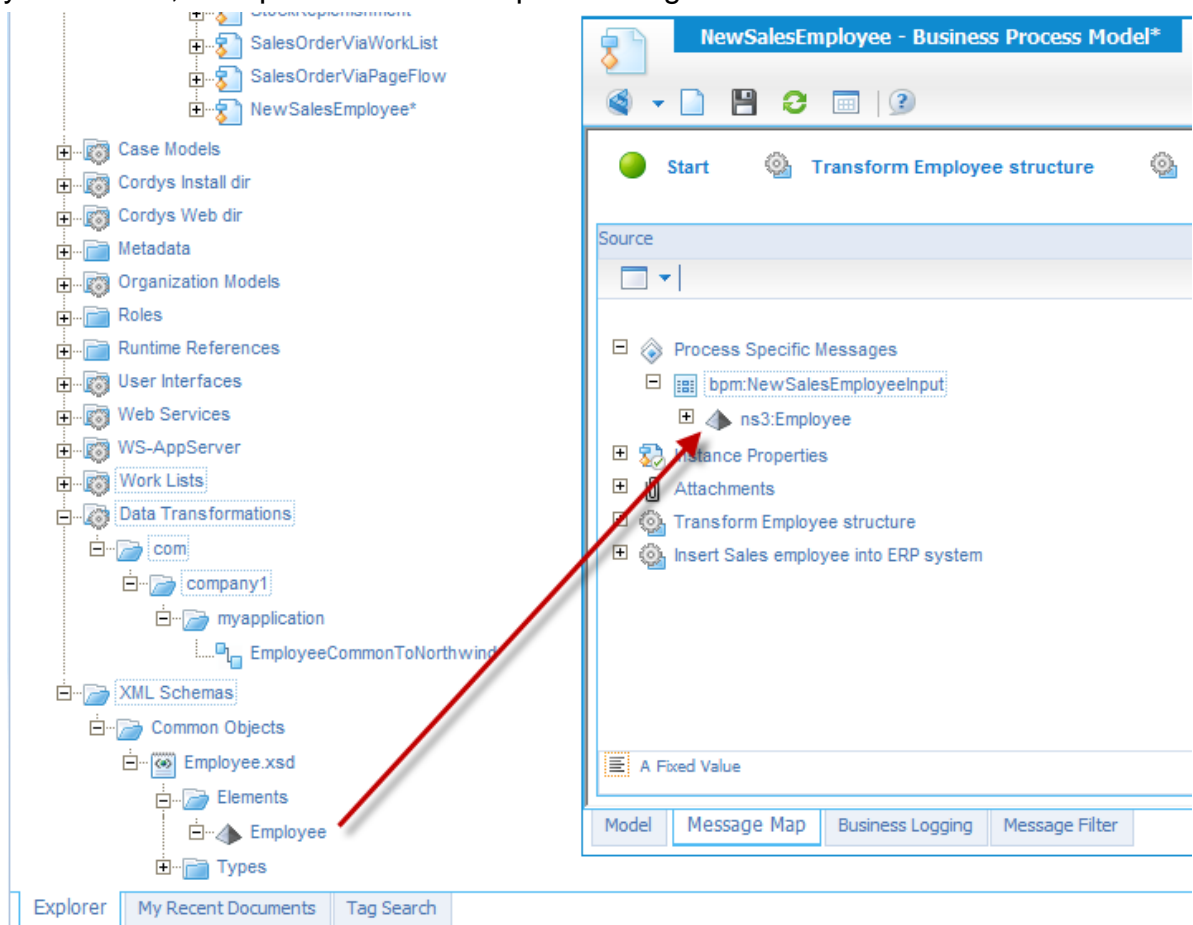
5. Drag and drop the *EmployeeToNorthwind* Web Service operation on the first activity.
6. Drag and drop the *UpdateEmployees (Northwind → Basic)* web service operation on the second activity.

