Codelistenauflösung für XRechnung-Visualization

Inhalt

Autor 1
Version 1
Zweck der Skripte 1
Installation 3
Benutzung 4
Skripte erzeugen 5

Autor

Dr. Jan Thiele, openxrechnungtoolbox@gmx.net

Version

Skripte basierend auf der XRechnung-Visualization Release 2022-01-31 compatible with XRechnung 2.2.0 und sind Bestandteil der OpenXRechnungToolbox (OXT) by Dr. Jan Thiele.

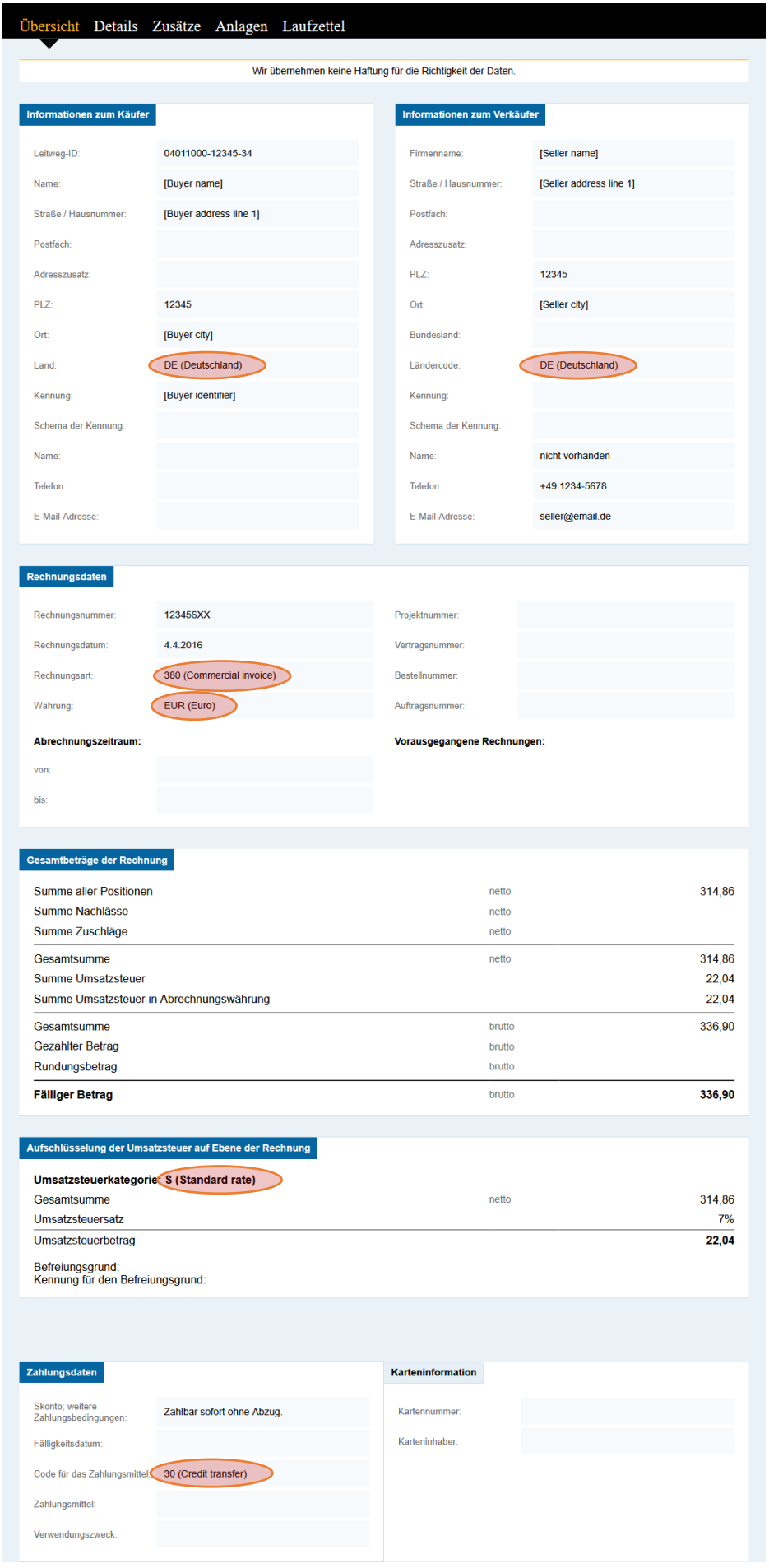
Zweck der Skripte

Mit den im GitHub Repositorium der KoSIT bereitgestellten XSL Transformatoren (https://github.com/itplr-kosit/xrechnung-visualization) kann eine elektronische Rechnung gem. der Europäischen Norm EN16931 sowie davon abgeleitete CIUS, wie z.B. XRechnung, in eine HTML-Ansicht überführt werden.

