

E-Rechnung / ZUGFeRD mit PowerBuilder

Welche E-Rechnungs-Formate gibt es?

Wo liegen die Herausforderungen?

Wie konnten sie gelöst werden?

Ein Erfahrungsbericht.

Agenda

1

Warum E-Rechnung und seit wann?

Pflichten, Fristen, Hintergründe

2

Formate

XRechnung, ZUGFeRD

3

Und wie jetzt umsetzen...?

Vorüberlegungen & Herangehensweise



4

Realisierung in PowerBuilder

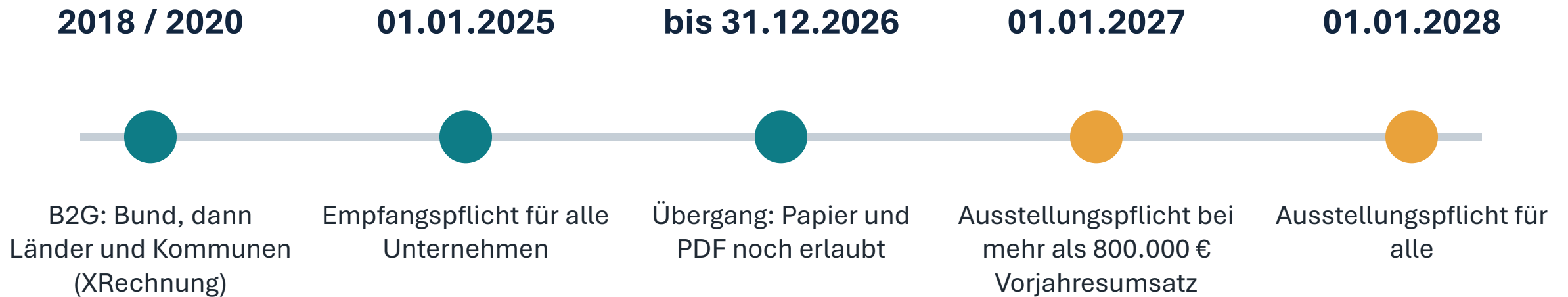
E-Rechnungs-Modul, Daten-Mapping

5

PDF, Prüfung & Validierung

ZUGFeRD-PDF, Validierungs-Tools

Seit wann? Die E-Rechnung kommt in Stufen:



Ausnahmen: Kleinbetragsrechnungen bis 250 €, Fahrausweise, Kleinunternehmer (nur Empfang)

Grundlage: § 14 UStG · BMF-Schreiben vom 15.10.2024 und 15.10.2025

Warum? Vier Haupt-Gründe:

Bekämpfung von Steuerbetrug

Strukturierte Daten erlauben automatische Prüfungen.

Ziel: weniger Umsatzsteuerbetrug und eine kleinere „VAT Gap“.

Kostensenkung und Effizienzsteigerung

Papierlos, kein Abtippen, keine OCR-Fehler, weniger Medienbrüche, schnellere und kostengünstigere Prozesse.

Gesetzliche Vorgaben und Harmonisierung

EU-Vorgaben, Initiative „ViDA“ (VAT in the Digital Age)

Nationale Gesetze in Deutschland

Nachhaltigkeit und Platzersparnis

Ressourcenschonung: Verzicht auf Papier
Digitale Archivierung

E-Rechnungs-Formate

XRechnung

- Reines XML
- Standard für öffentliche Auftraggeber (B2G)

ZUGFeRD / Factur-X

- Hybrid: PDF/A-3 mit eingebettetem XML
- Mensch liest das PDF, Software das XML
- Sehr beliebt im normalen B2B-Geschäftsverkehr
- Aktuelle ZUGFeRD-Version V2.5.2, im Projekt V2.4
- „Zentraler User Guide des Forums elektronische Rechnung Deutschland“

PEPPOL

- KEIN Rechnungsformat
- Netzwerk-Standard für den europaweiten E-Rechnungs-Austausch

