UNIFACE

Advanced Development Technology

Creating XML with Uniface

22 November 2018

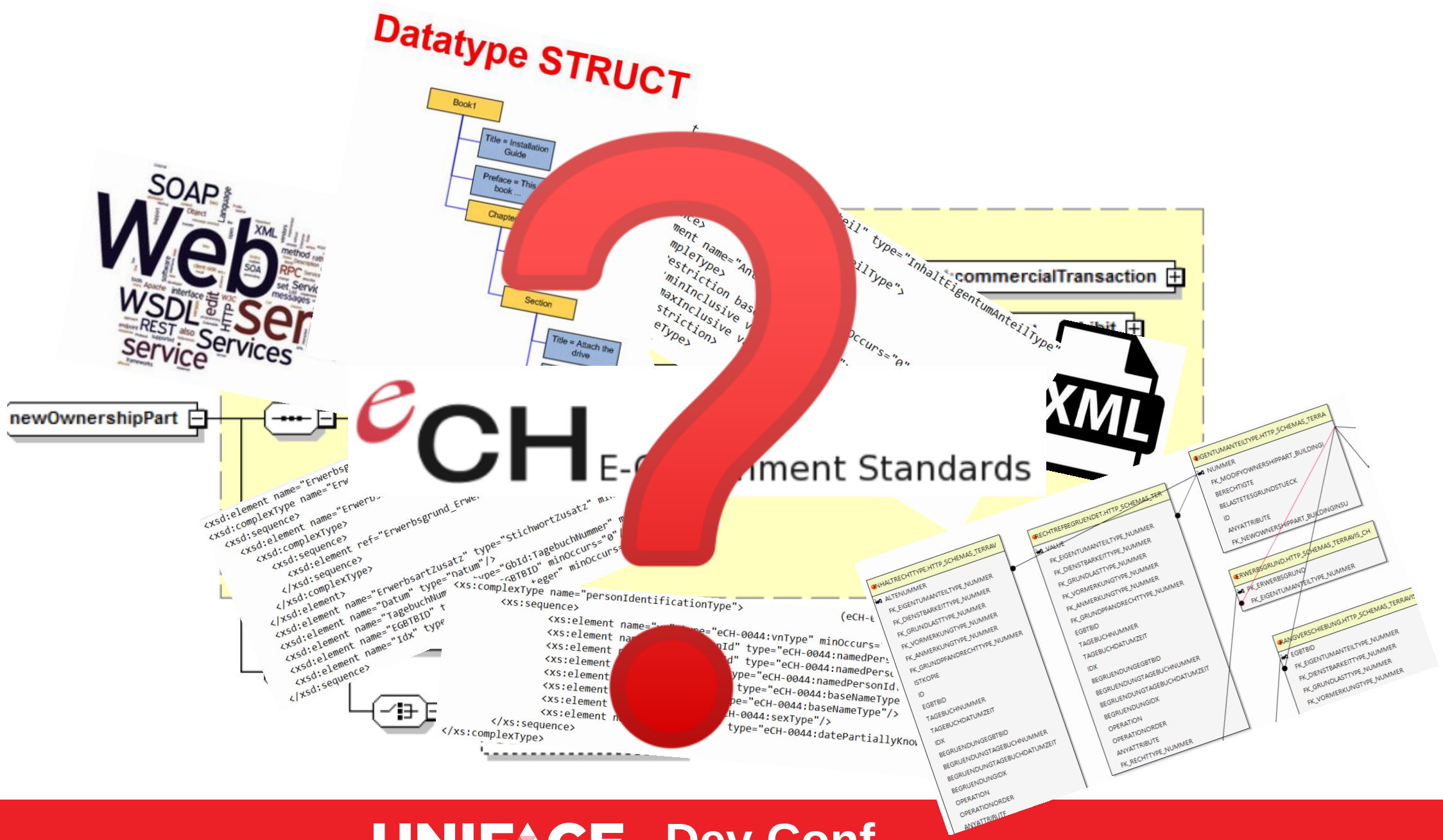
Daniel Iseli, Senior Technical Support Analyst

Agenda

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

Task

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


Datatype STRUCT
Book1
Title = Installation Guide
Preface = This book ...
Section
Title = Attach the drive
SOAP
Web
WSDL
REST
Services
service
commercialTransaction
newOwnershipPart
eCH
E-Government Standards
XML

