LENIC EDI-KONVERTER

Benutzerhandbuch

Einrichten und bedienen · das Mapping · die Definitionssprache

Version 1.15 · Stand 02.11.2026
Ljuban Magdelinic, IT Consulting

Inhalt

1 Der Konverter im Betrieb

1.1 Systemvoraussetzungen
1.2 Sprache
1.3 Installation
1.4 Lizenz anfordern und einspielen
1.5 Erster Start
1.6 Mitgelieferte Beispiele
1.7 Update auf eine neue Version
1.8 Der Editor
1.9 ZUGFeRD-Rechnungen
1.10 Daten aus dem SQL Server (SQL-LINK)
1.11 Der Aufruf im Betrieb
1.12 Rückgabewerte
1.13 Log, Laufprotokoll und Verlauf
1.14 Archiv, Wiederherstellen und Löschschutz
1.15 Gleichzeitige Läufe
1.16 Datensicherung
1.17 Abweichende Pfade
1.18 E-Mail über Microsoft 365
1.19 Fehlerbehebung und häufige Fragen
1.20 Deinstallation
1.21 Kontakt

2 Das Mapping

2.1 Was ein Mapping ist
2.2 Die Satzbeschreibung
2.3 Beide Richtungen aus einer Beschreibung
2.4 Ein Durchlauf zum Mitlesen
2.5 Lieferavis: Sendung, Paletten, Positionen
2.6 Rechnung: Parteien, Rabatte, Steuern
2.7 Den Entwurf von Hand weiterschreiben

3 Die Definitionssprache

3.1 Aufbau einer Definition
3.2 Werte
3.3 GLOBAL-CONF
3.4 SOURCE-CONF
3.5 TARGET-CONF
3.6 START-CONF, RUN-CONF, END-CONF
3.7 Aufruf und Rückgabewerte
3.8 Vollständiges Beispiel

TEIL 1

Der Konverter im Betrieb

Einrichten, bedienen, laufen lassen

Der LeNic EDI-Konverter wandelt Dateien zwischen Ihrem Warenwirtschaftssystem und Ihren Geschäftspartnern um – in beide Richtungen:

- **Outbound:** aus Ihren Inhouse-Daten wird eine EANCOM/EDIFACT-Nachricht, etwa eine Rechnung (INVOIC).
- **Inbound:** aus einer EDIFACT-Nachricht Ihres Partners werden Inhouse-Daten, etwa eine Bestellung (ORDERS) als CSV-Datei für Ihre Warenwirtschaft.

Er besteht aus zwei Programmen, die sich Datenbank und Lizenz teilen:

- Der **Editor** ist die Oberfläche. Hier pflegen Sie Partner, Konfigurationen und die Konverterdefinitionen, und hier probieren Sie eine Wandlung aus.
- Der **Konverter** ist das Programm, das im Betrieb läuft. Er wird mit genau einem Parameter aufgerufen – der Nummer einer gespeicherten Konfiguration – und braucht keine Oberfläche.

1.1 Systemvoraussetzungen

- Windows 10 oder Windows 11 (64 Bit)
- **.NET-10-Laufzeit**, und zwar die Variante **Windows Desktop Runtime** (x64); sie enthält die Laufzeit für den Konverter mit. Das Setup-Programm bringt sie mit und installiert sie bei Bedarf selbst.
- Eine gültige Lizenz für den Rechner, auf dem gearbeitet wird (siehe „Lizenz anfordern und einspielen“).
- Für die Installation Administratorrechte; im Betrieb genügt ein normales Benutzerkonto mit Schreibrecht in den Ordnern, aus denen gelesen und in die geschrieben wird.

Eine Datenbank müssen Sie nicht einrichten: Der Konverter bringt SQLite mit und legt seine Datei selbst an. Nur wer mit `SQL-LINK` Daten aus der Warenwirtschaft liest, braucht Zugang zu deren Microsoft SQL Server (siehe „Daten aus dem SQL Server“).

1.2 Sprache

Editor und Konverter sprechen **Deutsch** und **Englisch**. Ohne Einstellung folgen sie der Anzeigesprache von Windows – Deutsch bei deutschem Windows, sonst Englisch.

Umgestellt wird im Editor unter **Einstellungen** → **Sprache**: „Wie Windows“, „Deutsch“ oder „English“. Die Einstellung steht in der Datenbank und gilt deshalb für Editor und Konverter gleichermaßen:

- Der **Konverter** verwendet sie ab dem nächsten Lauf – für Meldungen, Befunde und die Logdatei.
- Der **Editor** fragt, ob er gleich neu starten soll; danach gilt die neue Sprache.

Was in der Datenbank steht, bleibt, wie es geschrieben wurde: Das Laufprotokoll behält die Meldungen in der Sprache des Laufs. Die Definitionssprache selbst – `IF`, `WRITE`, `NAD+BY+VALUE` … – ist in beiden Sprachen dieselbe.

