



Creating XML with Uniface

**22 November
2018**

Daniel Iseli, Senior Technical Support Analyst

Agenda

- Task
- Uniface XML functionality
- Additional tools
- Use case

Task

- Creating complex XML (based on eCH standards)
- Homogeneous and efficient Uniface solution

XML handling

- Uniface offers standard functionality for creating and transforming XML data
 - STRUCTS
 - A tree-like data structure in memory that is used to dynamically manipulate complex data and transform it (e.g.) from or to XML or Uniface component data.
 - Available since Uniface 9.5 (2013)

XML handling

- UXMLWRITER
 - Uniface component that allows you to generate XML documents instead of encoding each character in the XML stream itself.
 - Implemented using the SAX XML parser
 - Powerful for large XML files > 50 MB
 - Available since Uniface 9.1 (2007)

Selected procedure with Structs

- Read the required data according to the XSD structure
- componentToStruct maps the retrieved data structure as a Struct
- Modifications of the Struct so that XSD-compliant XML can be generated
- structToXml/schema creates XML file according to XSD
- Validation according to XSD

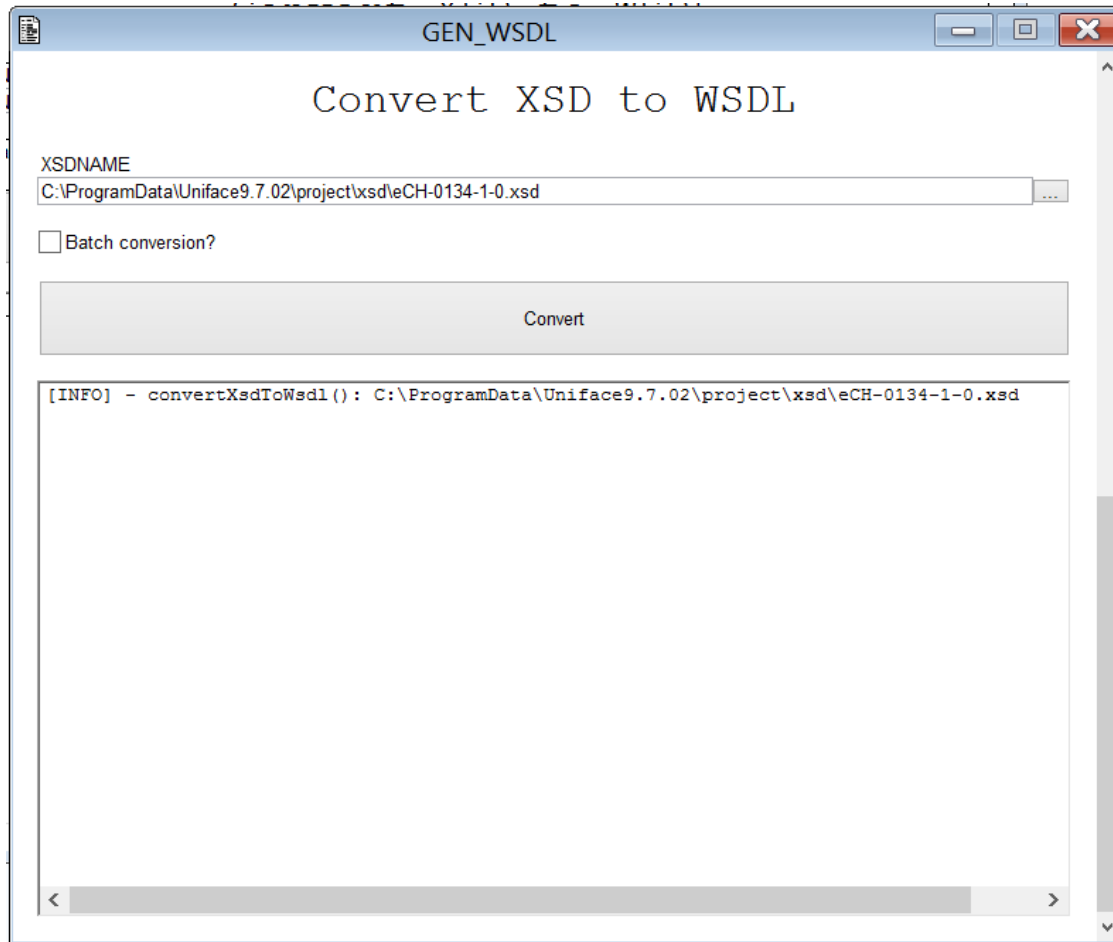
Use case newOwnershipPart (eCH-0134)

- Goal is to create XML according to eCH standard
- Splitting up the use case in 5 parts for reasons of clarity:
 - eCH-0134: deliveryHeader (DELIVERY_XML)
 - gbdbbs:AnmeldungType (DELIVERY_XML1)
 - eCH-0178:legalGroundExhibitType (DELIVERY_XML2)
 - gbdbbs:EigentumAnteilType (DELIVERY_XML3)
 - gbdbbs:PersonGBType (DELIVERY_XML4)
 - Putting all the parts together and create the XML output (DELIVERY_START)