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

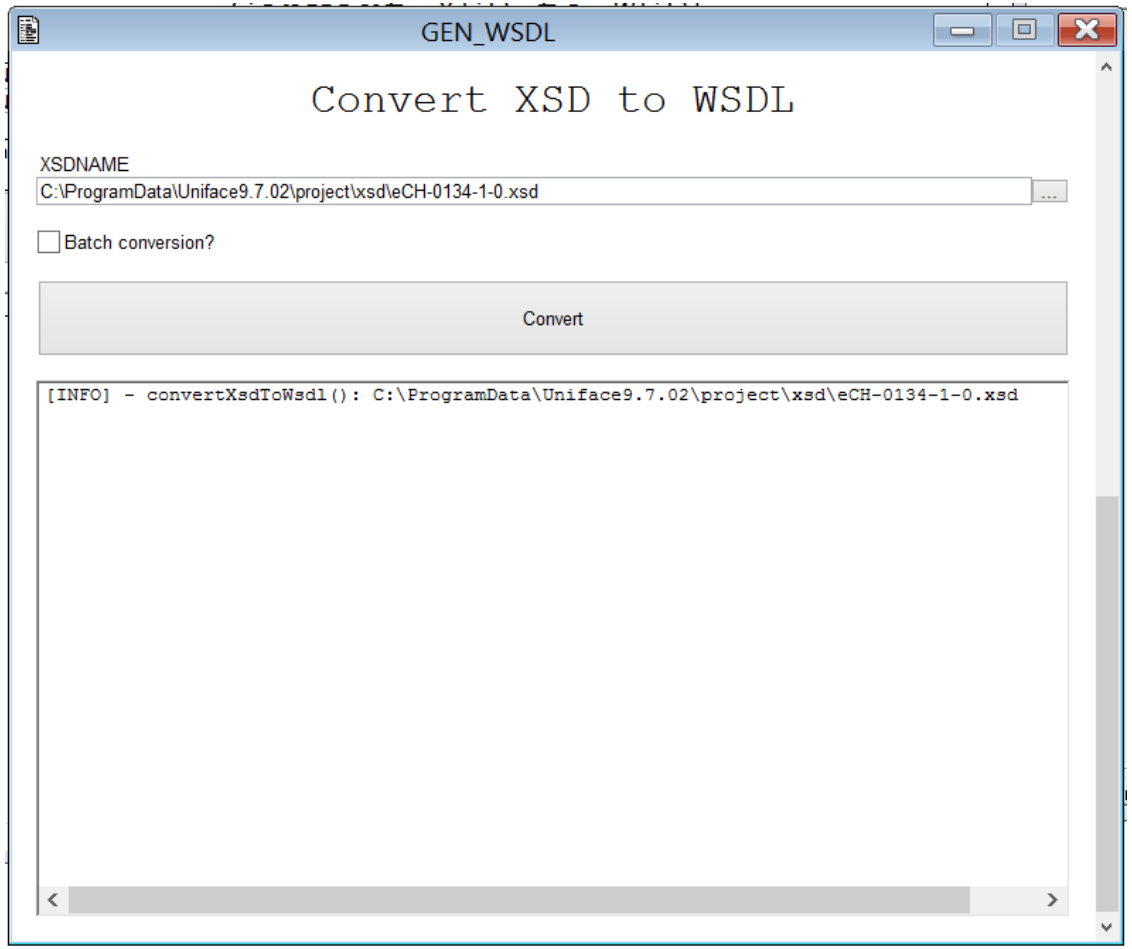
Use case newOwnershipPart (eCH-0134)

