

E-Rechnung per Excel erstellen

Vorab: Ich habe nicht vor, alle Varianten abzubilden, die möglich sind, weil ich die meisten nicht benötige. Dazu konzentriere ich mich auf einige wenige Formate.

Abgebildet habe ich XRechnung (UBL), XRechnung (CII) und ZUGFeRD mit XRechnung.

Vorgestellt wird hier ein Prototyp, der dann nach eigenen optischen Wünschen modifiziert werden kann.

Der erste Schritt ist eine Oberfläche, um alle relevanten Felder einzugeben. Dabei habe ich alle Zellen mit Namen versehen, um die Eindeutigkeit durch die BT-Bezeichnungen zu gewährleisten.

Entsprechend schaut auch das Namensverzeichnis aus:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1	20250207																						
2		Rechnungsdaten						Rechnungsempfänger						Rechnungsteller					Zahlungsdaten				
3		Rechnungsnummer*	Test-x1	BT-1				Name Rechnungsempfänger	Hampel Hart	BT-44				Name*	PeLSoft	BT-27			Zahlungsart*	SEPA Überw	BT-61		
4		Rechnungsdatum*	17. Jan	BT-2				Straße*	Teststraße 1	BT-50				Straße									
5																							
6		Fälligkeitsdatum*	20. Jan	BT-9				PLZ*	12345	BT-53				PLZ*									
7		Lieferdatum	02. Jan	BT-72				Ort*	Kirchseeon	BT-52				Ort*									
8		Leitweg-ID		BT-10				Land*	Deutschland	BT-55				Land*									
9		Abrechnungszeitraum (von)		BT-73				E-Mail*	armingham	BT-49				E-Mail									
10		Abrechnungszeitraum (bis)		BT-74				Ansprechpartner		BT-56				Umsa									
11		Rechnungstyp*	Rechnung	BT-3				Telefonnummer		BT-57				Steu									
12		Projektkennung		BT-11				E-Mail		BT-58				Anspr									
13		Vertragsnummer		BT-12										Telef									
14		Bestellnummer		BT-13										E-Mail									
15		Bemerkung	keine	BT-22										Kred									
16																							
17																							
18																							
19																							
20																							
21																							
22																							
23																							
24																							
25																							
26																							
27																							
28																							
29																							
30																							
31		Aufschlüsselung Umsatzsteuerkategorien																					
32		Umsatzsteuersatz %	19	BT-119				7	BT-119														
33																							
34		Ust. Kategorie		BT-118					BT-118														
35																							
36																							
37		Gesamt netto		61%	BT-116			19,6	BT-116														

Daneben benötige ich eine Liste mit sämtlichen Kürzeln (Länder, Einheiten, ...):

	A	B	C	D
1	Kategorie	Text	Code	Sortiernummer
2	Einheit	Stück	C62	1
3	Einheit	Minuten		2
4	Einheit	Stunden	HUR	3
5	Einheit	Überstunden		4
6	Einheit	Betriebsstunden		5
7	Einheit	Tage		6
8	Einheit	Arbeitstage		7
9	Einheit	Monate		8
10	Einheit	Quartale		9
11	Einheit	Jahre		10
12	Einheit	Liter		11
13	Einheit	Milligramm		12
14	Einheit	Gramm		13
15	Einheit	Kilogramm		14
16	Einheit	Tonnen		15
17	Einheit	Kubikmeter		16
18	Einheit	Quadratmeter		17
19	Einheit	Hektar		18
20	Einheit	Millimeter		19
21	Einheit	Zentimeter		20
22	Einheit	Meter		21
23	Einheit	Zoll / Inch		22
24	Einheit	kWh		23
25	Land	Deutschland	DE	
26	Land	Österreich	AT	
27	Land	Schweiz	CH	
28	Rechnungstyp	Rechnung	380	
29	Rechnungstyp	Rechnungskorrektur		
30	Rechnungstyp	Schlussrechnungen für Bauleistungen		
31	Rechnungstyp	Selbst ausgestellte Rechnung (Gutschrift im Gutschriftsverfahren)		
32	Rechnungstyp	Teilrechnung		
33	Rechnungstyp	Teilrechnungen für Bauleistungen		
34	Rechnungstyp	Teil-Schlussrechnungen für Bauleistungen		
35	Zahlungsart	SEPA Überweisung	58	1
36	Zahlungsart	SEPA Lastschrift		2
37				

Diese habe ich nur zum Teil befüllt, mit dem, was ich brauche. Umfangreicher sind diese unter <https://easyfirma.net/e-rechnung/xrechnung/codes> zu finden.