Additional (homebrewed) tools

- Convert XSD to WSDL file (GEN_WSDL)
- Importing WSDL
- Generating Technical Keys for the Imported Models (GEN_TECH_PK)
- XSD Analyzer (XSD2UNI)

Convert XSD to WSDL file (GEN_WSDL)



Apache CXF

- Using Apache CXF, XSD files are converted to WSDL files.
- Operations and parameters are added to the WSDL so that it can be imported into Uniface as a web service.
- <http://cxf.apache.org/download.html>
- Java SDK shipped with Uniface used

Convert XSD to WSDL file (GEN_WSDL)

```
<xsd:complexType name="newOwnershipPart">  
  <xsd:complexContent>  
    <xsd:extension base="eCH-0134:baseMessageType">  
      <xsd:sequence>  
        <xsd:element name="delivery10">  
          <xsd:choice>  
            <xsd:element name="deliveryHeader" type="eCH-0058:headerType"/>  
            <xsd:element name="newOwnershipPart" type="eCH-0134:newOwnershipPart"/>  
          </xsd:choice>  
        </xsd:sequence>  
      </xsd:extension>  
    </xsd:complexContent>  
  </xsd:complexType>
```

XSD

WSDL

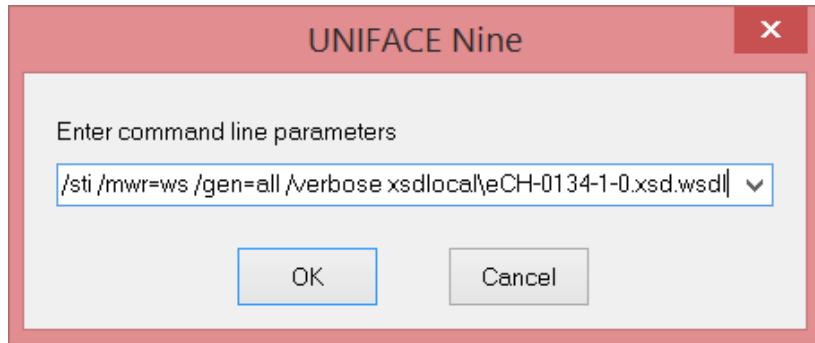
```
<element name="delivery10">  
  <complexType>  
    <sequence>  
      <xsd:element name="deliveryHeader" type="eCH-0058:headerType"/>  
      <xsd:element name="newOwnershipPart" type="eCH-0134:newOwnershipPart"/>  
    </sequence>  
  </complexType>  
</element>  
  
<message name="delivery10Request">  
  <part name="parameters" element="eCH-0134-1-0:delivery10"/>  
</message>  
  
<operation name="delivery10OPER">  
  <input message="eCH-0134-1-0:delivery10Request"/>  
</operation>  
<operation name="delivery10OPER">  
  <soap:operation soapAction=""/>  
  <input>  
    <soap:body use="literal"/>  
  </input>  
</operation>
```

Import des WSDL

DRIVER_SETTINGS]

... .