- 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)
  - gbdbs:AnmeldungType (DELIVERY_XML1)
  - eCH-0178:legalGroundExhibitType (DELIVERY_XML2)
  - gbdbs:EigentumAnteilType (DELIVERY_XML3)
  - gbdbs: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="
          <xsd:choice>
            <xsd:element na
            <xsd:element na
          </xsd:choice>
     </xsd:sequence>
   </xsd:extension>
 </xsd:complexContent>
</xsd:complexType>
```

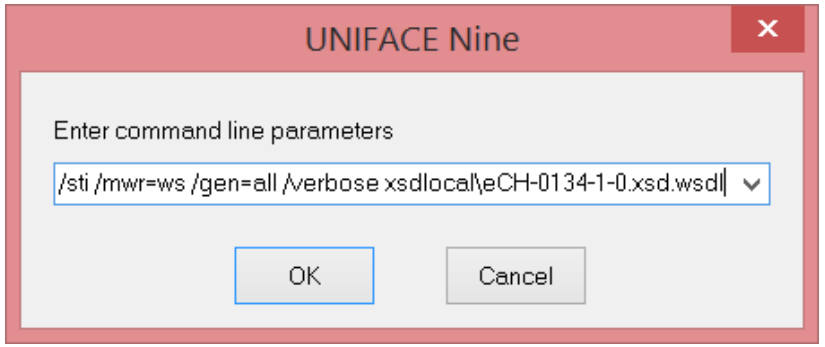
XSD