Mit den hier bereitgestellten Skripten kann innerhalb dieser XSL Transformation zusätzlich zu den enthaltenen Codelistenwerten eine Beschreibung des jeweiligen Codelistenwertes ergänzt werden. Die originalen Codewerte werden dazu nicht verändert, sondern die Beschreibung lediglich in Klammer ergänzt.

Hier ein Beispiel:

Stand: 2022-11-22



Übersicht Details Zusätze Anlagen Laufzettel

Wir übernehmen keine Haftung für die Richtigkeit der Daten.

Informationen zum Käufer

Leitweg-ID:	04011000-12345-34
Name:	[Buyer name]
Straße / Hausnummer:	[Buyer address line 1]
Postfach:	
Adresszusatz:	
PLZ:	12345
Ort:	[Buyer city]
Land:	DE (Deutschland)
Kennung:	[Buyer identifier]
Schema der Kennung:	
Name:	
Telefon:	
E-Mail-Adresse:	

Informationen zum Verkäufer

Firmenname:	[Seller name]
Straße / Hausnummer:	[Seller address line 1]
Postfach:	
Adresszusatz:	
PLZ:	12345
Ort:	[Seller city]
Bundesland:	
Ländercode:	DE (Deutschland)
Kennung:	
Schema der Kennung:	
Name:	nicht vorhanden
Telefon:	+49 1234-5678
E-Mail-Adresse:	seller@email.de

Rechnungsdaten

Rechnungsnummer:	123456XX	Projektnummer:	
Rechnungsdatum:	4.4.2016	Vertragsnummer:	
Rechnungsart:	380 (Commercial invoice)	Bestellnummer:	
Währung:	EUR (Euro)	Auftragsnummer:	

Abrechnungszeitraum:

von:	
bis:	

Vorausgegangene Rechnungen:

Gesamtbeträge der Rechnung

Summe aller Positionen	netto	314,86
Summe Nachlässe	netto	
Summe Zuschläge	netto	
Gesamtsumme	netto	314,86
Summe Umsatzsteuer		22,04
Summe Umsatzsteuer in Abrechnungswährung		22,04
Gesamtsumme	brutto	336,90
Gezahlter Betrag	brutto	
Rundungsbetrag	brutto	
Fälliger Betrag	brutto	**336,90**

Aufschlüsselung der Umsatzsteuer auf Ebene der Rechnung

Umsatzsteuerkategorie: S (Standard rate)

Gesamtsumme	netto	314,86
Umsatzsteuersatz		7%
Umsatzsteuerbetrag		**22,04**

Befreiungsgrund:
Kennung für den Befreiungsgrund:

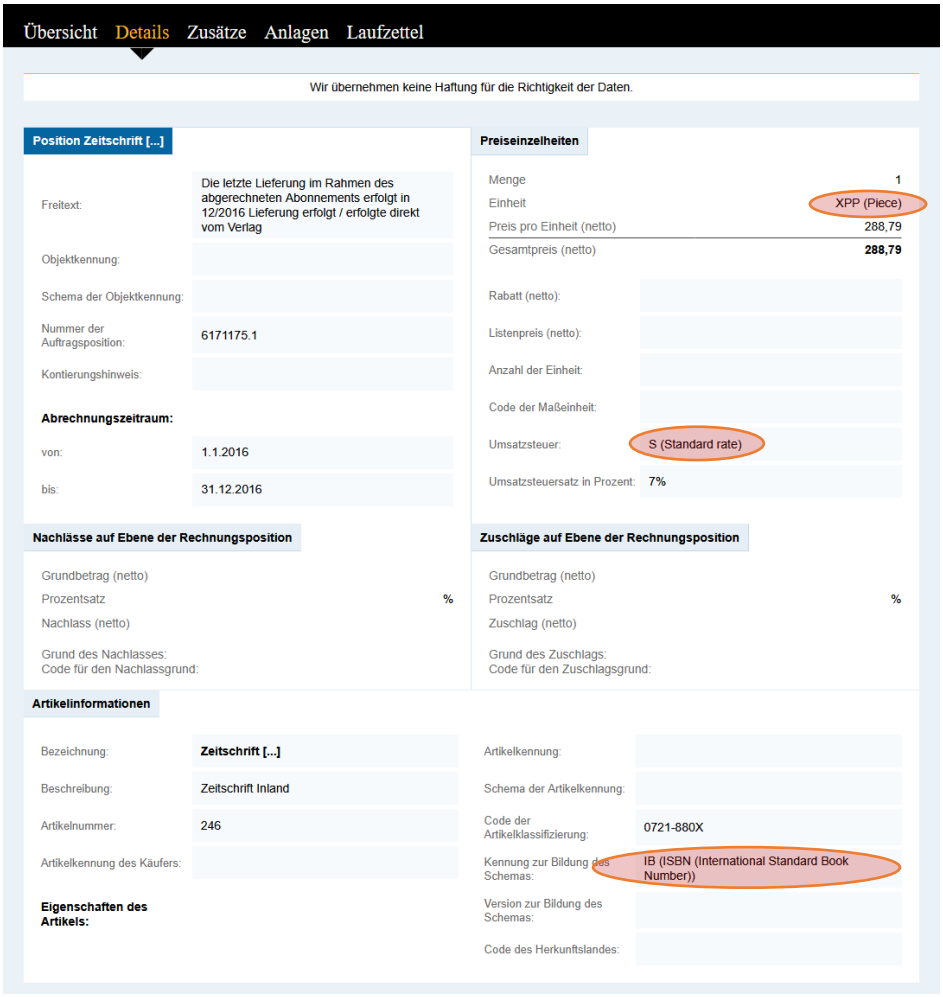
Zahlungsdaten

Skonto; weitere Zahlungsbedingungen:	Zahlbar sofort ohne Abzug.
Fälligkeitsdatum:	
Code für das Zahlungsmittel:	30 (Credit transfer)
Zahlungsmittel:	
Verwendungszweck:	

Karteninformation

Kartennummer:	
Karteninhaber:	