Außerdem habe ich die ganzen Regeln in einer Tabelle:

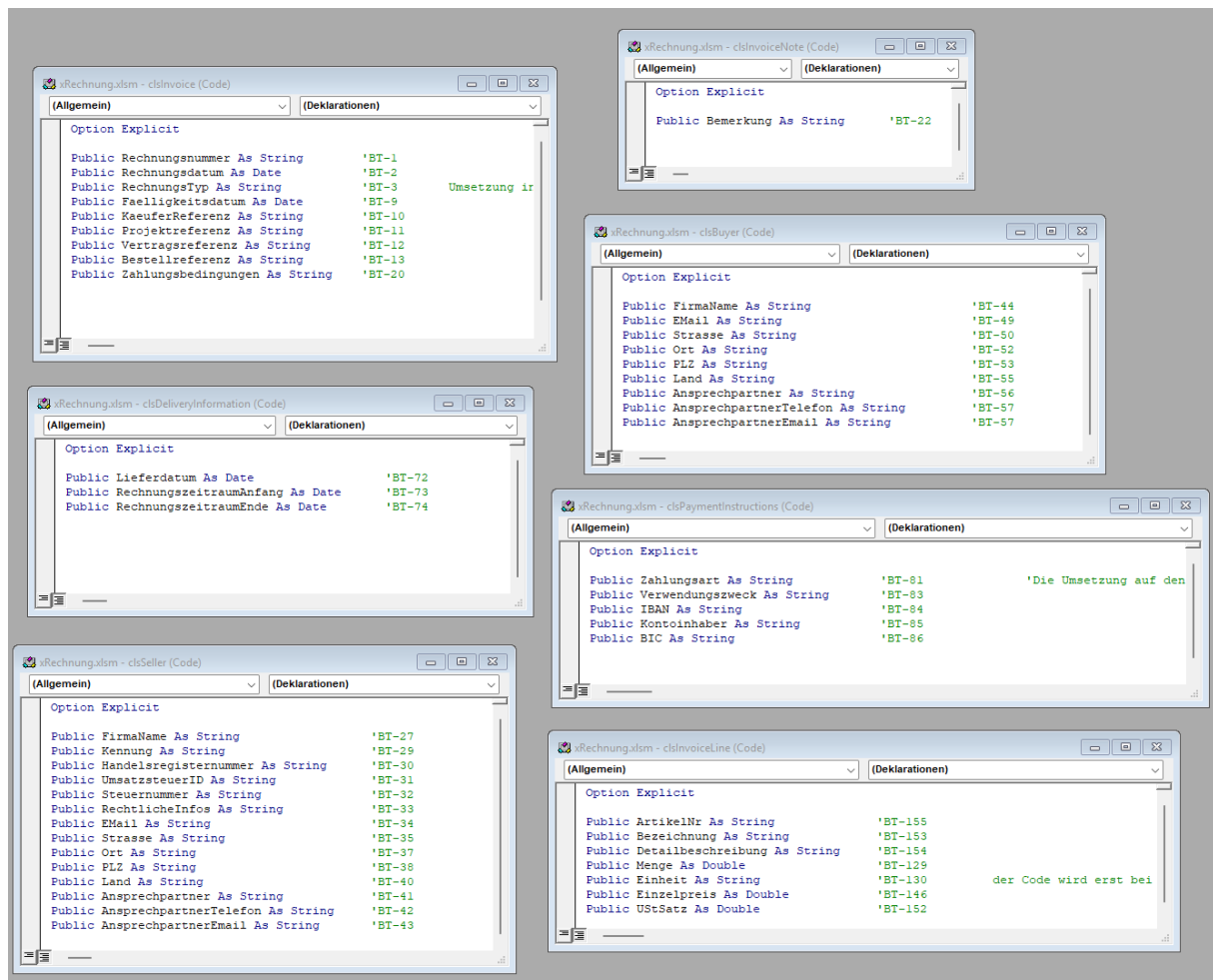
	A	B	C	D
1	ID	Beschreibung	Ziel / Kontext	Informationselement
2	BR-1	Eine Rechnung (INVOICE) muss eine Spezifikationskennung „Specification identifier“ (BT-24) enthalten.	Process control	BT-24
3	BR-2	Eine Rechnung (INVOICE) muss eine Rechnungsnummer „Invoice number“ (BT-1) enthalten.	Invoice	BT-1
4	BR-3	Eine Rechnung (INVOICE) muss ein Rechnungsdatum „Invoice issue date“ (BT-2) enthalten.	Invoice	BT-2
5	BR-4	Eine Rechnung (INVOICE) muss einen Rechnungstyp-Code „Invoice type code“ (BT-3) enthalten.	Invoice	BT-3
6	BR-5	Eine Rechnung (INVOICE) muss einen Währungs-Code „Invoice currency code“ (BT-5) enthalten.	Invoice	BT-5
7	BR-6	Eine Rechnung (INVOICE) muss den Verkäufersnamen „Seller name“ (BT-27) enthalten.	Seller	BT-27
8	BR-7	Eine Rechnung (INVOICE) muss den Erwerbersnamen „Buyer name“ (BT-44) enthalten.	Buyer	BT-44
9	BR-8	Eine Rechnung (INVOICE) muss die postalische Anschrift des Verkäufers „SELLER POSTAL ADDRESS“ (BG-5) enthalten.	Seller	BT-5
10	BR-9	Die postalische Anschrift des Verkäufers „SELLER POSTAL ADDRESS“ (BG-5) muss einen Verkäufer-Ländercode „Seller country code“ (BT-40) enthalten.	Seller Postal Address	BT-40
11	BR-10	Eine Rechnung (INVOICE) muss die postalische Anschrift des Erwerbers „BUYER POSTAL ADDRESS“ (BG-8) enthalten.	Buyer	BT-8
12	BR-11	Die postalische Anschrift des Erwerbers „BUYER POSTAL ADDRESS“ (BG-8) muss einen Erwerber-Ländercode „Buyer country code“ (BT-55) enthalten.	Buyer Postal Address	BT-55
13	BR-12	Eine Rechnung (INVOICE) muss die Summe der Nettobeträge der Rechnungspositionen „Sum of Invoice line net amount“ (BT-106) enthalten.	Document totals	BT-106
14	BR-13	Eine Rechnung (INVOICE) muss den Gesamtbetrag der Rechnung ohne Umsatzsteuer „Invoice total amount without VAT“ (BT-109) enthalten.	Document totals	BT-109
15	BR-14	Eine Rechnung (INVOICE) muss den Gesamtbetrag der Rechnung mit Umsatzsteuer „Invoice total amount with VAT“ (BT-112) enthalten.	Document totals	BT-112
16	BR-15	Eine Rechnung (INVOICE) muss den ausstehenden Betrag „Amount due for payment“ (BT-115) enthalten.	Document totals	BT-115
17	BR-16	Eine Rechnung (INVOICE) muss mindestens eine Rechnungsposition „INVOICE LINE“ (BG-25) enthalten.	Invoice	BT-25
18	BR-17	Eine Rechnung (INVOICE) muss den Namen des Zahlungsempfängers „Payee name“ (BT-59) enthalten, wenn sich der Zahlungsempfänger „PAYEE“ (BG-10) vom Verkäufer „SELLER“ (BG-4) unterscheidet.	Payee	BT-59