```
<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>
```

WSDL

Import des WSDL

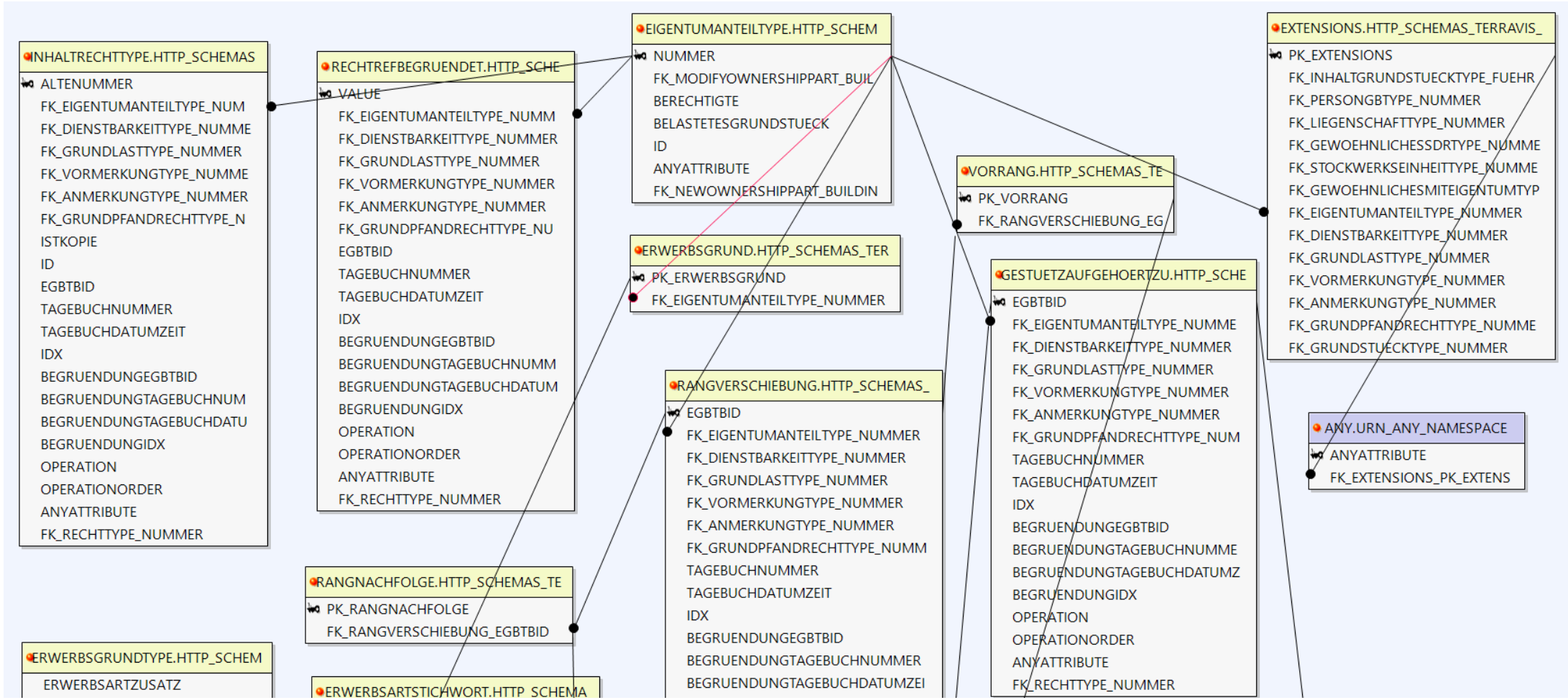
```
DRIVER_SETTINGS]
…..
SOP     U2.0
```



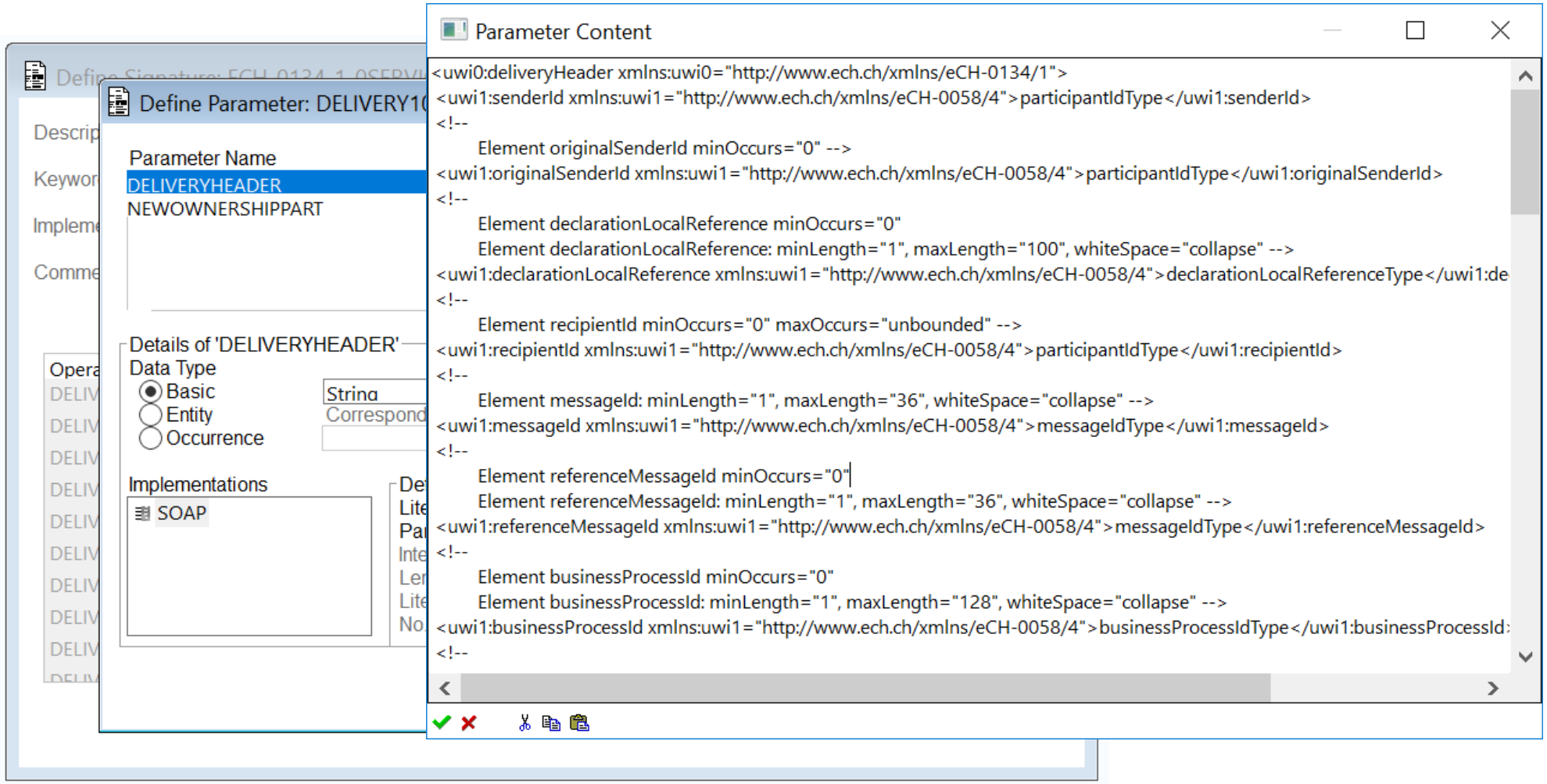
- This generates the signature of the Web service with operations and parameters.
- Models, entities and relations are created according to the XSD structures