SOP U2.0

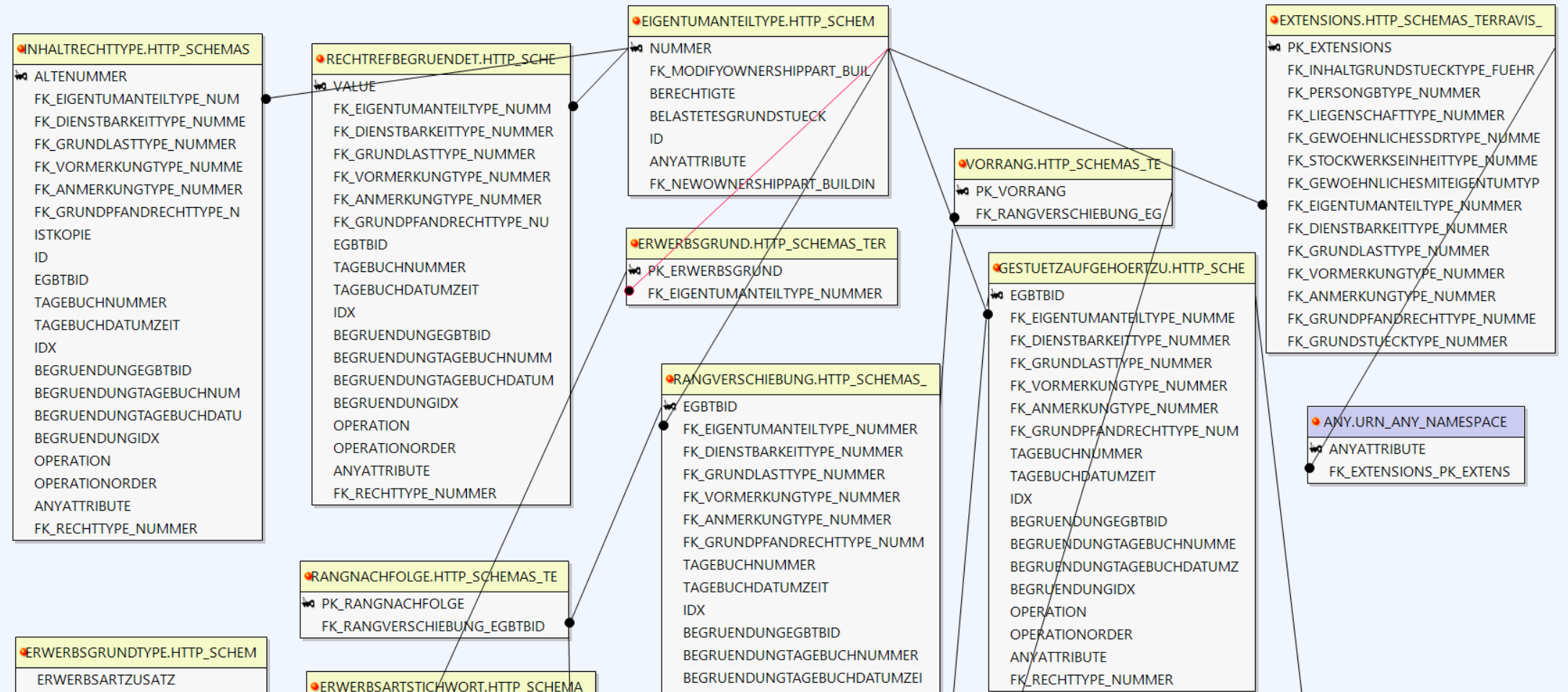


- This generates the signature of the Web service with operations and parameters.
- Models, entities and relations are created according to the XSD structures

```
2018-06-28 17:22:32.53 - Uniface session started
INFO: Import WSDL File Utility for Web Services Call-Out U2.0
INFO:   WSDL File:xsdlocal\ech-0134-1-0.wsdl
INFO:   Options: Create signatures, XML templates, parameter entities,
INFO:             complex types as entities, repeating fields as entities,
INFO:             repeating groups as entities, choice contents as entities,
INFO:             any types as entities, any attributes as fields,
INFO:             entity relationships, primary key fields, Verbose
...
INFO:   Creating entity: NEWOWNERSHIPPART
INFO:   Entity: NEWOWNERSHIPPART created
INFO:   Group: NEWOWNERSHIPPART created
INFO:   Creating field: BUILDINGINSURANCENUMBER
```

... .