#### Data flow

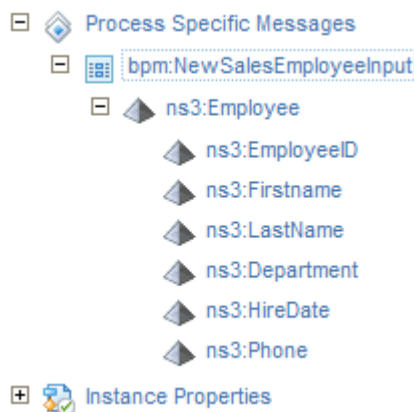
1. Go to the *Message Map*.
2. Right click *Process Specific Message* and select *Create Message*.



3. Enter the message name: **NewSalesEmployeeInput**.
4. From the *Workspace Documents*, drag and drop the *Employee* schema fragment, you created, on top of the created input message.

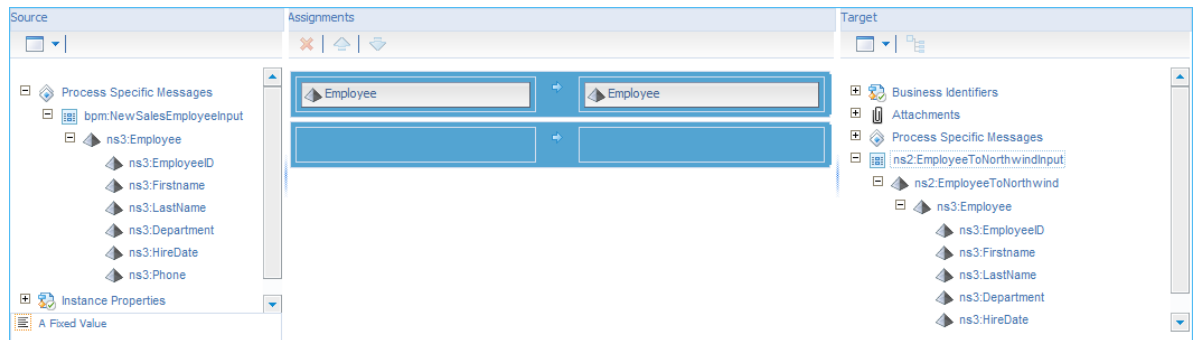


Notice the difference in the namespace (prefix) of the message and the employee element!

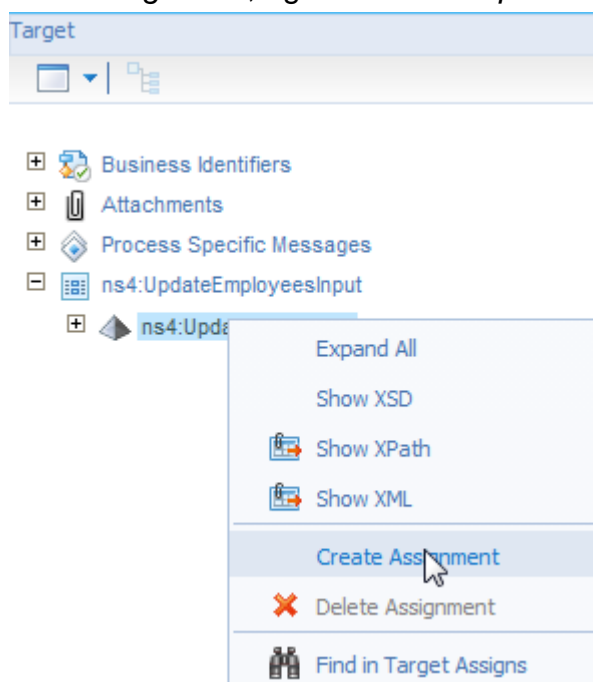


5. On the properties of the *Start Event* select the created message as the *Input Message*.
6. Select the *Transform* activity.
7. Return to the Message Map.

8. Create an assignment from *ns3:Employee* (process specific message) to *ns3:Employee* (input message transform).



9. In the *Use* select box select **Replace With** → **Select**. (the namespace is the same ns3, so **Select** is okay)
10. Select the *Insert* activity.
11. In the *Target* box, right Click *ns4:UpdateEmployees* and select Create assignment.