```
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



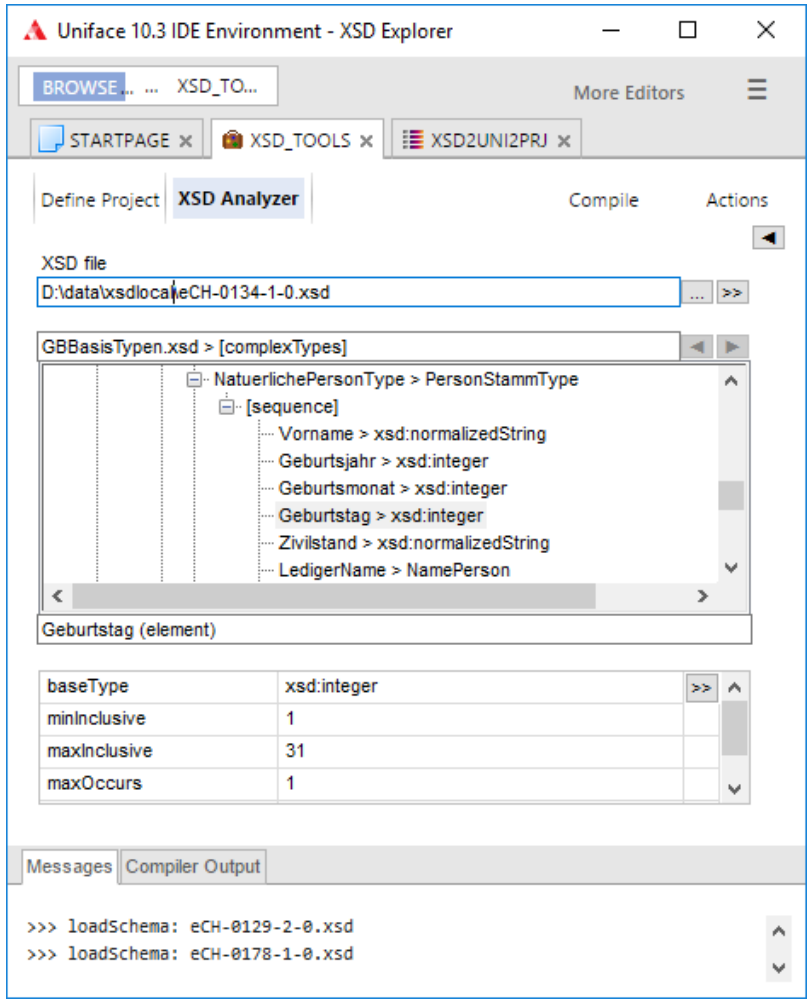
Imported Signature from the WSDL



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)



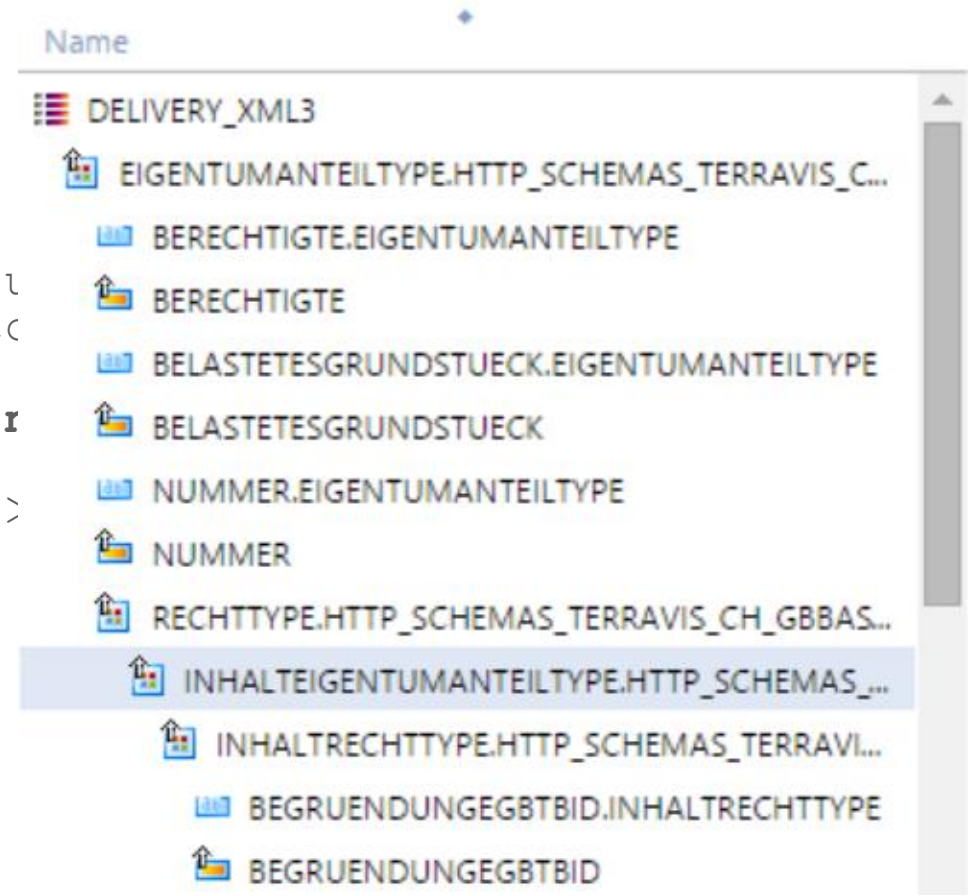
- XSD as tree structure
- Imports
- Data types
- Restrictions, Enumerations
- Choices
- Substitutions
- Mapping for generated entities
- «Enrich» generated entities