Import des WSDL



Imported Signature from the WSDL

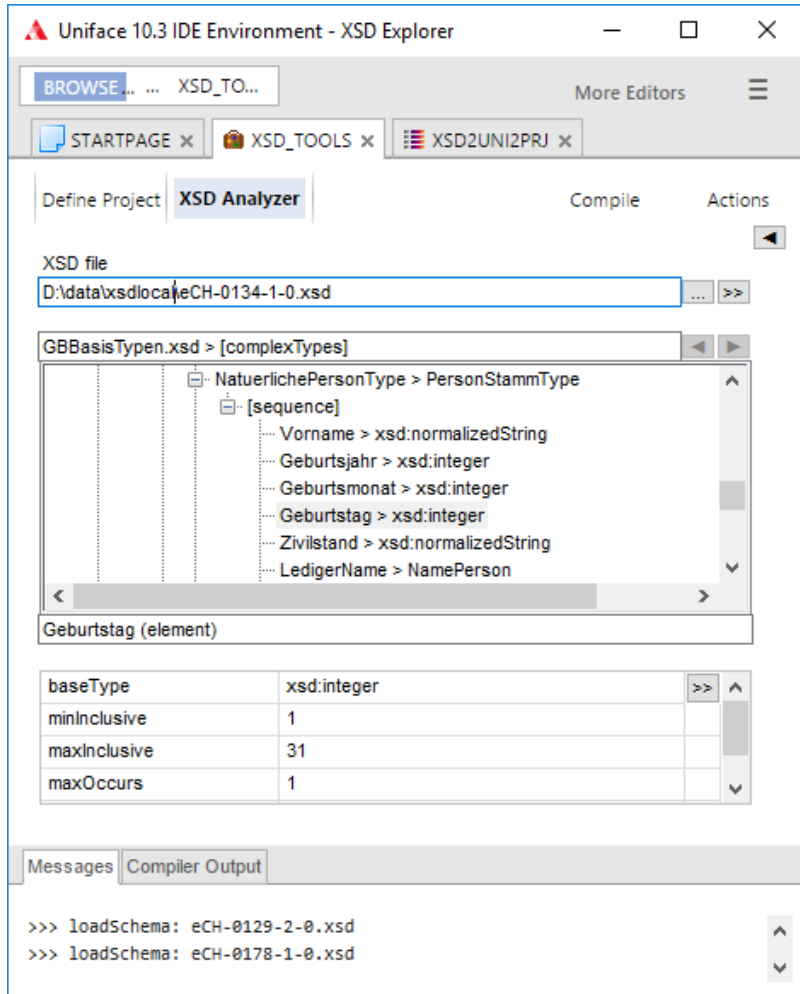
The image shows a software interface with two overlapping windows. The background window is titled 'Define Parameter: DELIVERY10' and has a 'Details of 'DELIVERYHEADER'' section. In this section, 'Data Type' is set to 'String' and 'Basic' is selected under 'Data Type' options. The 'Implementations' list contains 'SOAP'. The foreground window is titled 'Parameter Content' and displays the following XML schema:

```
<uwi0:deliveryHeader xmlns:uwi0="http://www.ech.ch/xmlns/eCH-0134/1">  
<uwi1:senderId xmlns:uwi1="http://www.ech.ch/xmlns/eCH-0058/4">participantIdType</uwi1:senderId>  
<!--  
  Element originalSenderId minOccurs="0" -->  
<uwi1:originalSenderId xmlns:uwi1="http://www.ech.ch/xmlns/eCH-0058/4">participantIdType</uwi1:originalSenderId>  
<!--  
  Element declarationLocalReference minOccurs="0"  
  Element declarationLocalReference: minLength="1", maxLength="100", whiteSpace="collapse" -->  
<uwi1:declarationLocalReference xmlns:uwi1="http://www.ech.ch/xmlns/eCH-0058/4">declarationLocalReferenceType</uwi1:de  
<!--  
  Element recipientId minOccurs="0" maxOccurs="unbounded" -->  
<uwi1:recipientId xmlns:uwi1="http://www.ech.ch/xmlns/eCH-0058/4">participantIdType</uwi1:recipientId>  
<!--  
  Element messageId: minLength="1", maxLength="36", whiteSpace="collapse" -->  
<uwi1:messageId xmlns:uwi1="http://www.ech.ch/xmlns/eCH-0058/4">messageIdType</uwi1:messageId>  
<!--  
  Element referenceMessageId minOccurs="0"  
  Element referenceMessageId: minLength="1", maxLength="36", whiteSpace="collapse" -->  
<uwi1:referenceMessageId xmlns:uwi1="http://www.ech.ch/xmlns/eCH-0058/4">messageIdType</uwi1:referenceMessageId>  
<!--  
  Element businessProcessId minOccurs="0"  
  Element businessProcessId: minLength="1", maxLength="128", whiteSpace="collapse" -->  
<uwi1:businessProcessId xmlns:uwi1="http://www.ech.ch/xmlns/eCH-0058/4">businessProcessIdType</uwi1:businessProcessId:  
<!--
```

Generating Technical Keys for the Imported Models (GEN_TECH_PK)

- The first simple type of the XSD is automatically defined as the primary key for imported models, which can lead to unwanted blank references in the generated XML files.
- GEN_TECH_PK replaces the automatically defined PK field for all imported models (HTTP*) with a new PK_<EntName> field.

XSD Analyzer (XSD2UNI)



The screenshot shows the Uniface 10.3 IDE Environment with the XSD Explorer tool. The tool is displaying the XSD file `D:\data\xsd\local\ech-0134-1-0.xsd`. The tree structure shows the following elements:

- GBBasisTypen.xsd > [complexType]
- NatuerlichePersonType > PersonStammType
- [sequence]
- Vorname > xsd:normalizedString
- Geburtsjahr > xsd:integer
- Geburtsmonat > xsd:integer
- Geburtstag > xsd:integer
- Zivilstand > xsd:normalizedString
- LedigerName > NamePerson

The 'Geburtstag (element)' table shows the following restrictions:

Property	Value
baseType	xsd:integer
minInclusive	1
maxInclusive	31
maxOccurs	1

The Compiler Output window shows the following messages:

```
>>> loadSchema: ech-0129-2-0.xsd
>>> loadSchema: ech-0178-1-0.xsd
```

- XSD as tree structure
- Imports
- Data types
- Restrictions, Enumerations
- Choices
- Substitutions
- Mapping for generated entities
- «Enrich» generated entities

XSD Choices

```
<xsd:complexType name="newOwnershipPart">
  <xsd:complexContent>
    <xsd:extension base="eCH-0134:baseMessageType">
.....
<xsd:complexType name="baseMessageType">
  <xsd:sequence>
    <xsd:element name="commercialTransactionInfo">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="commercialTransaction" type="gbdb:AnmeldungType"/>
          <xsd:element name="legalGroundExhibit" type="eCH-0178:legalGroundExhibitType"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:choice minOccurs="0">
      <xsd:element name="realEstateId">
        .....
      </xsd:element>
      <xsd:element name="personId">
        .....
    </xsd:choice>
    <xsd:element ref="eCH-0134:extensions" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

XSD Substitution