Das Benutzerhandbuch gibt es auf Englisch als `USER-MANUAL.md` und als PDF `LeNic-EDI-Converter-User-Manual.pdf`.

1.3 Installation

Ausgeliefert wird in zwei Formen, jede auf Deutsch und auf Englisch. Die englischen Dateien tragen `-en` im Namen (`…-Setup-en.exe`, `…-x64-en.msi`); ihr Assistent und ihr Lizenzdialog sind englisch. Beide Sprachen installieren dasselbe Programm und ersetzen einander bei einem Update.

- **`LeNicEdiKonverter-<Fassung>-Setup.exe`** – das Setup-Programm. Es bringt die .NET Desktop Runtime 10 mit und installiert sie, falls sie fehlt, vor dem Programm. Eine Internetverbindung braucht es nicht. Für die meisten Installationen ist das der einfachste Weg.
- **`LeNicEdiKonverter-<Fassung>-x64.msi`** – das Installationspaket allein, etwa für die Verteilung durch eine Softwareverwaltung. Es setzt die .NET Desktop Runtime 10 voraus und bricht mit einem Hinweis ab, wenn sie fehlt.

Welche Datei?

Beide installieren dasselbe Programm in denselben Ordner – das Setup-Programm enthält das MSI-Paket und führt es selbst aus. Der Unterschied ist die .NET-Laufzeit:

	Setup-Programm (`Setup.exe`)	Installationspaket (`x64.msi`)
Größe	rund 70 MB	rund 12 MB
Inhalt	MSI-Paket und .NET Desktop Runtime 10	nur der LeNic EDI-Konverter
Fehlt .NET 10	wird zuerst installiert, ohne Internet	Abbruch mit Hinweis, .NET 10 vorher installieren
Gedacht für	die Installation von Hand auf einem Rechner	die zentrale Verteilung (Gruppenrichtlinie, Intune, SCCM), wenn .NET 10 dort schon vorhanden ist
Ohne Rückfragen	`/quiet InstallFolder=…`	`msiexec /i … /qn INSTALLFOLDER=…`

Beide aktualisieren eine vorhandene Installation – etwa 1.3 auf 1.4 – in ihrem bisherigen Ordner; Datenbank, Lizenz und Ihre Dateien bleiben erhalten.

Faustregel: Wer selbst installiert, nimmt das **Setup-Programm**. Das **MSI-Paket** ist für Administratoren, die Software zentral verteilen und die .NET Desktop Runtime 10 dabei schon haben oder eigens mitverteilen.

Warnung von Microsoft Defender SmartScreen

Die Pakete sind noch nicht digital signiert. Windows warnt deshalb beim Herunterladen und beim ersten Start – das ist bei neuer, unsignierter Software üblich und sagt nichts über die Datei selbst.

- **Edge blockiert den Download:** Download-Liste öffnen (Strg+J), bei der Datei auf **„…“** klicken → **„Behalten“** → **„Mehr anzeigen“** → **„Trotzdem behalten“**.
- **Windows hat den Start verhindert: „Weitere Informationen“**, dann **„Trotzdem ausführen“**.

Ob die Datei unverändert angekommen ist, zeigt ihre Prüfsumme. Sie steht in `SHA256SUMS.txt` neben den Downloads und lässt sich in PowerShell vergleichen:

```
Get-FileHash .\LeNicEdiKonverter-1.9.1-Setup.exe -Algorithm SHA256
```

Der Assistent

Starten Sie eines davon und folgen Sie dem Assistenten:

- **Lizenzvertrag** lesen und annehmen.
- **Zielordner**: Vorgeschlagen ist `C:\Tools\LeNicEdiKonverter`; im Setup-Programm ändern Sie ihn unter **Optionen**. Bei einem Update ist es der Ordner der bisherigen Installation.
- **Installieren** – Windows fragt nach Administratorrechten.

Danach finden Sie im Startmenü unter „LeNic EDI-Konverter“ den **Editor** und das **Benutzerhandbuch**.

Für eine Installation ohne Rückfragen, etwa per Softwareverteilung:

```
LeNicEdiKonverter-<Fassung>-Setup.exe /quiet InstallFolder=D:\EDI\LeNicEdiKonverter
msiexec /i LeNicEdiKonverter-<Fassung>-x64.msi /qn INSTALLFOLDER=D:\EDI\LeNicEdiKonverter\
```