19	BR-18	Eine Rechnung (INVOICE) muss den Namen des Steuervertreeters des Verkäufers „Seller tax representative name“ (BT-62) enthalten, wenn der Verkäufer „SELLER“ (BG-4) einen Steuervertreter „SELLER TAX REPRESENTATIVE PARTY“ (BG-11) hat.	Seller tax representative	BT-62
20	BR-19	Eine Rechnung (INVOICE) muss die postalische Anschrift des Steuervertreeters „SELLER TAX REPRESENTATIVE POSTAL ADDRESS“ (BG-12) enthalten, wenn der Verkäufer „SELLER“ (BG-4) einen Steuervertreter „SELLER TAX REPRESENTATIVE PARTY“ (BG-11) hat.	Seller tax representative	BT-12
21	BR-20	Die postalische Anschrift des Steuervertreeters des Verkäufers „SELLER TAX REPRESENTATIVE POSTAL ADDRESS“ (BG-12) muss einen Steuervertreter-Ländercode „Tax representative country code“ (BT-69) enthalten, wenn der Verkäufer „SELLER“ (BG-4) einen Steuervertreter „SELLER TAX REPRESENTATIVE PARTY“ (BG-11) hat.	Seller tax representative postal address	BT-69
22	BR-21	Jede Rechnungsposition „INVOICE LINE“ (BG-25) muss eine eindeutige Bezeichnung „Invoice line identifier“ (BT-126) haben.	Invoice Line	BT-126
23	BR-22	Jede Rechnungsposition „INVOICE LINE“ (BG-25) muss die Menge der in der betreffenden Position in Rechnung gestellten Waren oder Dienstleistungen als Einzelposten „Invoiced quantity“ (BT-129) enthalten.	Invoice Line	BT-129
24	BR-23	Jede Rechnungsposition „INVOICE LINE“ (BG-25) muss eine Einheit zur Mengenangabe „Invoiced quantity unit of measure code“ (BT-130) enthalten.	Invoice Line	BT-130
25	BR-24	Jede Rechnungsposition „INVOICE LINE“ (BG-25) muss den Nettobetrag der Rechnungsposition „Invoice line net amount“ (BT-131) enthalten.	Invoice Line	BT-131
26	BR-25	Jede Rechnungsposition „INVOICE LINE“ (BG-25) muss den Namen des Postens „Item name“ (BT-153) enthalten.	Item information	BT-153
27	BR-26	Jede Rechnungsposition „INVOICE LINE“ (BG-25) muss den Preis des Postens, ohne Umsatzsteuer, nach Abzug des für diese Rechnungsposition geltenden Rabatts „Item net price“ (BT-146) beinhalten.	Price details	BT-146
28	BR-27	Der Artikel-Nettobetrag „Item net price“ (BT-146) darf nicht negativ sein.	Item net price	BT-146
29	BR-28	Der Einheitspreis ohne Umsatzsteuer vor Abzug des Postenpreistrabatts einer Rechnungsposition „Item gross price“ (BT-148) darf nicht negativ sein.	Price details	BT-148
30	BR-29	Wenn Start- und Enddatum des Rechnungszeitraums gegeben sind, muss das Enddatum „Invoicing period end date“ (BT-74) nach dem Startdatum „Invoicing period start date“ (BT-73) liegen oder mit	Invoicing Period	BT-74