```
<xsd:complexType name="EigentumAnteilType">
  <xsd:complexContent>
    <xsd:extension base="RechtType">

<xsd:complexType name="RechtType">
  <xsd:sequence>
    <xsd:element ref="InhaltRecht" maxOccurs="1"
    <xsd:element name="Rangverschiebung" minOcc

<xsd:element name="InhaltEigentumAnteil" type="Ir
  substitutionGroup="InhaltRecht"/>
<xsd:complexType name="InhaltEigentumAnteilType">
<xsd:complexContent>
  <xsd:extension base="InhaltRechtType">
```

Name
DELIVERY_XML3
EIGENTUMANTEILTYPE.HTTP_SCHEMAS_TERRAVIS_C...
BERECHTIGTE.EIGENTUMANTEILTYPE
BERECHTIGTE
BELASTETESGRUNDSTUECK.EIGENTUMANTEILTYPE
BELASTETESGRUNDSTUECK
NUMMER.EIGENTUMANTEILTYPE
NUMMER
RECHTTYPE.HTTP_SCHEMAS_TERRAVIS_CH_GBBAS...
INHALTEIGENTUMANTEILTYPE.HTTP_SCHEMAS_...
INHALTRECHTTYPE.HTTP_SCHEMAS_TERRAVI...
BEGRUENDUNGEGBTBID.INHALTRECHTTYPE
BEGRUENDUNGEGBTBID

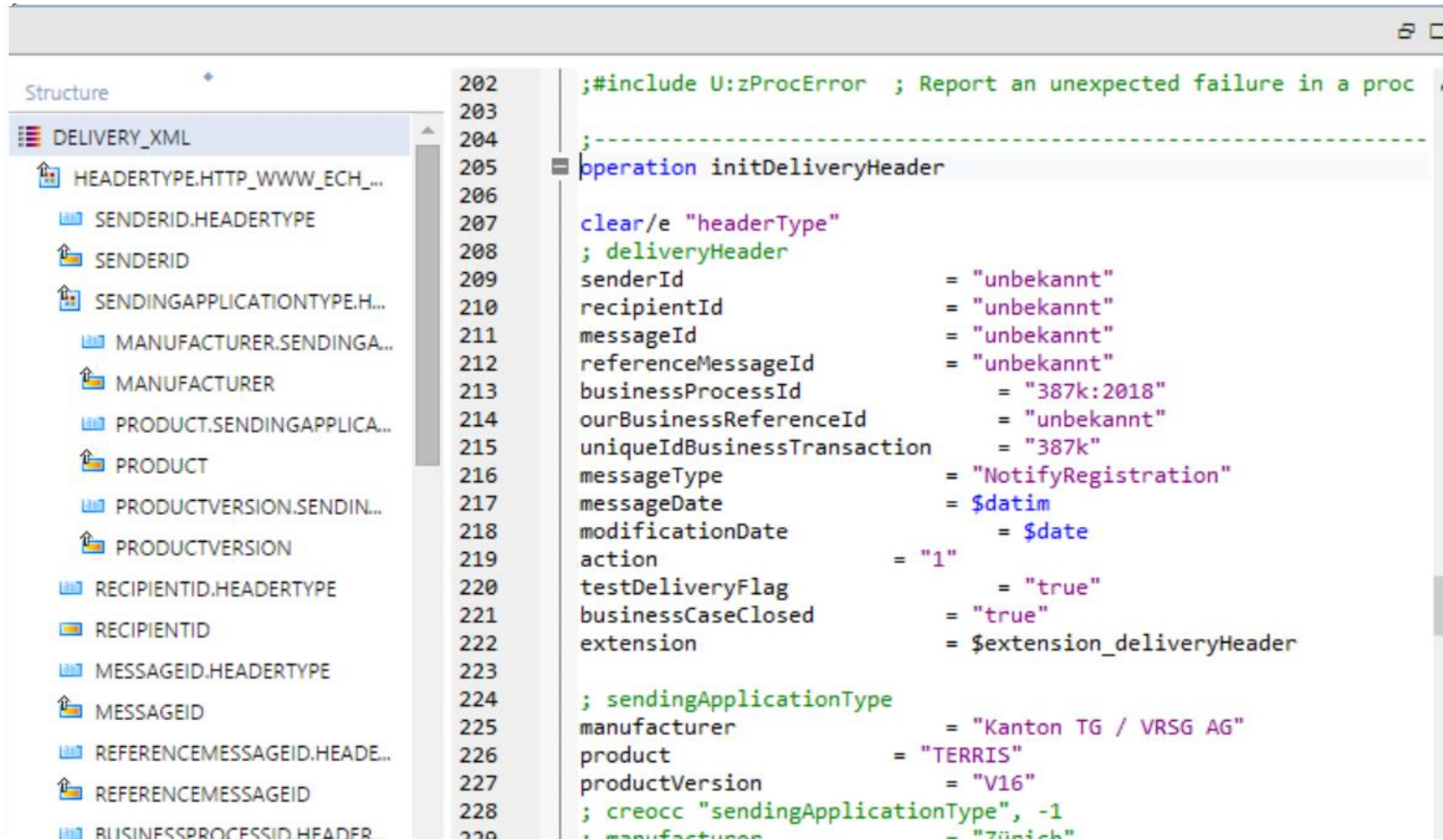
Read the required data according to the XSD

- Using the Imported Entities in the XSD Structure
- Simulation of reading the data by assigning initial values to the corresponding fields

Read the required data according to the XSD