Der Installationsordner

Der Zielordner heißt im Folgenden **Installationsordner**. Darunter liegt:

```
C:\Tools\LeNicEdiKonverter\          ← Installationsordner
    Konverter\LeNic.EdiKonverter.App.exe    ← das Programm für den Betrieb
    Editor\LeNic.EdiKonverter.Wpf.exe       ← die Oberfläche
    data\LeNic.EdiKonverter.db              ← Datenbank, mit den Beispielen
    Lizenz\license.dat                      ← Ihre Lizenzdatei
    temp\                                   ← Logs und Sperren
    Import\  INHOUSE\  Export\  Archiv\       ← Ein- und Ausgabe der Beispiele
    Beispiele\Beispiele.db                  ← die Beispiele dieser Fassung
    LeNic-EDI-Konverter-Benutzerhandbuch.pdf, BENUTZERHANDBUCH.md
    LeNic-EDI-Converter-User-Manual.pdf, USER-MANUAL.md   ← auf Englisch
    LICENSE.md, THIRD-PARTY-NOTICES.md (dazu .en.md auf Englisch)
```

Beide Programme finden ihren Installationsordner selbst: Es ist der Ordner **über** dem Programmordner. Sie müssen nichts eintragen, solange Sie den Aufbau so lassen. Abweichende Pfade sind möglich, siehe „Abweichende Pfade".

In `data\`, `Lizenz\`, `temp\`, `Import\`, `INHOUSE\`, `Export\` und `Archiv\` dürfen alle Benutzer des Rechners schreiben – der Editor und der Konverter brauchen das, auch ohne Administratorrechte. Die Programmordner `Konverter\` und `Editor\` bleiben schreibgeschützt.

Ihre Daten gehören Ihnen: Die Datenbank und die Testdaten legt das Setup nur an, wenn es sie noch nicht gibt. Eine neue Version und auch die Deinstallation fassen sie nicht an – ebenso wenig `Lizenz\` und Ihre Dateien in den Ein- und Ausgabeordnern.

1.4 Lizenz anfordern und einspielen

Der LeNic EDI-Konverter ist lizenzpflichtig. Eine Lizenz gilt **für einen bestimmten Rechner** und bis zu einem Ablaufdatum.

- Den Rechnernamen, den die Lizenz tragen muss, nennt Ihnen der Editor: Er steht im Lizenzfenster beim Start und unter **Einstellungen → Daten & Lizenz → Lizenz**.
- Fordern Sie die Lizenz beim Lizenzgeber an und geben Sie dabei den Rechnernamen und das Produkt **„LeNic EDI-Konverter"** an. Eine Lizenz für den LeNicScheduler gilt hier nicht.
- Die Datei heißt `license.dat` und gehört in den Ordner `Lizenz\` des Installationsordners. Bequemer geht es im Editor: **Einstellungen → Daten & Lizenz → Lizenz → „Lizenzdatei einspielen …"**. Startet der Editor mangels Lizenz gar nicht, bietet er das Einspielen gleich im Lizenzfenster an.
- Eine ungültige Datei ersetzt **nie** eine vorhandene gültige Lizenz.

Ohne gültige Lizenz startet der Editor nicht, und der Konverter wandelt nichts; er endet mit Rückgabewert 13.

Module

Zur Lizenz gehört die **Basis**; drei **Module** kommen nach Bedarf dazu, jedes mit eigenem Ablaufdatum:

Teil	Umfasst
Basis	CSV, Text und feste Satzlänge – in beide Richtungen
Modul **EDIFACT**	EDIFACT als Eingabe oder Ausgabe – erkannt an `FORMAT = EDIFACT` und am Inhalt der Datei (UNA, UNB oder UNH)
Modul **JSON/XML**	JSON und XML als Quelle oder Ziel
Modul **ZUGFeRD**	Ausgabe als ZUGFeRD-PDF, samt der XML-Rechnung darin – dafür braucht es JSON/XML nicht

Welche Module Ihre Lizenz umfasst und bis wann, steht unter **Einstellungen → Daten & Lizenz → Lizenz**. Braucht ein Lauf ein Modul, das fehlt oder abgelaufen ist, wandelt der Konverter nichts und endet mit Rückgabewert 13; die Meldung nennt das Modul. Bearbeiten und mit dem **Testlauf** ausprobieren lassen sich alle Definitionen auch ohne Modul. Meldungen und Befunde stehen dann vollständig da, von der Ausgabe aber nur der Anfang – höchstens 20 Zeilen bzw. EDIFACT-Segmente; ebenso im Ablauf.

Eine Lizenz aus der Zeit vor den Modulen gilt weiter – mit EDIFACT und JSON/XML, aber ohne ZUGFeRD. Umgekehrt gilt eine Lizenz mit Modulen erst ab Version 1.10; eine ältere Fassung lehnt sie ab. Deshalb erst aktualisieren, dann die neue Lizenz einspielen.

ZUGFeRD ausprobieren: Im Testlauf auf der Seite „Konverter" zeigt **„Als ZUGFeRD-PDF ansehen"** die Ausgabe als Rechnung. Ohne das Modul ZUGFeRD ist es ein **Muster** – mit Wasserzeichen und ohne die eingebettete XML, also keine E-Rechnung, die man versenden könnte.