Der VBA-Teil ist mehrteilig:

- Das Modul „mdlDemo“ sorgt dafür, dass die Daten aus dem Tabellenblatt in Variablen übernommen werden.
- Das Modul „mdlXRechnung“ erstellt die Rechnung und sorgt für Plausibilitätstests.
- Die Klassenmodule bündeln die ganzen Variablen, so dass sie logisch zusammengefasst werden können.

Die Struktur den Klassen habe ich an den XRechnung-Aufbau angelehnt, der hier zu finden ist:

<https://xeinkauf.de/app/uploads/2024/07/302-XRechnung-2024-06-20.pdf> , auf Seite 26.



Im mdlDemo werden die Eingaben in die Variablen geschrieben, in der einzigen Prozedur, die der Anwender startet: „Demo“:

```
xRechnung.xlsm - mdlDemo (Code)
(Allgemein)

Option Explicit

Dim Kaeufer As clsBuyer
Dim LieferInfo As clsDeliveryInformation
Dim Rechnung As clsInvoice
Dim RechnungInfo As clsInvoiceNote
Dim Zahlung As clsPaymentInstructions
Dim Verkaeufer As clsSeller
Dim Artikel() As clsInvoiceLine

Sub Demo ()

Dim Zeile As Long
Dim ArtikelPos As Integer
Dim Fehler As String

'Variablen initialisieren
Set Kaeufer = New clsBuyer
Set LieferInfo = New clsDeliveryInformation
Set Rechnung = New clsInvoice
Set RechnungInfo = New clsInvoiceNote
Set Zahlung = New clsPaymentInstructions
Set Verkaeufer = New clsSeller
'Artikel kommt später

'Variablen füllen
Rechnung.Rechnungsnummer = Range("BT_1")
Rechnung.Rechnungsdatum = Range("BT_2")
Rechnung.Faelligkeitsdatum = Range("BT_9")
Rechnung.KaeuferReferenz = Range("BT_10")
Rechnung.RechnungsTyp = Range("BT_3")
Rechnung.Projektreferenz = Range("BT_11")
Rechnung.Vertragsreferenz = Range("BT_12")
Rechnung.Bestellreferenz = Range("BT_13")
Rechnung.Zahlungsbedingungen = Range("BT_20")

LieferInfo.Lieferdatum = Range("BT_72")
LieferInfo.RechnungszeitraumAnfang = Range("BT_73")
LieferInfo.RechnungszeitraumEnde = Range("BT_74")

RechnungInfo.Bemerkung = Range("BT_22")

Kaeufer.FirmaName = Range("BT_44")
Kaeufer.Strasse = Range("BT_50")
Kaeufer.PLZ = Range("BT_53")
Kaeufer.Ort = Range("BT_52")
```