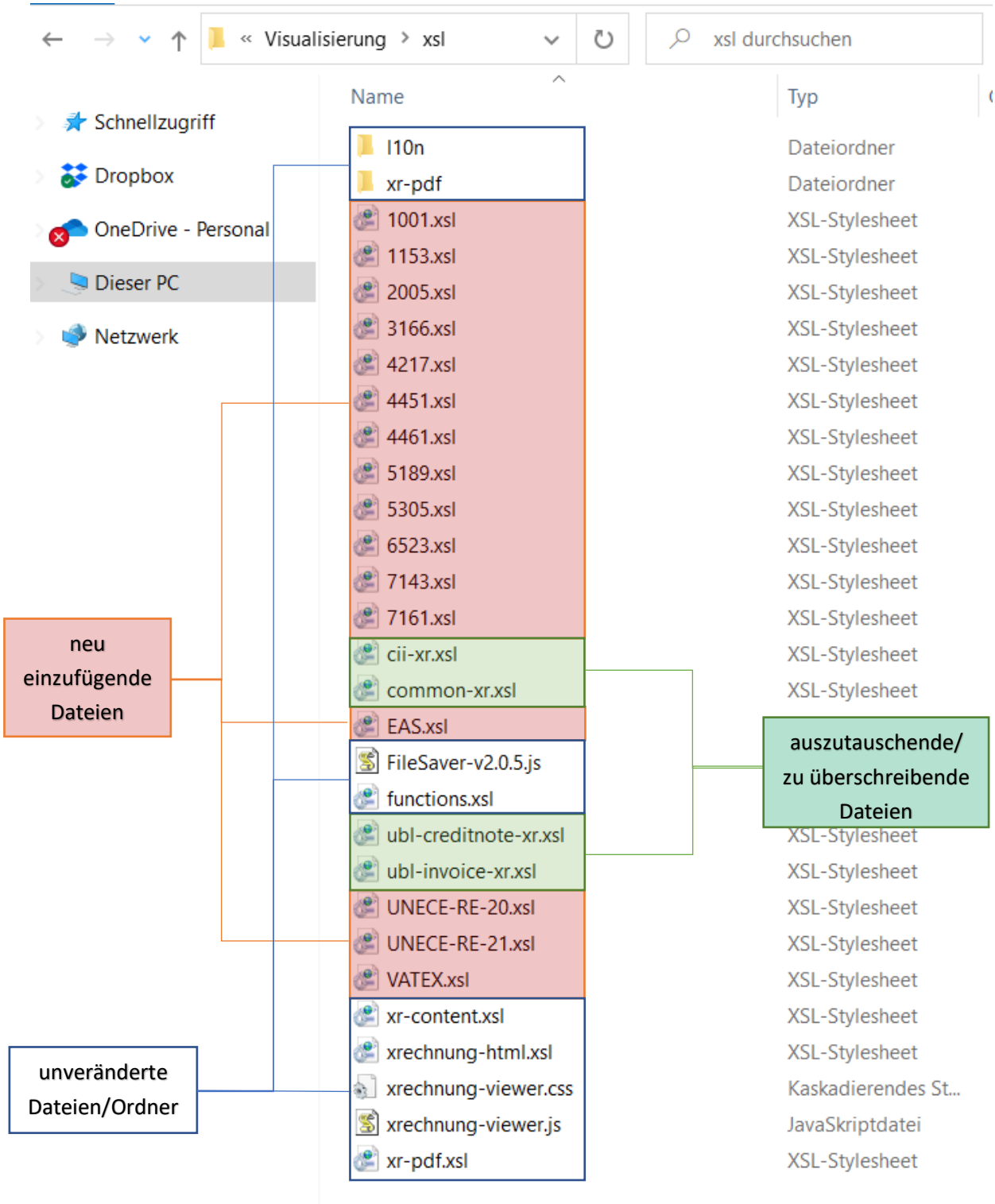
Stand: 2022-11-22



Aufgrund der Menge an Codelisten sowie der jeweils teilweise großen Menge an enthaltenen Codes geht die Auflösung der Codelisten zu Lasten der Performance. Erste Messungen haben eine Erhöhung der Laufzeit um einen Faktor von ca. 2,0 ergeben. Der Wert schwankt stark in Abhängigkeit der Nutzung von Codelisten in einer Rechnungsinstanz, da auch für optionale Informationselemente Codelisten zum Einsatz kommen.

Installation

In dem bereitgestellten Archiv <datum>_build.zip findet sich im Ordner xsl für jede Codeliste eine eigene XSL-Datei. Darüber hinaus beinhaltet das Archiv erweiterte Versionen der XSL-Dateien cii-xr.xls, ubl-creditnote-xr.xsl, ubl-invoice-xr.xsl und common-xr.xsl. Die Inhalte des zip-Archivs sind zu entpacken und die Dateien in das Verzeichnis abzulegen, in dem die XSL-Transformationsskripte der Visualisierung liegen. Dabei sind die vorhandenen XSL-Dateien cii-xr.xls, ubl-creditnote-xr.xsl, ubl-invoice-xr.xsl und common-xr.xsl zu überschreiben und die anderen xsl-Dateien hinzuzufügen.



Benutzung

Nach der Installation kann die Visualisierung wie gewohnt aufgerufen werden. Enthaltene Codelistenwerte werde sodann um die Kurzbeschreibung der Codelistenwerte in Klammern ergänzt. Diese Transformation erfolgt bei der Erzeugung der syntaxneutralen Darstellung und kann daher auch genutzt werden, wenn anschließend im zweiten Schritt keine Transformation in eine HTML-Ansicht, sondern eine alternative Transformation genutzt wird.