1.5 Erster Start

Starten Sie den Editor. Die Auslieferung bringt eine Datenbank mit lauffähigen Beispielen mit (siehe unten); fehlt sie, legt der Editor beim ersten Start eine leere unter `data\` an. Für einen eigenen Partner richten Sie in dieser Reihenfolge ein:

- **Partner** anlegen (Firmenname und GLN),
- **Konverter** schreiben oder aus einer Satzbeschreibung erzeugen lassen,
- **Konfiguration** für diesen Partner anlegen, die beides zusammenbringt.

1.6 Mitgelieferte Beispiele

Damit Sie sofort sehen, wie ein Durchlauf aussieht, bringt die Auslieferung einen Beispielpartner mit elf Konfigurationen und passenden Testdaten mit. Partner ist „Magdelinic Consulting" mit der Test-GLN `9999999999994`; die Bestellungen, Lieferavise und Rechnungen gehören zusammen (Bestellungen PO-2026-100234 und PO-2026-100235).

Nr.	Konverter	Richtung	Eingabe	Ausgabe
3	ORDER-INBOUND	Bestellung herein	`{BASIS}\Import\ORDER*.EDI`	`{BASIS}\INHOUSE\ORDER_{ZAEHLER}.CSV`
4	ORDER-OUTBOUND	Bestellung hinaus	`{BASIS}\INHOUSE\ORDER_OUT.CSV`	`{BASIS}\Export\ORDER_OUT_{ZAEHLER}.EDI`
5	DESADV-OUTBOUND	Lieferavis hinaus	`{BASIS}\INHOUSE\DESADV_OUT.CSV`	`{BASIS}\Export\DESADV_OUT_{ZAEHLER}.EDI`
6	INVOIC-OUTBOUND	Rechnung hinaus	`{BASIS}\INHOUSE\INVOIC_OUT.CSV`	`{BASIS}\Export\INVOIC_OUT_{ZAEHLER}.EDI`
7	ZUGFERD-CSV	Rechnung als ZUGFeRD-PDF (EN 16931)	`{BASIS}\INHOUSE\ZUGFERD_OUT.CSV`	`{BASIS}\Export\ZUGFERD_OUT_{ZAEHLER}.PDF`
8	ZUGFERD-CSV	Sammelrechnung als ZUGFeRD-PDF (EN 16931)	`{BASIS}\INHOUSE\ZUGFERD_SAMMEL.CSV`	`{BASIS}\Export\ZUGFERD_SAMMEL_{ZAEHLER}.PDF`
9	ZUGFERD-INVOIC	EANCOM-Rechnung als ZUGFeRD-PDF (EXTENDED)	`{BASIS}\INHOUSE\ZUGFERD_INVOIC.EDI`	`{BASIS}\Export\ZUGFERD_INVOIC_{ZAEHLER}.PDF`
10	ORDER-INBOUND-XML	Bestellung herein, als XML	`{BASIS}\Import\XML_ORDER*.EDI`	`{BASIS}\INHOUSE\ORDER_{ZAEHLER}.XML`
11	ORDER-INBOUND-JSON	Bestellung herein, als JSON	`{BASIS}\Import\JSON_ORDER*.EDI`	`{BASIS}\INHOUSE\ORDER_{ZAEHLER}.JSON`
12	ORDER-OUTBOUND-XML	Bestellung hinaus, aus XML	`{BASIS}\INHOUSE\ORDER_OUT.XML`	`{BASIS}\Export\ORDER_OUT_XML_{ZAEHLER}.EDI`
13	ORDER-OUTBOUND-JSON	Bestellung hinaus, aus JSON	`{BASIS}\INHOUSE\ORDER_OUT.JSON`	`{BASIS}\Export\ORDER_OUT_JSON_{ZAEHLER}.EDI`

`{BASIS}` ist der Installationsordner – die Beispiele laufen also in jedem Zielordner, ohne dass Sie Pfade anpassen.

Die Ordner liegen im Installationsordner:

- `Import\` – was von Partnern hereinkommt (EDIFACT),
- `INHOUSE\` – Ihre Inhouse-Dateien, in beiden Richtungen,
- `Export\` – was an Partner hinausgeht (EDIFACT),
- `Archiv\` – verarbeitete Eingabedateien.

So probieren Sie es aus: Auf der Seite „Partner" den Beispielpartner wählen und bei einer Konfiguration **Run** drücken. Die Ausgabe entsteht im Ordner `Export\` bzw. `INHOUSE\`, der Verlauf zeigt den Lauf. Beachten Sie: Die Beispiele **archivieren** – die Eingabedatei wandert nach dem Lauf nach `Archiv\`. Wollen Sie denselben Lauf wiederholen, holen Sie sie mit **Archiv → Wiederherstellen** zurück. Ohne Ausgabe ausprobieren geht mit dem **Testlauf** auf der Seite „Konverter".