Beispiel ZUGFeRD-Rechnung

26 0121 INV.pdf - PDF24 Reader

1 / 1 92%

Suche

factor-x.xml

Rechnung

Seite: 1 / 1
 Datum: 23.02.2026
 Nr.: 26 0121
 Unser Zeichen:
 Durchwahl:
 Auftragsnr.: 2604 0035
 USt-ID:
 St.-Nr.:

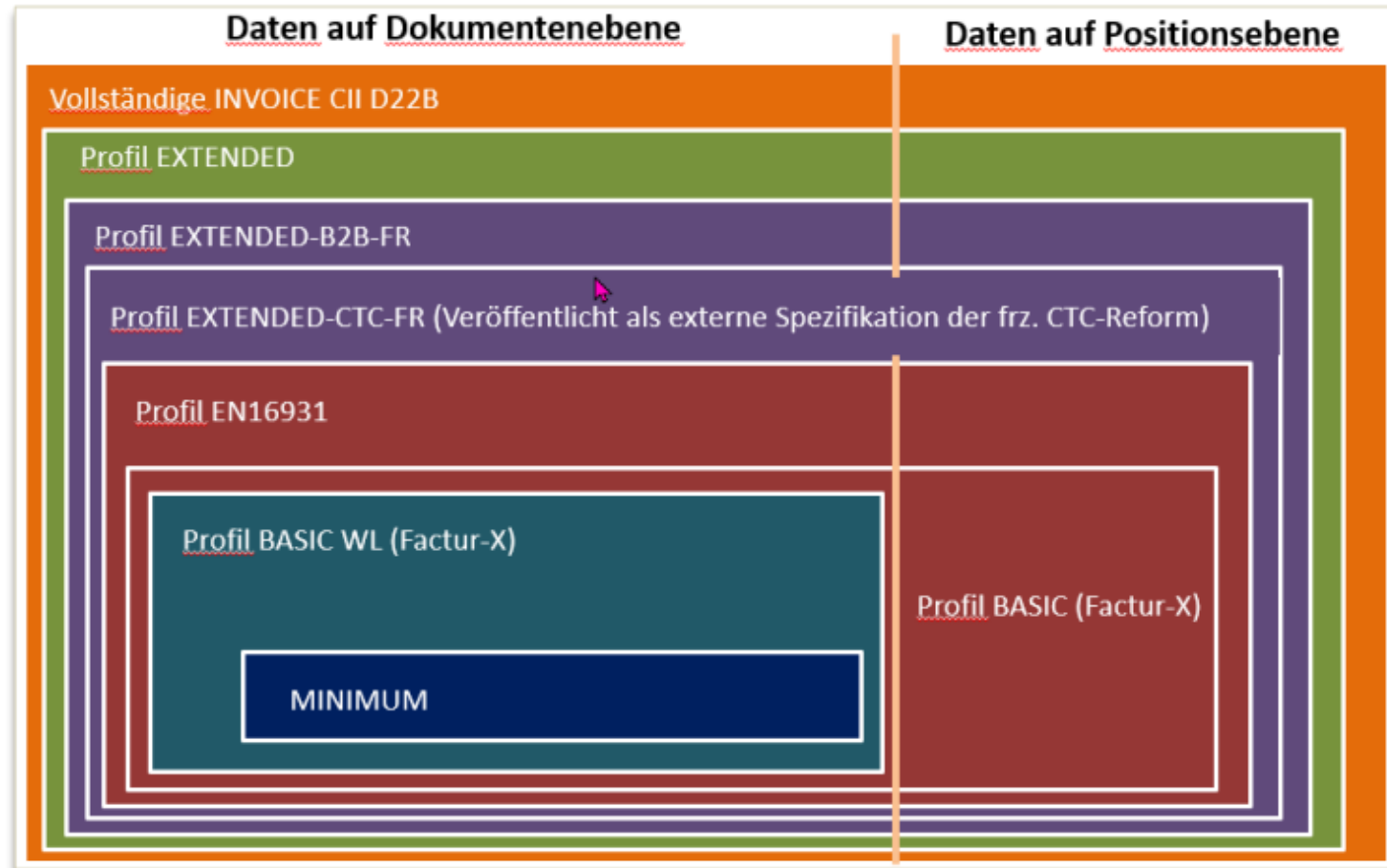
 Kundennr.: 12209
 Ihre USt-ID:
 Ihre Zeichen:
 Ihre Bestellung: 24024509
 vom: 20.01.2026