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="gbdbs: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="u
      <xsd:element name="Rangverschiebung" minOcc

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



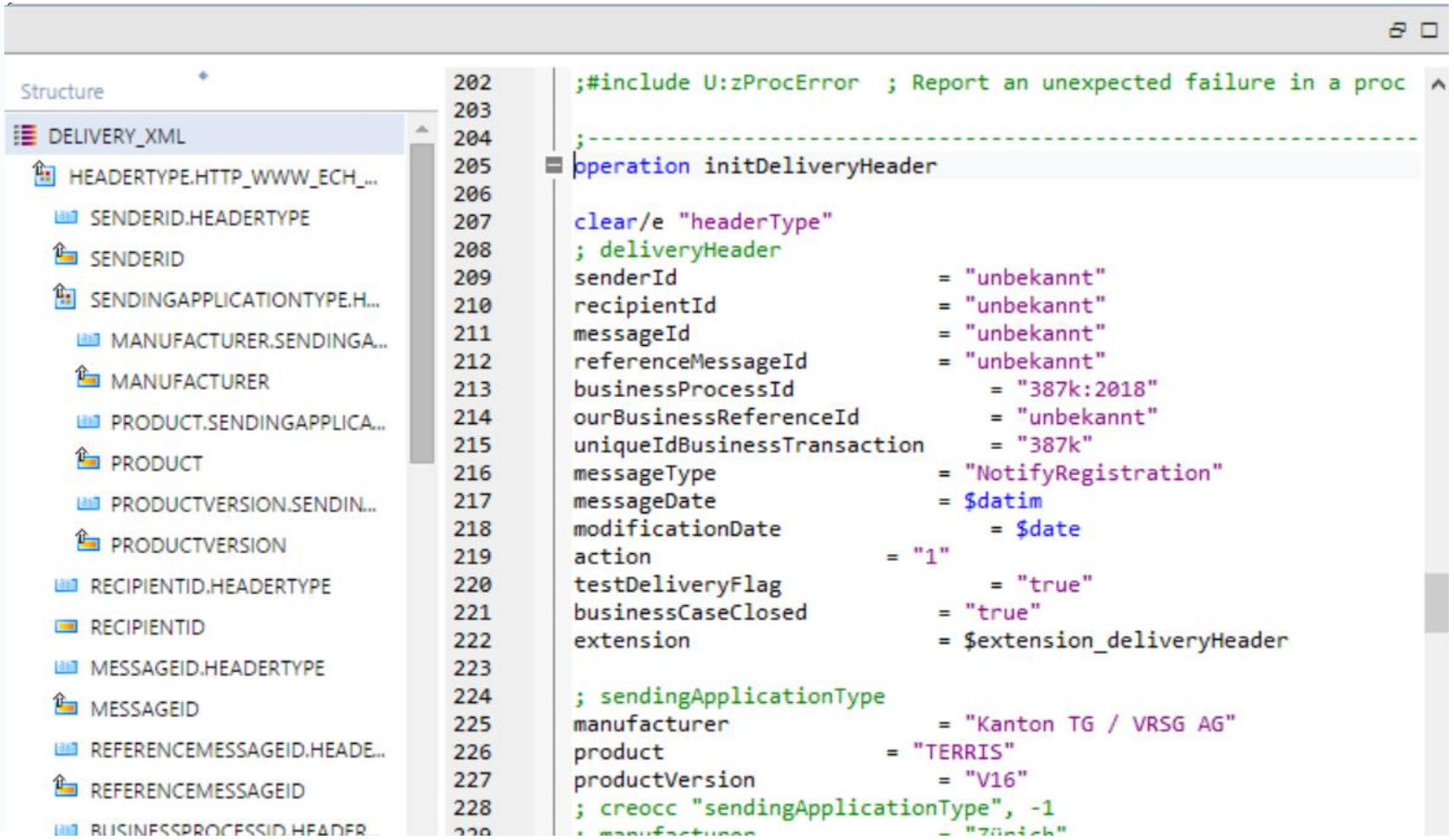
Read the required data according to the XSD