Haben Sie Beispiele gelöscht oder kommt mit einem Update ein neues dazu, holen Sie sie unter **Einstellungen → Daten & Lizenz → Beispiele → „Beispiele übernehmen …"** zurück. Übernommen wird nur, was fehlt – Partner nach GLN, Konverter nach Namen, Konfigurationen nach Konverter und Eingabedatei –, mit dem Zähler 0.

Die vier EDIFACT-Konverter sind aus den mitgelieferten Satzbeschreibungen erzeugt (siehe „Aus Satzbeschreibung") und eignen sich als Ausgangspunkt für Ihre eigenen – kopieren, umbenennen, anpassen. Wie Bestellung, Lieferavis und Rechnung im Einzelnen gebaut sind, zeigt der Teil „Das Mapping" mit den Inhouse-Zeilen und der fertigen Nachricht. Die drei ZUGFeRD-Beispiele machen aus einer Rechnung ein ZUGFeRD-PDF, siehe „ZUGFeRD-Rechnungen": **ZUGFERD-CSV** liest das Inhouse-CSV – in Konfiguration 7 eine einfache Rechnung, in 8 eine Sammelrechnung über zwei Lieferscheine –,

ZUGFERD-INVOIC eine EANCOM-Rechnung. Jeder trägt seine **Ausgabevorlage** – die Rechnung als XML – selbst, siehe „Ausgabevorlagen“.

Die Konfigurationen 10 bis 13 zeigen dieselbe Bestellung wie 3 und 4, aber mit einer **XML- bzw. JSON-Datei** statt des Inhouse-CSV. Beide Formate haben denselben Aufbau: `bestellungen` mit je einer `bestellung` (Nummer, Bestelldatum und Liefertermin als JJJJ-MM-TT, `partner` für Besteller und Lieferanschrift, `position` für jede Position). Die Partner unterscheidet `typ` – in JSON ein eigener Eintrag, in XML ein Attribut (`<partner typ="DP">`). **ORDER-INBOUND-XML** und **-JSON** schreiben jede Nachricht der EDIFACT-Datei als eigene Bestellung. **ORDER-OUTBOUND-XML** und **-JSON** lesen eine Bestellung je Datei und erzeugen daraus dieselbe EANCOM-Bestellung (D.01B) wie ORDER-OUTBOUND; steht eine zweite Bestellung in der Datei, bricht der Lauf mit einer Meldung ab. Die Definitionen liegen auch unter `Konverter\Samples\` . Die Eingaben heißen `XML_ORDER.EDI` und `JSON_ORDER.EDI` , damit Konfiguration 3 (`ORDER*.EDI`) sie nicht mitnimmt.

1.7 Update auf eine neue Version

Beenden Sie Editor und Konverter und starten Sie das neue Installationspaket. Es ersetzt die bisherige Fassung im selben Ordner – auch wenn Sie bei der ersten Installation einen anderen als den vorgeschlagenen gewählt haben. Ersetzt werden nur die Programmordner `Konverter\` und `Editor\` , das Handbuch und die Lizenztexte. **Ihre Daten bleiben:** Datenbank (`data\`), Lizenz (`Lizenz\`) und Ihre Ein- und Ausgabeordner werden nicht angefasst. Eine ältere Fassung über eine neuere zu installieren, lehnt das Setup ab.

Neue Beispiele einer Fassung kommen deshalb nicht von selbst in Ihre Datenbank. Übernehmen Sie sie bei Bedarf im Editor unter **Einstellungen → Beispiele → „Beispiele übernehmen …“**: Angelegt wird nur, was fehlt; Konverter und Konfigurationen, die Sie schon haben oder geändert haben, bleiben unberührt.

Braucht die neue Version eine Änderung an der Datenbank, geschieht das beim ersten Öffnen von selbst:

- Vorher legt das Programm **eine vollständige Sicherung** daneben, benannt wie `LeNic.EdiKonverter.vor-Schema1-20260919-115102.db` .
- Danach meldet der Editor oben „Datenbank aktualisiert“ und nennt die Sicherung.
- Öffnen zwei Programme gleichzeitig eine alte Datenbank, wartet das zweite; geändert wird genau einmal. Schlägt etwas fehl, bleibt alles, wie es war.