 Versand-KW: 9
 Zahlungsbedingung: 30 Tage rein netto
 Lieferbedingung:
 Versandart: Spedition

Pos. Bezeichnung Menge Einzelpreis Gesamtpreis €

1	WSC	50,00 kg	197,75	9.887,50
	Fracht			35,00
			Summe:	9.922,50
			19,00 % MwSt	1.885,28
			Gesamtbetrag:	€ 11.807,78

ZUGFeRD/Factor-X-Profile



Vorüberlegungen & Herangehensweise

Wie erzeuge ich die XML-Datei mit den Rechnungsdaten? =>
Analyse von Schema-Struktur und XML-Beispielen

6_ZUGFeRD_2.4_Technischer_Anhang_Profil_EN16931.pdf - PDF24 Reader

ZUGFeRD 2.4 / Factur-X 1.08 Spezifikation - Technischer Anhang

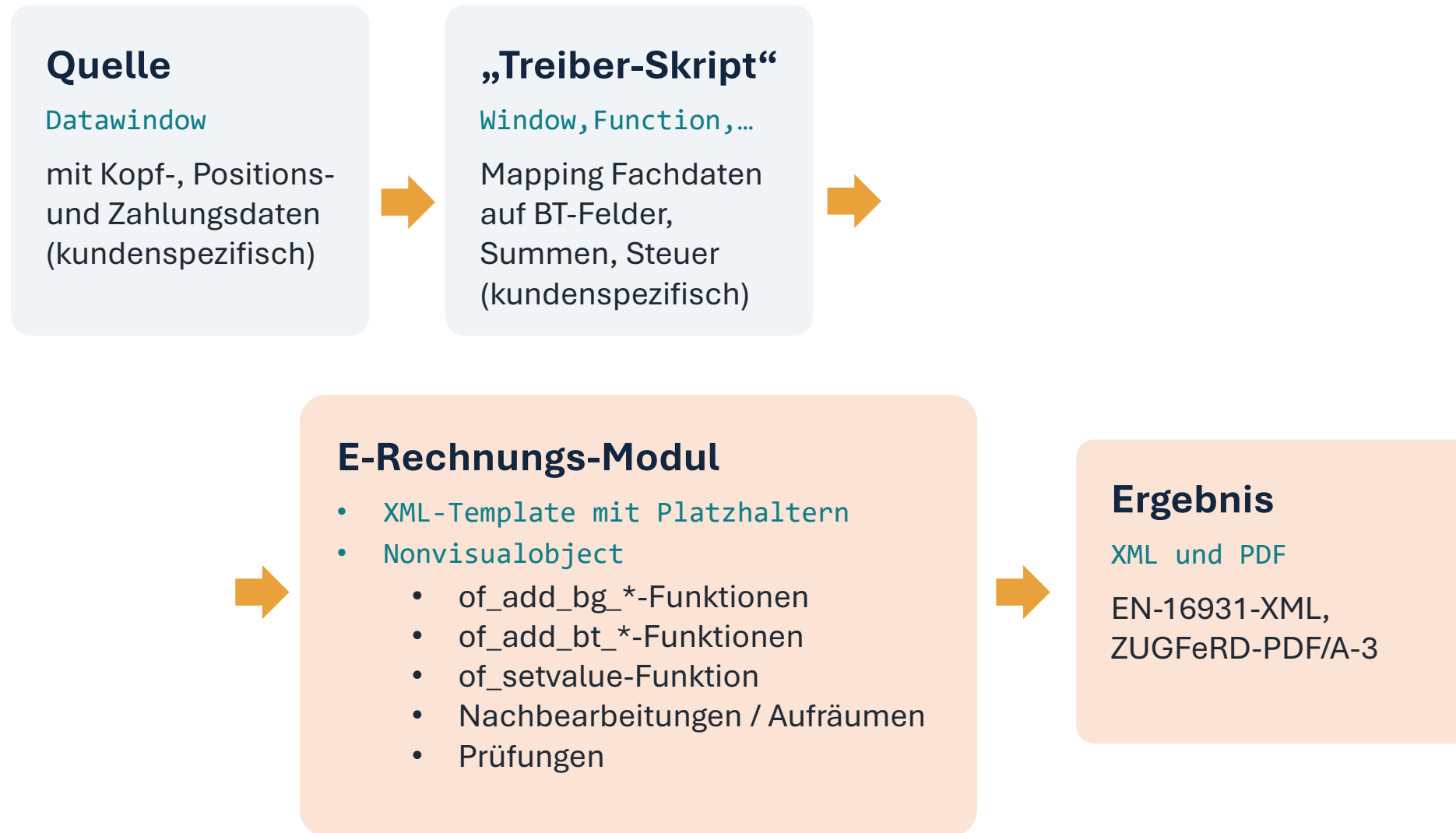
Struktur des CII-Schemas gemäß CEN/TS 16931-3-3 (identisch mit EN16931)