- 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



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] = "ID11111"
        [TOTALNUMBEROFPACKAGES] = 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>
  .....
```

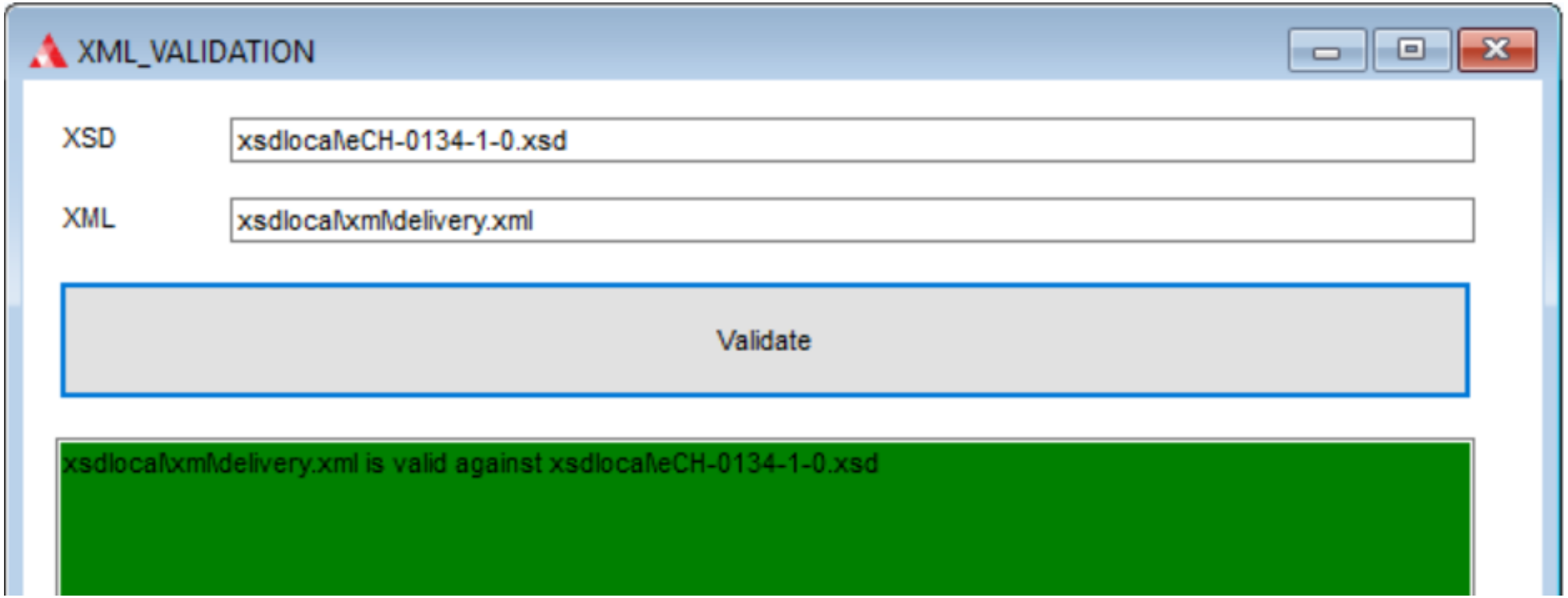
Putting it all together

```
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