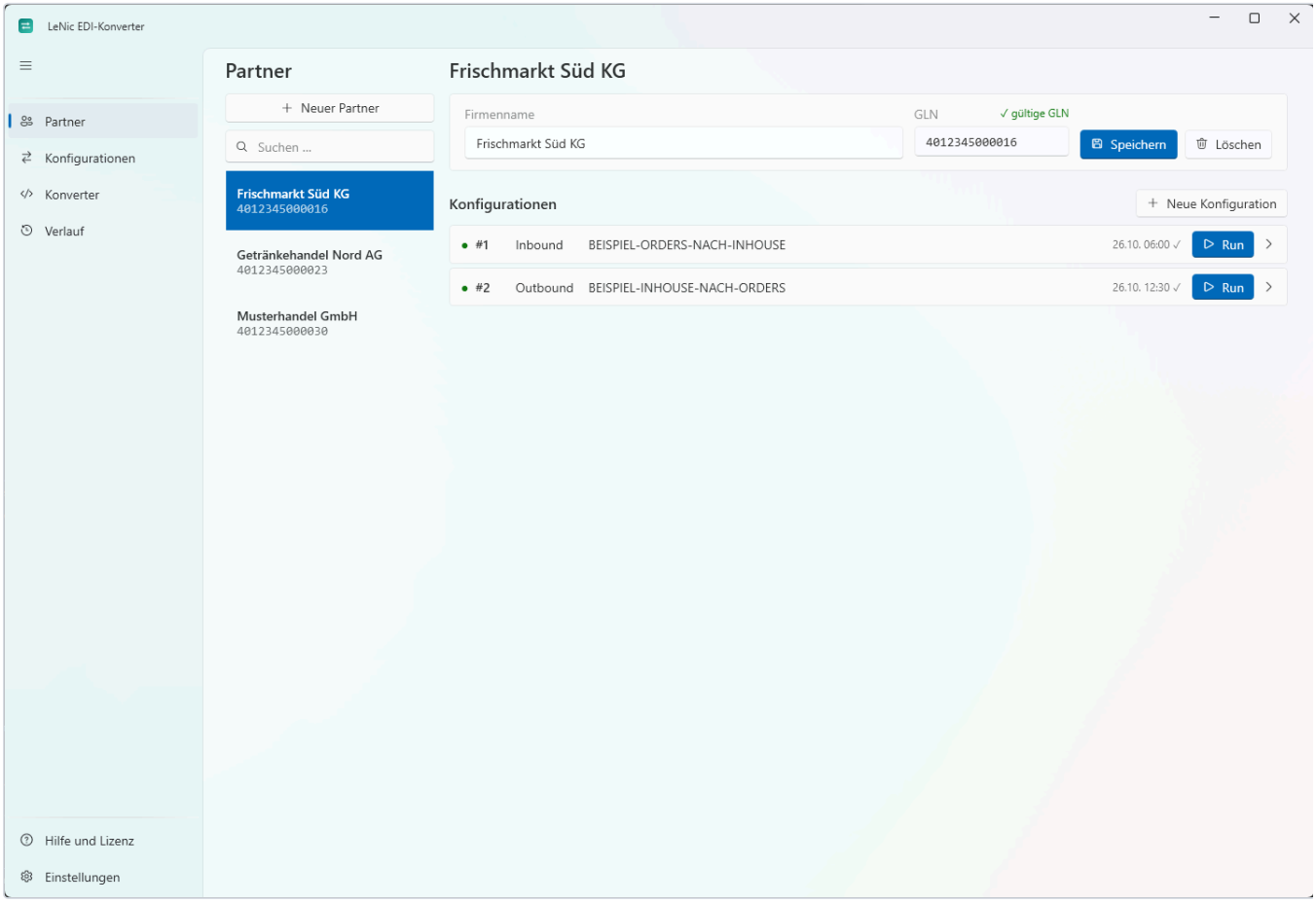
Eine Datenbank, die von einer **neueren** Programmfassung stammt, wird abgewiesen – dann gehört das Programm aktualisiert, nicht die Datenbank.

Laufen geplante Aufgaben, die den Konverter aufrufen, halten Sie sie für die Dauer des Updates an.

1.8 Der Editor

Links wählen Sie die Seite: **Partner**, **Konverter**, **Verlauf**, unten **Hilfe und Lizenz** und **Einstellungen**.

Seite „Partner“



Die Partnerseite: links die Partner, rechts der gewählte Partner mit einer Zeile je Konfiguration. Der Punkt zeigt den letzten Lauf, „Run“ führt eine Konfiguration sofort aus.