CrossIndustryInvoice		Rechnung	BG-0
1 .. 1	xs:sequence		
1 .. 1	ExchangedDocumentContext	PROZESSSTEUERUNG	BG-2
0 .. 1	xs:sequence		
	BusinessProcessSpecifiedDocumentContextParameter	Gruppierung der Geschäftsprozessinformationen	BT-23-00
1 .. 1	xs:sequence		
1 .. 1	ID	Geschäftsprozessstyp	BT-23
1 .. 1	GuidelineSpecifiedDocumentContextParameter	Gruppierung der Anwendungsempfehlungsinformationen	BT-24-00
1 .. 1	xs:sequence		
1 .. 1	ID	Spezifikationskennung	BT-24
1 .. 1	ExchangedDocument	Gruppierung der Eigenschaften, die das gesamte Dokument betreffen.	BT-1-00
1 .. 1	xs:sequence		
1 .. 1	ID	Rechnungsnummer	BT-1
1 .. 1	TypeCode	Code für den Rechnungstyp	BT-3
1 .. 1	IssueDateTime	Rechnungsdatum	BT-2-00
1 .. 1	xs:choice		
required	DateTimeString	Rechnungsdatum	BT-2
	format	Datum, Format	BT-2-0

```
<rsm:CrossIndustryInvoice xmlns:rsm="urn:un:unece:uncefact:data:standard:CrossIndustryInvoice:100" xmlns:qdt="urn:un:unece:uncefact:data:standard:QualifiedDataType:100" xmlns:ram="urn:un:unece:uncefact:data:standard:ReusableAggregateBusinessInformationEntity:100" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:udt="urn:un:unece:uncefact:data:standard:UnqualifiedDataType:100">
  <rsm:ExchangedDocumentContext>
    <ram:GuidelineSpecifiedDocumentContextParameter>
      <ram:ID>urn:cen.eu:en16931:2017</ram:ID>
    </ram:GuidelineSpecifiedDocumentContextParameter>
  </rsm:ExchangedDocumentContext>
  <rsm:ExchangedDocument>
    <ram:ID>471102</ram:ID>
    <ram:TypeCode>380</ram:TypeCode>
    <ram:IssueDateTime>
      <udt:DateTimeString format="102">20241115</udt:DateTimeString>
    </ram:IssueDateTime>
    <ram:IncludedNote>
      <ram:Content>Rechnung gemäß Bestellung vom 01.11.2024.</ram:Content>
    </ram:IncludedNote>
  </rsm:ExchangedDocument>