Am Ende wird ein Funktionsaufruf gestartet, der sowohl die Rechnungserstellung als auch die Rückgabe von Fehlermeldungen ermöglicht:

```

'Rechnung erstellen
Fehler = RechnungErstellen("XRechnung CII", Rechnung, RechnungInfo, Kaeufer, Verkaeuffer, LieferInfo, Zahlung, Artikel, Environ("Temp"), "Test Rechnung")

If Fehler = "" Then
    MsgBox ("fertig")
Else
    SetClipboard (Fehler)
    MsgBox ("Es sind Fehler aufgetreten. Diese sind auch in die Zwischenablage kopiert." & vbCrLf & vbCrLf & Fehler)
End If
End Sub

```

Die Funktion „RechnungErstellen“ ist dann schon in mdIXRechnung.

„RechnungErstellen“ verteilt die einzelnen Aufgaben:

```

Function RechnungErstellen(RechnungTyp As String, xRechnung As clsInvoice, xRechnungInfo As clsInvoiceNote, xKaeufer As clsBuyer, xVerkaeuffer As clsSeller, xLieferInfo As clsDeliveryInforma
'übernimmt die Daten und verteilt die Erstellung je nach gewünschtem Ergebnis
'die Sache mit der Shell und dem Warten ist von https://stackoverflow.com/questions/17956651/execute-a-command-in-command-prompt-using-excel-vba
Dim wsh As Object
Dim waitOnReturn As Boolean
Dim windowStyle As Integer

Dim AttachString As String
Dim RechnungNameXML As String 'Der Rechnungsname vom XML-Teil wie er dann beim ZUGFeRD als Anhang auch verwendet wird

Select Case RechnungTyp
    Case Is = "XRechnung CII"
        RechnungErstellen = XRechnungErstellenCII(xRechnung, xRechnungInfo, xKaeufer, xVerkaeuffer, xLieferInfo, xZahlung, xArtikel(), SpeicherPfad, Rechnungsname)
    Case Is = "XRechnung UBL"
        RechnungErstellen = XRechnungErstellenUBL(xRechnung, xRechnungInfo, xKaeufer, xVerkaeuffer, xLieferInfo, xZahlung, xArtikel(), SpeicherPfad, Rechnungsname)
    Case Is = "ZUGFeRD (XRechnung)"
        'die beiden Einzelteile XML und PDF kommen ins TEMP, das Ergebnis ins gewählte Verzeichnis
        RechnungNameXML = "ReNr " & xRechnung.Rechnungsnummer

        RechnungErstellen = XRechnungErstellenCII(xRechnung, xRechnungInfo, xKaeufer, xVerkaeuffer, xLieferInfo, xZahlung, xArtikel(), Environ("Temp"), RechnungNameXML)

        If RechnungErstellen = "" Then 'wenn der XRechnungsteil korrekt erstellt wurde, kann auch das PDF erstellt werden
            'speichern als PDF/A
            ActiveSheet.ExportAsFixedFormat Type:=xlTypePDF, Filename:= " " & Environ("Temp") & "\" & xRechnung.Rechnungsnummer & " Temp.pdf", Quality:=xlQualityStandard, _
                IncludeDocProperties:=True, IgnorePrintAreas:=False, OpenAfterPublish:=False

            'den Anhang dazu hängen
            Set wsh = VBA.CreateObject("WScript.Shell")
            waitOnReturn = True
            windowStyle = 1

            AttachString = "pdftk.exe "" " & Environ("Temp") & "\" & xRechnung.Rechnungsnummer & " Temp.pdf"" attach_files "" " & Environ("Temp") & "\" & RechnungNameXML & ".xml"" output '
            wsh.Run AttachString, windowStyle, waitOnReturn

        End If
    Case Else
        RechnungErstellen = "RechnungTyp ist unbekannt"
    End Select
End Function