```
<xs:complexType name="headerType">
  <xs:sequence>
    <xs:element name="senderId" type="eCH-0058:participantIdType"/>
    <xs:element name="originalSenderId" type="eCH-0058:participantIdType" minOccurs="0"/>
    <xs:element name="declarationLocalReference" type="eCH-0058:declarationLocalReferenceType" minOccurs="0"/>
    <xs:element name="recipientId" type="eCH-0058:participantIdType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="messageId" type="eCH-0058:messageIdType"/>
    .....
    <xs:element name="sendingApplication" type="eCH-0058:sendingApplicationType"/>
    <xs:element name="partialDelivery" type="eCH-0058:partialDeliveryType" minOccurs="0"/>
    .....
    <xs:element name="subject" type="eCH-0058:subjectType" minOccurs="0"/>
    <xs:element name="comment" type="eCH-0058:commentType" minOccurs="0"/>
```

Read the required data according to the XSD



The screenshot displays the SAP ABAP development environment. On the left, the 'Structure' browser shows a tree view for 'DELIVERY_XML'. The tree includes several nodes, with 'DELIVERY_XML' expanded to show its components: HEADERTYPE.HTTP_WWW_ECH..., SENDERID.HEADERTYPE, SENDERID, SENDINGAPPLICATIONTYPE.H..., MANUFACTURER.SENDINGA..., MANUFACTURER, PRODUCT.SENDINGAPPLICA..., PRODUCT, PRODUCTVERSION.SENDIN..., and PRODUCTVERSION. Below these are RECIPIENTID.HEADERTYPE, RECIPIENTID, MESSAGEID.HEADERTYPE, MESSAGEID, REFERENCEMESSAGEID.HEADE..., REFERENCEMESSAGEID, and BUSINESSPROCESSID HEADER.

The main editor window shows ABAP code for an 'operation initDeliveryHeader'. The code includes a comment: `;-#include U:zProcError ; Report an unexpected failure in a proc`. The code then defines several variables and assigns them values:

```
clear/e "headerType"
; deliveryHeader
senderId                = "unbekannt"
recipientId             = "unbekannt"
messageId               = "unbekannt"
referenceMessageId     = "unbekannt"
businessProcessId      = "387k:2018"
ourBusinessReferenceId = "unbekannt"
uniqueIdBusinessTransaction = "387k"
messageType             = "NotifyRegistration"
messageDate             = $datim
modificationDate       = $date
action                  = "1"
testDeliveryFlag       = "true"
businessCaseClosed     = "true"
extension               = $extension_deliveryHeader

; sendingApplicationType
manufacturer            = "Kanton TG / VRSG AG"
product                 = "TERRIS"
productVersion          = "V16"
; creocc "sendingApplicationType", -1
; manufacturer          = "Zürich"
```

Mapping of the read data structure as a Struct

```
componenttostruct $vStruct$, "headerType"
```

```
[HEADERTYPE.HTTP_WWW_ECH_CH_XMLNS_ECH_0058_4]  
  [OCC]  
    [SENDERID] = "unbekannt"  
    [SENDINGAPPLICATIONTYPE.HTTP_WWW_ECH_CH_XMLNS_ECH_0058_4]  
      [OCC]  
        [MANUFACTURER] = "Kanton TG / VRSG AG"  
        [PRODUCT] = "TERRIS"  
        [PRODUCTVERSION] = "V16"  
    [RECIPIENTID] = "unbekannt"  
    [MESSAGEID] = "unbekannt"  
    [REFERENCEMESSAGEID] = "unbekannt"  
    [BUSINESSPROCESSID] = "387k:2018"  
    [PARTIALDELIVERYTYPE.HTTP_WWW_ECH_CH_XMLNS_ECH_0058_4]  
      [OCC]  
        [UNIQUEIDDELIVERY] = "ID11111"  
        [TOTALNUMBEROFPACKAGES] = 12  
        [NUMBEROFACTUALPACKAGE] = 25
```

Target format of Struct