```

Architektur im Überblick



Generischer Teil wiederverwendbar, **Kundenlogik** sauber getrennt

Das Herzstück: ein XML-Template mit Platzhaltern

```
RECHNUNGSPPOSITION:
####BG-025_START####
<ram:IncludedSupplyChainTradeLineItem>
  <ram:LineID>!BT-126[1..1]!</ram:LineID>
  <ram:IncludedNote><ram:Content>
    !BT-127[0..1]!
  </ram:Content></ram:IncludedNote>
  ...
  <ram:BilledQuantity unitCode=
    "!BT-130[1..1]!">!BT-129[1..1]!
  </ram:BilledQuantity>
</ram:IncludedSupplyChainTradeLineItem>
####BG-025_END####
```

- Platzhalter: !BT-nnn[Kardinalität]!
 - [1..1] Pflicht, [0..n] optional
- Wiederholgruppen zwischen START- und END-Marker
- Import per ImportFile in einen Datastore („ids_xmlTemplate“)
- Datei als UTF-8 mit BOM (Umlaute!)
- Rechnung entsteht in einem weiteren Datastore („ids_xml4inv“)

Werte setzen mit of_setValue

```
// Zeile mit dem Platzhalter suchen
ls_search = "(content like '%!' + as_bt_id + "[%'])"
li_find = ids_xml4inv.Find(ls_search, 1, &
                          ids_xml4inv.RowCount())

if IsNull(as_value) or Trim(as_value) = '' then
  // leer: Platzhalter nur markieren
  as_value = Left(ls_placeholder, Pos(ls_ph, '[')-1) + &
              '(EMPTY)' + Mid(ls_ph, Pos(ls_ph, '['))
end if
as_value = of_escapeXML(as_value)
ls_content = of_replace(ls_content, ls_placeholder, as_value)
ids_xml4inv.SetItem(li_find, 'content', ls_content)
```

- Platzhalter suchen, Wert einsetzen, Zeile zurückschreiben
- Leerer Wert: Zeile wird später entfernt, wenn optional
- Datum: YYYYMMDD, Dezimalzahl: Punkt
- Sonderzeichen werden maskiert (&, <, ...)

Das „Treiber-Skript“: Mapping Fachdaten auf BT-Felder, Summen, Steuern...

```
for ll_p = 1 to ll_pcount
    ...
    // BG-025[1..n] RECHNUNGSPPOSITION(EN) (IncludedSupplyChainTradeLineItem)
    // Parameter1: BT-126 : [1..1] Wert für "LineID => Kennung der Rechnungsposition"
    // Parameter2: BT-127 : [0..1] Wert für "IncludedNote Content => Freitext zur Rechnungsposition"
    // Parameter3: BT-129 : [1..1] Wert für "BilledQuantity => In Rechnung gestellte Menge"
    // Parameter4: BT-130 : [1..1] Wert für "unitCode => Code der Maßeinheit der in Rechnung gestellten Menge"
    // (Stück=H87, Stunde=HUR, Kilogramm=KGM, Gramm=GRM, Liter=LTR, ...)
    ls_param1 = String(ll_p)
    ls_param2 = ""
    ls_param3 = String(ll_qty)
    ls_param4 = ls_unit
    li_ret = Inv_xml.of_add_bg_025_tradeLineItem(ls_param1, ls_param2, ls_param3, ls_param4)

    // BG-031[1..1] ARTIKELINFORMATIONEN (SpecifiedTradeProduct)
    // Parameter1: BT-157 : [0..1] Wert für "GlobalID => Kennung eines Artikels nach registriertem Schema"
    // Parameter2: BT-157-1 : [0..1] Wert für "schemeID => Kennung des Schemas"
    // Parameter3: BT-155 : [0..1] Wert für "SellerAssignedID => Artikelkennung des Verkäufers"
    // Parameter4: BT-156 : [0..1] Wert für "BuyerAssignedID => Artikelkennung des Käufers"
    // Parameter5: BT-153 : [1..1] Wert für "Name => Artikelname "
    // Parameter6: BT-154 : [0..1] Wert für "Description => Artikelbeschreibung"
    // Parameter7: BT-159 : [0..1] Wert für "ID => Artikelherkunftsland" (Germany=DE)
    ls_param1 = ""
    ls_param2 = ""
    ls_param3 = Trim(dw_inv.GetItemString(ll_row, 'invpos_artikel_code'))
    ls_param4 = ""
    ls_param5 = Trim(dw_inv.GetItemString(ll_row, 'invpos_artikel_bez'))
    ls_param6 = ls_artikelbeschreibung
    ls_param7 = ""
    li_ret = Inv_xml.of_add_bg_031_tradeProduct(ls_param1, ls_param2, ls_param3, ls_param4, ls_param5, ls_param6, ls_param7)
    ...
next
```