Ein Aufruf für eine Beispielrechnung 01.01a-INVOICE_ubl.xml in UBL-Syntax könnte z.B. so aussehen:

```
set input=testrechnungen/01.01a-INVOICE_ubl.xml

set output=testrechnungen/01.01a-INVOICE_ubl.html

java -jar lib\saxon-he-10.6.jar -s:%input% -xsl:xsl/ubl-invoice-
xr.xsl -o:%input%intermediate.xml

java -jar lib\saxon-he-10.6.jar -s:%input%intermediate.xml -
xsl:xsl/xrechnung-html.xsl -o:%output%
```

Ein Aufruf für eine Beispielgutschrift creditnote1.xml in UBL-Syntax könnte z.B. so aussehen:

```
set input=testrechnungen/creditnote1.xml

set output=testrechnungen/creditnote1.html

java -jar lib\saxon-he-10.6.jar -s:%input% -xsl:xsl/ubl-creditnote-
xr.xsl -o:%output%intermediate.xml

java -jar lib\saxon-he-10.6.jar -s:%input%intermediate.xml -
xsl:xsl/xrechnung-html.xsl -o:%output%
```

Ein Aufruf für eine Beispielrechnung 01.01a-INVOICE_uncefact.xml in UN/CEFACT CII-Syntax könnte z.B. so aussehen:

```
set input=testrechnungen/01.01a-INVOICE_uncefact.xml

set input=testrechnungen/01.01a-INVOICE_uncefact.html

java -jar lib\saxon-he-10.6.jar -s:%input% -xsl:xsl/cii-xr.xsl -
o:%input%intermediate.xml

java -jar lib\saxon-he-10.6.jar -s:%input%intermediate.xml -
xsl:xsl/xrechnung-html.xsl -o:%output%
```

Skripte erzeugen

Die XSL-Skripte im bereitgestellten zip-Archiv sind durch einen mehrstufigen, überwiegend automatisierten Erstellungsprozess erzeugt worden.

Die Ausführung erfolgt über ein ant-Script (build.xml). ant-contrib wird während der Ausführung nachgezogen. Zur Ausführung muss Apache Ant auf der Kommandozeile vorhanden sein. Der Aufruf erfolgt im entpackten Archiv <datum>_src.zip durch:

> ant

Die Erzeugung durch das ant-Script erfolgt in mehreren Schritten:

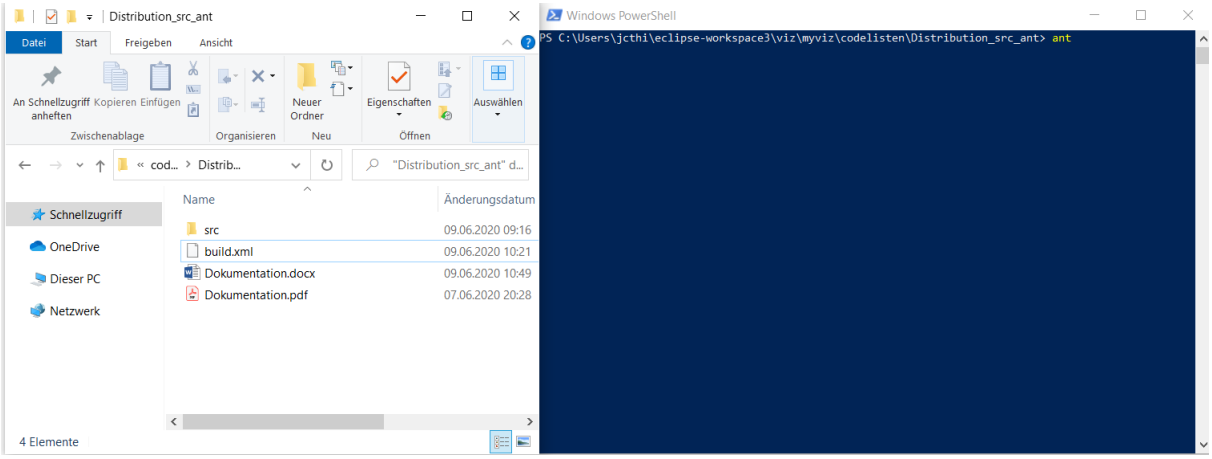
1. Codelisten im Genericode-Format werden aus dem XRepository herunterladen: https://www.xrepository.de/ (Ausnahme UN/ECE Recommendation No. 21 sowie ISO 6523. Hier wird die fertige xsl-Datei im src-Ordner bereits vorbereitet mitgeliefert. Bei UN/ECE Rec. No. 21 wurde das Template für die Codeliste UN/ECE Recommendation No. 20 eingebettet, da die Prüfung in BT-130 und BT-150 auf beide Codelisten erfolgen muss. So bleibt die