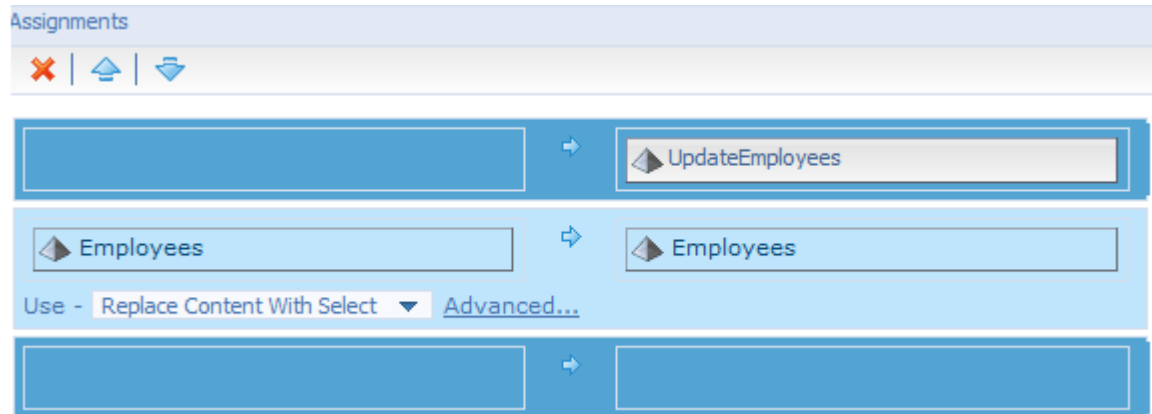
12. Select *Use*: **Delete**.

## NOTE

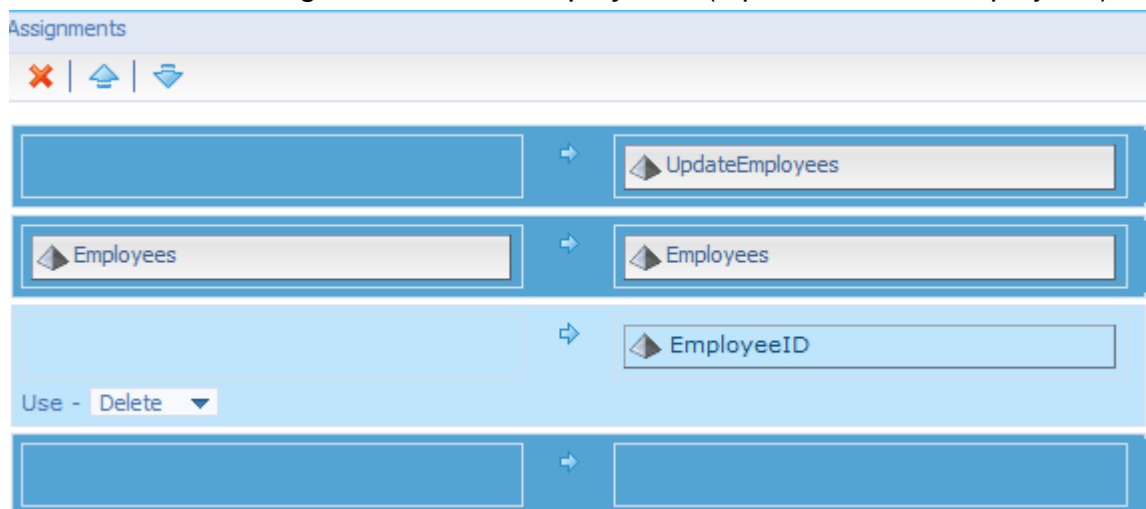
In the same process, you can include multiple actions on the same backend (i.e. insert, update and delete data) using the same input message structure. To safeguard that these actions and their data (i.e. the input message) do not interfere with each other, it is good practice to first perform a delete assignment to the input message of the backend action. This will ensure that any input data of a preceding action is removed.

The update operation can perform an insert, update and delete of the relevant data. The action to be performed is based on the input message. When the input message only contains a tuple → new structure, this implies an insert, since only new data is involved.

13. Create an assignment from *ns4:Employees* (transform output) to *ns4:Employees* (ns4:tuple → ns4:new input message insert)



14. Use: **Replace With** → **Select**. (the namespace is the same ns4, so **Select** is okay).  
 15. Create a **Delete** assignment for *ns4:EmployeeID* (tuple → new → Employees).