```
$vStruct$->OCC->$name = "deliveryHeader"
```

```
$vStruct$->deliveryHeader-><partialDelivery>->OCC->$name = "partialDelivery"
```

```
$vStruct$->deliveryHeader-><sendingApplication>->OCC->$name = "sendingApplication"
```

```
[HEADERTYPE.HTTP_WWW_ECH_CH_XMLNS_ECH_0058_4]
```

```
  [deliveryHeader]
```

```
    [SENDERID] = "unbekannt"
```

```
    [SENDINGAPPLICATIONTYPE.HTTP_WWW_ECH_CH_XMLNS_ECH_0058_4]
```

```
      [sendingApplication]
```

```
        [MANUFACTURER] = "Kanton TG / VRSG AG"
```

```
        [PRODUCT] = "TERRIS"
```

```
        [PRODUCTVERSION] = "V16"
```

```
    [RECIPIENTID] = "unbekannt"
```

```
    [MESSAGEID] = "unbekannt"
```

```
    [REFERENCEMESSAGEID] = "unbekannt"
```

```
    [BUSINESSPROCESSID] = "387k:2018"
```

```
    [PARTIALDELIVERYTYPE.HTTP_WWW_ECH_CH_XMLNS_ECH_0058_4]
```

```
      [partialDelivery]
```

```
        [UNIQUEIDDELIVERY] = "ID111111"
```

```
        [TOTALNUMBEROFFPACKAGES] = 12
```

```
        [NUMBEROFACTUALPACKAGE] = 25
```

Modifying the Struct

eCH-0134-1-0.xsd:

```
<xs:complexType name="headerType">
  <xs:sequence>
    <xs:element name="senderId" type="eCH-0058:participantIdType"/>
    <xs:element name="originalSenderId" type="eCH-0058:participantIdType" minOccurs="0"/>
    ....
    <xs:element name="sendingApplication" type="eCH-0058:sendingApplicationType"/>
    <xs:element name="partialDelivery" type="eCH-0058:partialDeliveryType" minOccurs="0"/>
    ....
  </xs:sequence>
</xs:complexType>
```


Creating the XML

```
putitem vSchemaList, -1, "xsdlocal\eCH-0134-1-0.xsd"  
putitem vSchemaList, -1, "xsdlocal\eCH-0058-4-0.xsd"  
structToXml/schema p_out, $vStruct$, vSchemaList, "element=delivery"
```

```
<ns0:delivery xmlns:ns0="http://www.ech.ch/xmlns/eCH-0134/1">  
  <ns0:deliveryHeader>  
    <ns1:senderId xmlns:ns1="http://www.ech.ch/xmlns/eCH-0058/4">unbekannt</ns1:senderId>  
    <ns1:recipientId xmlns:ns1="http://www.ech.ch/xmlns/eCH-0058/4">unbekannt</ns1:recipientId>  
    <ns1:messageId xmlns:ns1="http://www.ech.ch/xmlns/eCH-0058/4">unbekannt</ns1:messageId>  
    <ns1:referenceMessageId xmlns:ns1="http://www.ech.ch/xmlns/eCH-0058/4">unbekannt</ns1:referenceMessageId>  
    <ns1:businessProcessId xmlns:ns1="http://www.ech.ch/xmlns/eCH-0058/4">387k:2018</ns1:businessProcessId>  
    <ns1:ourBusinessReferenceId xmlns:ns1="http://www.ech.ch/xmlns/eCH-0058/4">unbekannt</ns1:ourBusinessReferenceId>  
    <ns1:uniqueIdBusinessTransaction xmlns:ns1="http://www.ech.ch/xmlns/eCH-0058/4">387k</ns1:uniqueIdBusinessTransaction>  
    <ns1:messageType xmlns:ns1="http://www.ech.ch/xmlns/eCH-0058/4">NotifyRegistration</ns1:messageType>  
    <ns1:sendingApplication xmlns:ns1="http://www.ech.ch/xmlns/eCH-0058/4">  
      <ns1:manufacturer>Kanton TG / VRSG AG</ns1:manufacturer>  
      <ns1:product>TERRIS</ns1:product>  
      <ns1:productVersion>V16</ns1:productVersion>  
    </ns1:sendingApplication>  
  </ns0:deliveryHeader>  
</ns0:delivery>
```

.....

Putting it all together