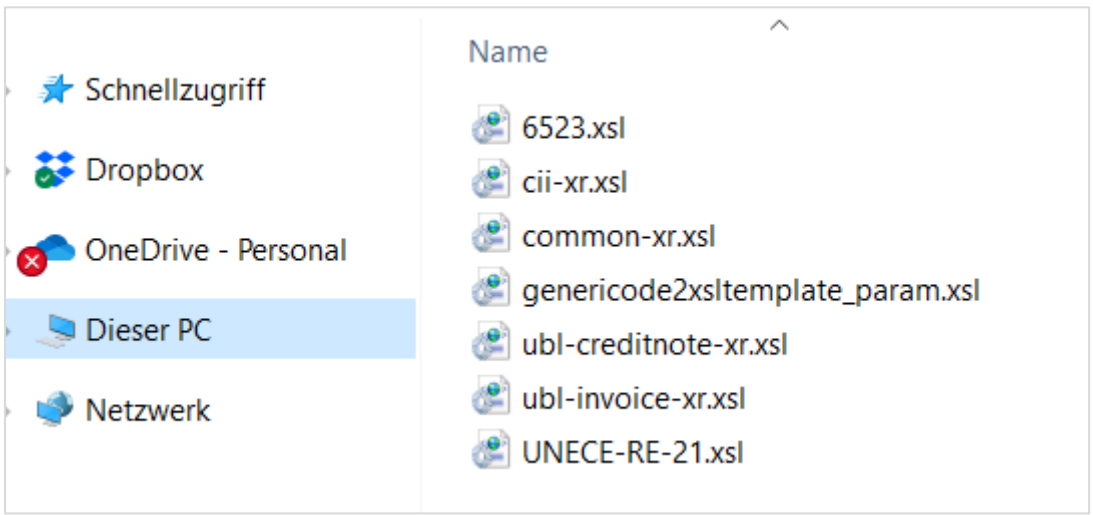
Codeliste UN/ECE Recommendation No. 20 automatisiert pflegbar. Die ISO 6523 enthält verschiedene Sonderzeichen, die vorab manuell behandelt wurden.)

2. Ant-contrib wird heruntergeladen und extrahiert.
3. Saxon-Prozessor wird heruntergeladen und extrahiert.
4. Aufruf der Transformation der Genericode-Listen in xsl-template-Definitionen.
5. XRechung-Testsuite wird heruntergeladen und extrahiert.
6. XRechnung-Visualisierung wird heruntergeladen, extrahiert, neue xsl-Dateien herüberkopiert.
7. Für die Test-Instanzen der Testsuite werden die neuen Visualisierungen prozessiert.
8. <datum>_build.zip und <datum>_src.zip werden erzeugt.
9. Während des build-Prozesses erzeugte Ordner werden gelöscht (Ausnahme: test, da derzeit keine automatisierte Überprüfung der Testergebnisse stattfindet; lib, da die enthaltenen jars noch geladen und daher nicht löschbar sind.)

Die Skripte cii-xr.xls, ubl-creditnote-xr.xsl und ubl-invoice-xr.xsl sind vorab manuell verändert worden und werden daher im src-Ordner mit ausgeliefert. Alle Änderungen sind durch einen Kommentar <!-- Begin: Jan Thiele --> und <!-- End: Jan Thiele --> ausgewiesen. Durch Import-Statements werden die Templates der Codelisten geladen. Für alle Informationselemente mit Nutzung eines Codes erfolgt eine Transformation unter Zuhilfenahme der oben erzeugten Templates.

Durch die gewählte Architektur können Aktualisierungen in den Codelisten einzeln gepflegt werden, indem die jeweilige XSL-Datei ausgetauscht wird. Gleiches gilt, wenn Aktualisierungen am Datenmodell vorgenommen wurden. Diese müssen allerdings händisch in den XSL-Dateien eingefügt werden, da dieser Prozess (bisher) nicht automatisiert wurde.





Stand: 2022-11-22