```

Wichtig ist im Fall von ZUGFeRD, dass hier eine weitere Komponente vorhanden sein muss: Der PDFtk-Server <https://www.pdfabs.com/tools/pdftk-server/> . Es gibt kaum Möglichkeiten, ein PDF/A mit einem Anhang zu versehen (also genau die ZUGFeRD-Kombination). Außerdem wollte ich diesen Schritt unbedingt auch automatisieren. Der PDFtk-Server ist ein kleines Programm, das installiert wird und dann per Befehlszeile gestartet werden kann. Diese Ausführung in der Shell mit Befehlszeile wird auch hier verwendet, wie im VBA-Code zu sehen ist.

Der XML-Text wird dann ziemlich primitiv zusammen geschrieben:

```

xRechnung.xlsm - mdIXRechnung (Code)
[Allgemein]
XRechnungErstellenCII

RechnungText = RechnungText & vbCrLf & MTab5 & "<ram:EmailURIUniversalCommunication>"
RechnungText = RechnungText & vbCrLf & MTab6 & "<ram:URIID>" & xVerkaeuer.AnsprechpartnerEmail & "</ram:URIID>"
RechnungText = RechnungText & vbCrLf & MTab5 & "</ram:EmailURIUniversalCommunication>"
End If

RechnungText = RechnungText & vbCrLf & MTab4 & "</ram:DefinedTradeContact>"
RechnungText = RechnungText & vbCrLf & MTab4 & "<ram:PostalTradeAddress>"
RechnungText = RechnungText & vbCrLf & MTab5 & "<ram:PostcodeCode>" & xVerkaeuer.PLZ & "</ram:PostcodeCode>"
RechnungText = RechnungText & vbCrLf & MTab5 & "<ram:LineOne>" & xVerkaeuer.Strasse & "</ram:LineOne>"
RechnungText = RechnungText & vbCrLf & MTab5 & "<ram:CityName>" & xVerkaeuer.Ort & "</ram:CityName>"

Code = CodeFinden("Land", xVerkaeuer.Land, "BR-9")
RechnungText = RechnungText & vbCrLf & MTab5 & "<ram:CountryID>" & Code & "</ram:CountryID>"
RechnungText = RechnungText & vbCrLf & MTab4 & "</ram:PostalTradeAddress>"

RechnungText = RechnungText & vbCrLf & MTab4 & "<ram:URIUniversalCommunication>"
RechnungText = RechnungText & vbCrLf & MTab5 & "<ram:URIID schemeID='EM'>" & xVerkaeuer.Email & "</ram:URIID>"
RechnungText = RechnungText & vbCrLf & MTab4 & "</ram:URIUniversalCommunication>"

If xVerkaeuer.UmsatzsteuerID <> "" Or xVerkaeuer.Steuernummer <> "" Then
    RechnungText = RechnungText & vbCrLf & MTab4 & "<ram:SpecifiedTaxRegistration>"

    If xVerkaeuer.UmsatzsteuerID <> "" Then
        RechnungText = RechnungText & vbCrLf & MTab5 & "<ram:ID schemeID='VA'>" & xVerkaeuer.UmsatzsteuerID & "</ram:ID>"
    End If

    If xVerkaeuer.Steuernummer <> "" Then
        RechnungText = RechnungText & vbCrLf & MTab5 & "<ram:ID schemeID='FC'>" & xVerkaeuer.Steuernummer & "</ram:ID>"
    End If

    RechnungText = RechnungText & vbCrLf & MTab4 & "</ram:SpecifiedTaxRegistration>"
End If

RechnungText = RechnungText & vbCrLf & MTab3 & "</ram:SellerTradeParty>"

'Käufer
RechnungText = RechnungText & vbCrLf & MTab3 & "<ram:BuyerTradeParty>"
RechnungText = RechnungText & vbCrLf & MTab4 & "<ram:ID>Kd-Nr. 0015</ram:ID>"
RechnungText = RechnungText & vbCrLf & MTab4 & "<ram:Name>" & xKaeufer.FirmaName & "</ram:Name>"
RechnungText = RechnungText & vbCrLf & MTab4 & "<ram:SpecifiedLegalOrganization>"

```

Hier müssen leere Felder heraus gelassen werden, und die Umsetzung der Eingaben in die Codes (z.B. „Stück“ → „C62“) werden über eine Funktion gesteuert.