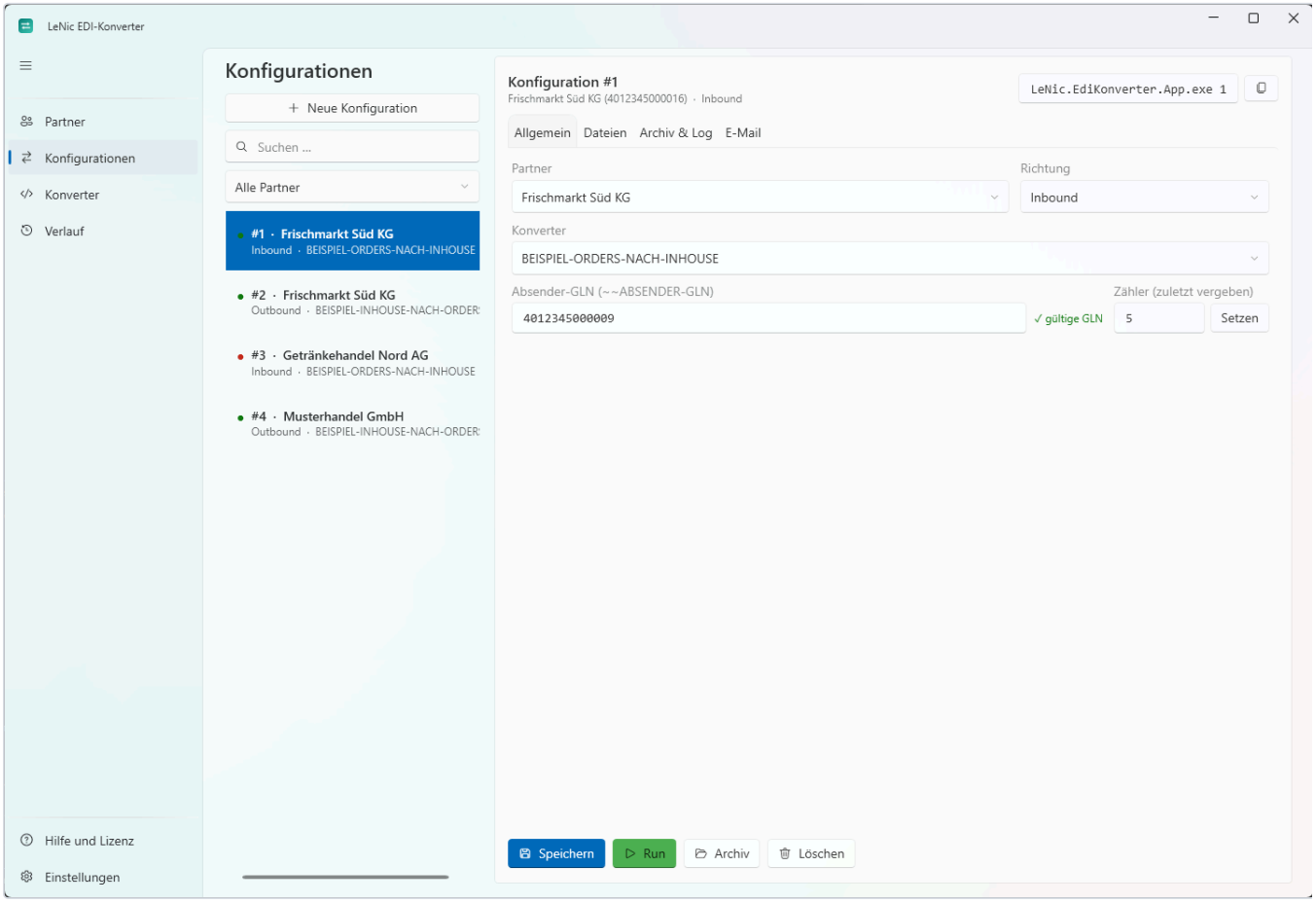
Die Arbeit geht vom Partner aus: Sie wählen links einen Partner und sehen rechts seine Daten und seine Konfigurationen.

- **Firmenname** – wie der Partner bei Ihnen heißt.
- **GLN** – die 13-stellige Globale Lokationsnummer des Partners. Der Editor prüft sie sofort und nennt Ihnen bei zwölf eingegebenen Ziffern die fehlende Prüfziffer. Jede GLN darf nur einmal vergeben sein.

Konfigurationen gibt es erst zu einem **gespeicherten** Partner. Jede steht in einer Zeile: farbiger Punkt für den letzten Lauf (grün erfolgreich, gelb mit Warnungen, rot gescheitert, grau noch keiner), Nummer, Richtung, Konverter und die Zeit des letzten Laufs. Ein- und Ausgabedatei zeigt der Tooltip.

- **Run** führt die Konfiguration sofort aus – derselbe Lauf wie im Betrieb.
- Ein Klick auf die Zeile öffnet sie auf der Seite „Konfigurationen“; „Neue Konfiguration“ legt dort eine für diesen Partner an.

Seite „Konfigurationen“



Die Seite „Konfigurationen“: links alle Konfigurationen, rechts die gewählte mit der Aufrufzeile für die Aufgabenplanung und ihren Feldern auf vier Reitern.

Links stehen **alle** Konfigurationen mit Partner, Richtung, Konverter und dem farbigen Punkt des letzten Laufs. Die Suche findet Nummer, Partner, Konverter und Dateinamen; darunter lässt sich nach Partner filtern. Rechts steht die gewählte Konfiguration: oben Nummer, Partner und die **Aufrufzeile** für die Aufgabenplanung mit einem Knopf zum Kopieren, darunter die Felder auf vier Reitern, unten **Speichern**, **Run**, **Archiv** und **Löschen**. „Run“ und „Archiv“ speichern vorher. Fehlt Partner oder Konverter oder ist eine Tageszahl keine Zahl, wechselt das Formular beim Speichern auf den Reiter mit diesem Feld.