Why do you have to model a delete assignment for the *EmployeeID*?

---



---

16. Return to the *Model* tab.

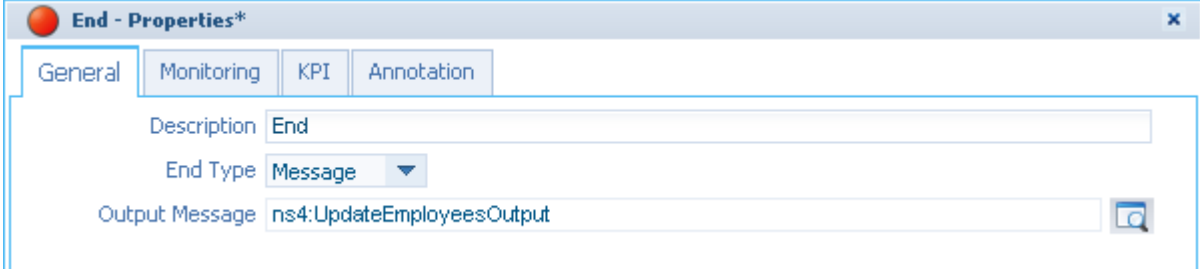
Is this a short or a long lived process?

---



---

17. Open the process properties.  
 18. Change *Execution Mode* to *Short Lived*.  
 19. Open the End Event properties  
 20. Set *End Type* to *Message*.

**21.** Select Output Message: *UpdateEmployeesOutput*.


The 'End - Properties' dialog box is shown with the 'General' tab selected. It contains the following fields:

- Description: End
- End Type: Message (dropdown menu)
- Output Message: ns4:UpdateEmployeesOutput

For the end event you simply use the web service output message directly to show the result instead of creating a process specific message and use that as output for your process.

**22.** Save, validate and publish your process.**23.** Make sure that your *Northwind* Service Container is started.**NOTE**

Different backends use different formats for entering date and time values. To circumvent this problem with Cordys you have to apply the Cordys date time format:

yyyy-mm-ddThh:mm:ss.Ms[MsMs], e.g. 2012-01-31T13:30:00.0

If you do not provide the time part 00:00:00.0 will be assumed

**24.** Run the process, and enter values for the new employee:

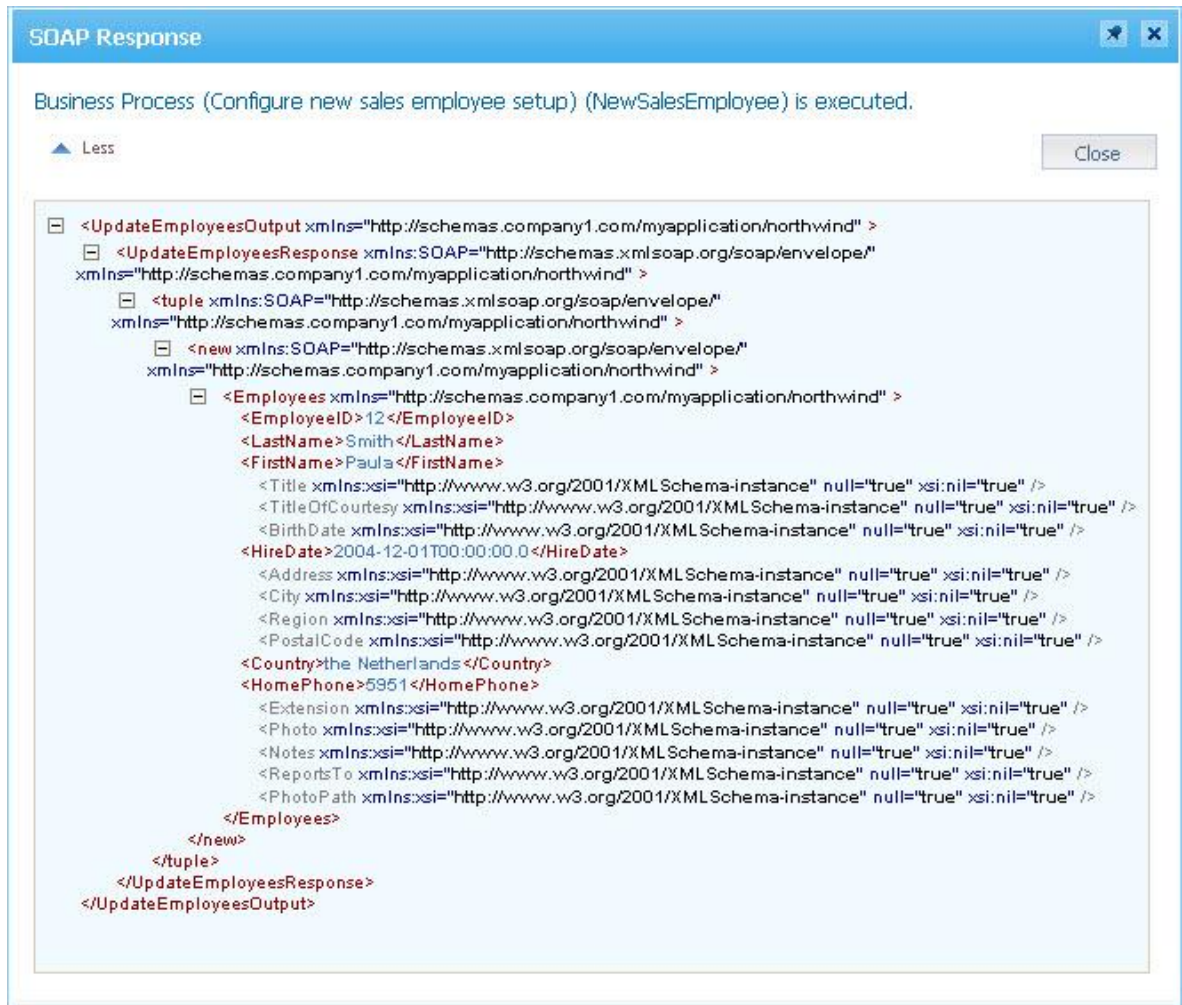

The 'Start Message' dialog box is shown with the following XML content in the text area:

```
<NewSalesEmployeeInput xmlns="http://schemas.company1.com/myapplication/salesprocesses">
  <Employee xmlns="http://schemas.company1.com/myapplication/employee">
    <EmployeeID xmlns="http://schemas.company1.com/myapplication/employee">123</EmployeeID>
    <FirstName xmlns="http://schemas.company1.com/myapplication/employee">Paula</FirstName>
    <LastName xmlns="http://schemas.company1.com/myapplication/employee">Smith</LastName>
    <Department xmlns="http://schemas.company1.com/myapplication/employee">Sales</Department>
    <HireDate xmlns="http://schemas.company1.com/myapplication/employee">2004-12-01</HireDate>
    <Phone xmlns="http://schemas.company1.com/myapplication/employee">5951</Phone>
  </Employee>
</NewSalesEmployeeInput>
```

At the bottom of the dialog are 'OK' and 'Cancel' buttons.

**25.** Click OK.

26. You should receive a response message similar to:



*What is the advantage of creating a process specific message for start and end event?*

---



---

*Is it possible to get the same functionality without using the transformation service?*

---



---

## 3.6 Transformations in User Interfaces

*How can you make use of transformation services in user interfaces?*

---



---


## 3.7 Make Changes Available to SCM

In this exercise you will make your developed content/changes available to your team members by sending the changes to the SCM application.

You should only make changes available after you have tested these to work correctly. This is to ensure that your team members' work is not affected when they incorporate your changes.

### NOTE

This only applies when your workspace was created using an SCM application.

1. If closed, open the Workspace Documents.
2. Click **Make Changes Available to Others** () in the toolbar.
3. Review the modified content.
4. Provide as comment **Data Transformations**.
5. Click **Make Available**.

## 4. Learning Report

---

### Achievements

- ☐ I know the concept of *business objects*.
- ☐ I know the advantage of business object abstraction.
- ☐ I can create XML schema definitions to reflect business data.
- ☐ I can create data transformation models.
- ☐ I can use data transformations in a business process.

### Notes