Aufräumen und Prüfen vor dem Speichern

1

of_removeUnusedElements

Optionale Elemente [0..x] ohne Wert werden gelöscht

2

of_removeCommentLines

Reine Kommentarzeilen und ungenutzte Block-Platzhalter entfallen

3

of_checkMandatoryElements

Pflichtfelder [1..x] ohne Wert landen in einer Fehlerliste

4

of_saveXml4Inv

Speichern als XML in UTF-8 mit BOM:
`ids_xml4inv.SaveAs(as_filename, Text!, False, EncodingUTF8!)`

Eine kurze Live-Demo...

Vom XML zum ZUGFeRD-PDF

1 XML speichern als „factur-x.xml“

2 PDF importieren und XML anhängen

3 Als PDF/A-3 speichern

4 ExifTool: XMP-Metadaten schreiben

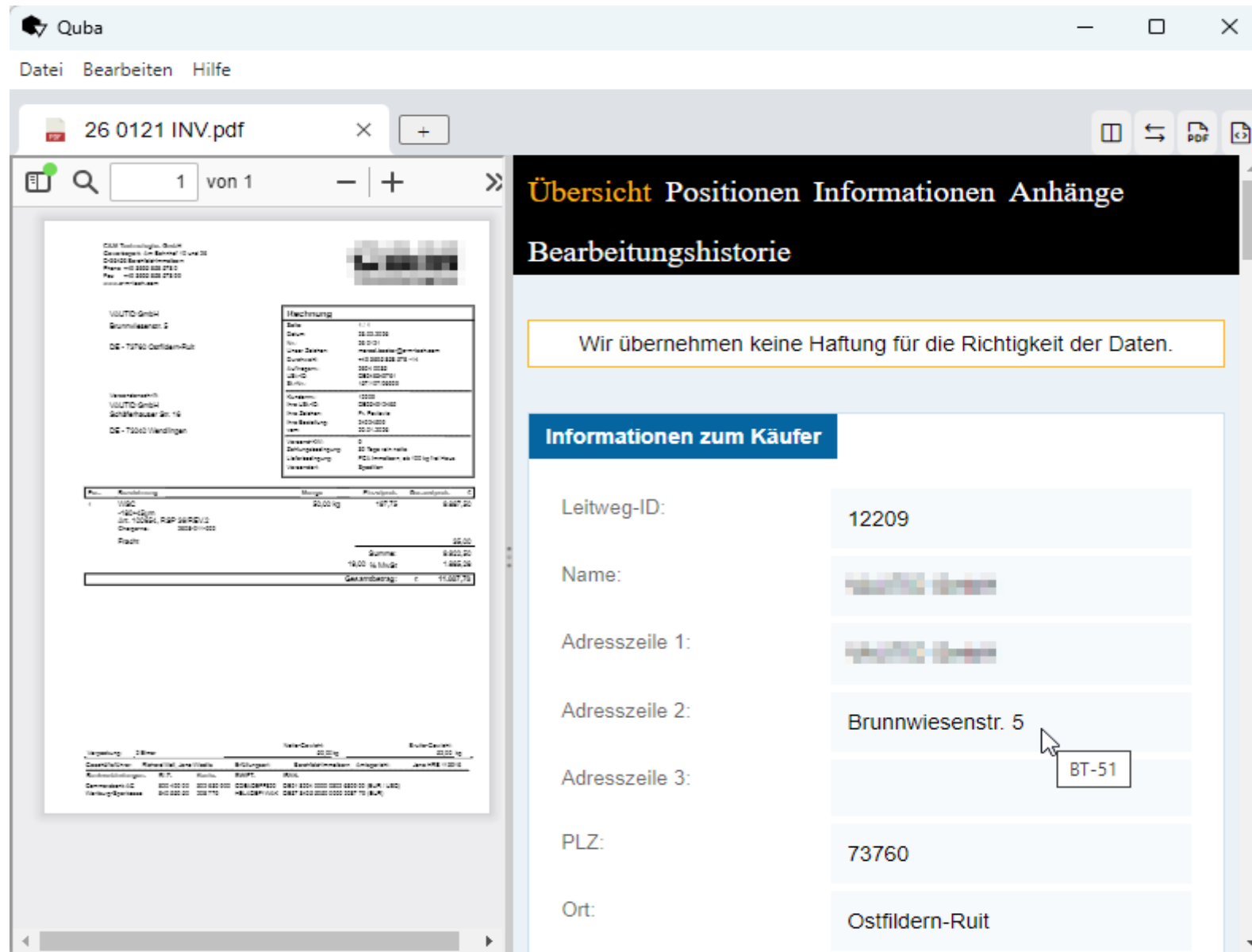
```
PDFdocument lpdf_doc
PDFattachment lpdf_attach

...
lpdf_doc.ImportPdf(as_pdffilename)
lpdf_attach.AddFile(ls_path + '\factur-x.xml')
lpdf_doc.attachment = lpdf_attach
lpdf_doc.Save(ls_path + '\' + ls_pdf, PDF_A_3A!)

// XMP-Metadaten ergänzen
Run('"exiftool.exe" -overwrite_original ' + &
    '-tagsfromfile zf_meta4pdf_en16931.xmp ' + &
    '-XMP rechnung.pdf', Minimized!)
```