```
activate "DELIVERY_XML".getDeliveryHeader($vStruct$)
activate "DELIVERY_XML1".getAnmeldungType(AnmeldungTypeStruct)
activate "DELIVERY_XML2".getlegalGroundExhibitType(legalGroundExhibitTypeStruct)
activate "DELIVERY_XML3".getEigentumAnteilType(EigentumAnteilTypeStruct)
activate "DELIVERY_XML4".getPersonGBType(PersonGBTypeStruct)

call modifyTags

call insertAnmeldungType(AnmeldungTypeStruct)
call insertlegalGroundExhibitType(legalGroundExhibitTypeStruct)
call insertEigentumAnteilType(EigentumAnteilTypeStruct)
call insertPersonGBType(PersonGBTypeStruct)

structtoxml output, $vStruct$ ; Endergebnis
```

Putting it all together

```
entry insertAnmeldungType
```

```
params
```

```
struct AnmeldungTypeStruct :IN
```

```
endparams
```

```
$vStruct$->delivery->newOwnershipPart->commercialTransactionInfo->commercialTransaction =  
AnmeldungTypeStruct->AnmeldungType
```

```
$vStruct$->delivery->newOwnershipPart->commercialTransactionInfo->commercialTransaction->  
>$tags->xmlClass = "element"
```

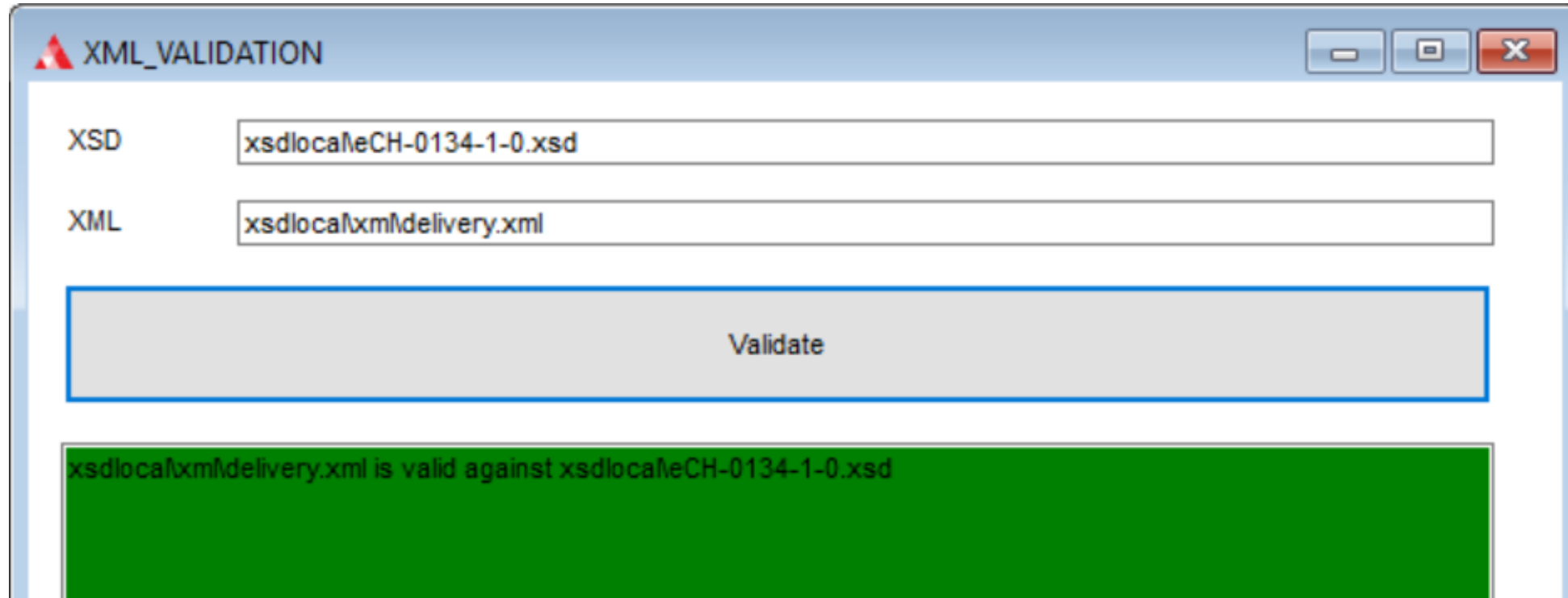
```
$vStruct$->delivery->newOwnershipPart->commercialTransactionInfo->commercialTransaction->  
>$tags->xmlNamespaceURI = "http://www.ech.ch/xmlns/eCH-0134/1"
```

```
$vStruct$->delivery->newOwnershipPart->commercialTransactionInfo->commercialTransaction->  
>$tags->xmlTypeCategory = "complex"
```

```
$vStruct$->delivery->newOwnershipPart->commercialTransactionInfo->commercialTransaction->  
>$tags->xmlDataType = "AnmeldungType"
```

```
$vStruct$->delivery->newOwnershipPart->commercialTransactionInfo->commercialTransaction->  
>$tags->xmlTypeNamespace = "http://schemas.terravis.ch/GBBasisTypen/2.0"
```

Validation



- Validation with Java XSDValidator



Demo

THANK YOU & QUESTIONS