Dann gibt es noch ein paar Nebenbedingungen, zum Beispiel muss der XML-Teil zwingend in UTF-8 codiert sein, was VBA standardmäßig nicht macht. Deshalb eine etwas aufwändigere Variante:

```

'Datei erstellen
Pfad = SpeicherPfad & "\"
DatName = RechnungName & ".xml"

If Fehler = "" Then
    Set AdoText = CreateObject("ADODB.Stream")
    AdoText.Charset = "UTF-8"

    AdoText.Mode = adModeReadWrite
    AdoText.Type = adTypeText
    AdoText.Open
    AdoText.WriteText (RechnungText)
    AdoText.SaveToFile Pfad & DatName, adSaveCreateOverWrite
    AdoText.Close
    Set AdoText = Nothing
End If

```

Den Aufbau mit den Modulen habe ich deshalb so gestaltet, dass im Zweifel nur einzelne Module ausgetauscht bzw. angepasst werden müssen. Wenn ich im Erstellungsteil mdIXRechnung einen Fehler habe, reicht es, nur dieses Modul durch das neue, korrigierte zu ersetzen, der Rest kann dann bleiben.

Beim Aufbau habe ich auch im Hinterkopf gehabt, dass ich mit relativ wenigen Modifikationen die Basis auf Access umstellen kann. Gerade die Tabelle mit den Codes (Einheiten, Länder, ...) ist deswegen so gestaltet, dass ich sie direkt in eine (1) Access-Tabelle übernehmen kann (oder umgekehrt).

Am Ende sollte eine korrekte E-Rechnung rauskommen. Ich muss aber jede testen, sowohl den XML-Teil, als auch das ZUGFeRD-Konglomerat, und dazu noch inhaltlich. Für den XML-Teil verwende ich gerne <https://erechnungsvalidator.service-bw.de/> , für ZUGFeRD <https://erechnungs-validator.de/> . Dazu lasse ich mir den XML-Teil noch in was besser lesbares umwandeln, zum Beispiel mit <https://www.papierkram.de/e-rechnung/e-rechnung-viewer-kostenlos> . Dadurch kommen auch inhaltliche Fehler raus (ich hatte als Beispiel mal den Ansprechpartner vom Ersteller zusätzlich als Ansprechpartner beim Rechnungsempfänger drin). Bei den Validatoren hatten wir mal einen Fehler, der nicht zu finden war. Der Anbieter vom Validator war aus Spanien und hat wahrscheinlich die spanischen Regeln zugrunde gelegt, obwohl die Oberfläche in Deutsch gehalten war. Es ist also ratsam, auch die Ergebnisse vom Validator zu validieren (schöne Konstruktion) und sich im Zweifel auf deutsche Webseiten zu konzentrieren.

Das mit den Validatoren hat halt den Nachteil, dass du geschäftliche auf eine eventuell nicht so zuverlässige Webseite schicken musst. Aber es geht derzeit nicht anders (wenigstens nicht kostenlos).

Zum Erstellen von E-Rechnungen ist [PDF24](#) sehr gut, weil dort praktisch alle Konstellationen zu bekommen sind (im Gegensatz zu anderen, wie z.B. Buhl, wo die Hälfte fehlt). Außerdem hat PDF24 die Möglichkeit, das Tool am eigenen Rechner zu installieren und umgeht dann die Problematik der fremden Webseiten. PDF24 habe ich dann auch später verwendet, um zu sehen, wie die bestimmte Konstellationen umsetzen, die bei mir Fehler produzieren, bei PDF24 aber nicht.

Noch ein Hinweis: Einiges im VBA-Code ist sehr schlampig und nicht gerade sauber und robust aufgebaut. Es ist ein Tool für den Eigengebrauch (und für genau einen weiteren Anwender), und deshalb sind auch viele Teile eher Q&D (quick and dirty) umgesetzt. Ich kann aber auch sauber, wenn's sein soll.

Bei Fragen einfach fragen, ich freue mich über eure Nachrichten: post@peter-listmann.de .

Schöne Grüße

Peter