Dateiname muss factur-x.xml heißen. Der PDF-Standard muss zur XMP-Vorlage passen.

Visualisierung des ZUGFeRD-PDF: z.B. mit „Quba“



Validierung des ZUGFeRD-PDF: z.B. über „eu-rechnung.de“

[eu-rechnung.de/e-rechnung-validieren/5a92617b-a92f-4da0-9e8d-449139cdce1a](#)

Bericht zur Konformitätsprüfung

1 / 3 | - 100% +

RECHNUNG

Rechnung 26 0121 vom 23.02.2026
(Konformitätsprüfung der E-Rechnung)

Bericht zur Konformitätsprüfung

1) Zusammenfassung der Prüfung

Konformität	Das geprüfte Dokument enthält weder Fehler noch Warnungen. Es ist konform zu den formalen Vorgaben. Es wird empfohlen das Dokument anzunehmen und weiter zu verarbeiten.
Referenz	5a92617b-a92f-4da0-9e8d-449139cdce1a
Verwendete Prüfsoftware	EU-Rechnung v3.22
Zeitpunkt der Prüfung	09.10.2026, 17:09:06
Dateiname der Rechnung	26 0121 INV.pdf
Dateigröße der Rechnung	162.077 Bytes
Prüfsumme (SHA-256)	5f2f9465eb200ad44854acc39c1cf9e8646a1db25ddf3706382391c01ea49838
Erkannter Dokumenttyp	ZUGFeRD 2 (EN16931)
Erkannte Rechnungsnummer	26 0121
Erkanntes Rechnungsdatum	23.02.2026

Fragen?