Reiter „Allgemein“

- **Partner** – zu welchem Partner die Konfiguration gehört. Er lässt sich hier auch wechseln.
- **Richtung** – „Outbound“ (Inhouse → EDIFACT) oder „Inbound“ (EDIFACT → Inhouse).
- **Konverter** – welche Definition verwendet wird.
- **Absender-GLN** – Ihre eigene GLN, in der Definition als `~~ABSENDER-GLN` verwendbar. Nur nötig, wenn die Definition sie braucht; bei Eingabe wird die Prüfziffer geprüft.
- **Zähler (zuletzt vergeben)** – die zuletzt verwendete Nummer. Speichern ändert sie nicht; nur „Setzen“ schreibt sie ausdrücklich neu, etwa nach einem Jahreswechsel.

Reiter „Dateien“

- **Eingabedatei** – der vollständige Pfad der Datei, die gelesen wird. `{BASIS}` steht für den Installationsordner: `{BASIS}\Import\ORDER*.EDI` liest aus dem Ordner `Import` der Installation, wo immer sie liegt. Das gilt ebenso für Ausgabedatei, Archiv- und Logordner. Statt eines festen Namens darf hier ein **Muster** stehen, wenn Ihr Partner unter wechselnden Namen liefert:
  - `*` steht für beliebig viele Zeichen: `C:\EDI\ein\INVOIC_*.csv`

- `?` steht für genau ein Zeichen: `C:\EDI\ein\ORDERS_??????.edi`

Platzhalter sind nur im Dateinamen erlaubt, nicht im Ordner – wo gesucht wird, muss feststehen. **Ein Lauf verarbeitet eine Datei**, und zwar die **älteste** passende; so bleibt die Reihenfolge der Anlieferung erhalten. Liegen mehrere bereit, sagt die Meldung, wie viele noch warten – rufen Sie den Konverter dann einfach erneut auf, bis er mit Rückgabewert 5 meldet, dass nichts mehr da ist. Unter dem Feld zeigt der Editor, welche Datei als Nächstes dran wäre.

Zu einem Muster gehört das Archivieren (siehe unten): Nur wenn die verarbeitete Datei wegkommt, ist beim nächsten Lauf die nächste an der Reihe. Bleibt sie liegen, nimmt jeder Lauf wieder dieselbe; der Editor warnt davor.

- **Zeichensatz der Eingabe** – „Automatisch" (Vorgabe, erkennt ihn am Inhalt), „Latin1" oder „Utf8". UTF-16-Dateien werden abgewiesen.
- **Zeichensatz der Ausgabe** – „Latin1" (Vorgabe), „Utf8" (ohne Byte-Reihenfolge-Marke) oder „Automatisch", also so, wie die Eingabe gelesen wurde. In Latin-1 wird jedes Zeichen zu genau einem Byte; ein Zeichen, das es dort nicht gibt – etwa das Eurozeichen –, bricht die Wandlung ab, statt stillschweigend ein Fragezeichen zu schreiben.
- **Ausgabe als** – „Datei, wie die Definition sie schreibt" (Vorgabe) oder „ZUGFeRD-PDF": Dann entsteht aus der XML-Rechnung der Definition eine sichtbare Rechnung als PDF/A-3 mit eingebetteter XML, siehe „ZUGFeRD-Rechnungen".
- **Ausgabedatei** – ein **Muster** mit Platzhaltern:
  - `{ZAEHLER}` – laufende Nummer dieser Konfiguration: 1, 2, 3 …
  - `{ZAEHLER:6}` – dieselbe Nummer mit Nullen auf sechs Stellen: `000001`
  - `{DATUM}` – Tagesdatum `JJJJMMTT`
  - `{ZEIT}` – Uhrzeit `HHMMSS`
  - `{RECHNUNGSNUMMER}`, gleichbedeutend `{RG-NO}` – die Nummer der Rechnung, etwa `RECHNUNG_{RECHNUNGSNUMMER}.pdf`. Nur bei einer Rechnung: EDIFACT INVOIC (Nummer aus dem `BGM`) oder ZUGFeRD (aus der XML-Rechnung). Bei jeder anderen Ausgabe – oder bei mehreren Rechnungen mit verschiedenen Nummern in einer Datei – endet der Lauf mit Rückgabewert 7 und schreibt nichts. Zeichen, die in einem Dateinamen nicht erlaubt sind, etwa `/`, werden zu `_`.
  - `{BASIS}` – der Installationsordner

  Beispiel: `C:\EDI\aus\INVOIC_{DATUM}_{ZAEHLER:6}.edi`. Unter dem Feld steht die Vorschau des nächsten Namens. Ordner aus dem Muster werden angelegt. Enthält das Muster `{ZAEHLER}`, entsteht die Ausgabe immer; eine schon belegte Nummer wird übersprungen. Ohne `{ZAEHLER}` wird eine vorhandene Datei **nicht** überschrieben – der Lauf endet dann mit Rückgabewert 6. Das gilt auch für dieselbe Rechnung ein zweites Mal.

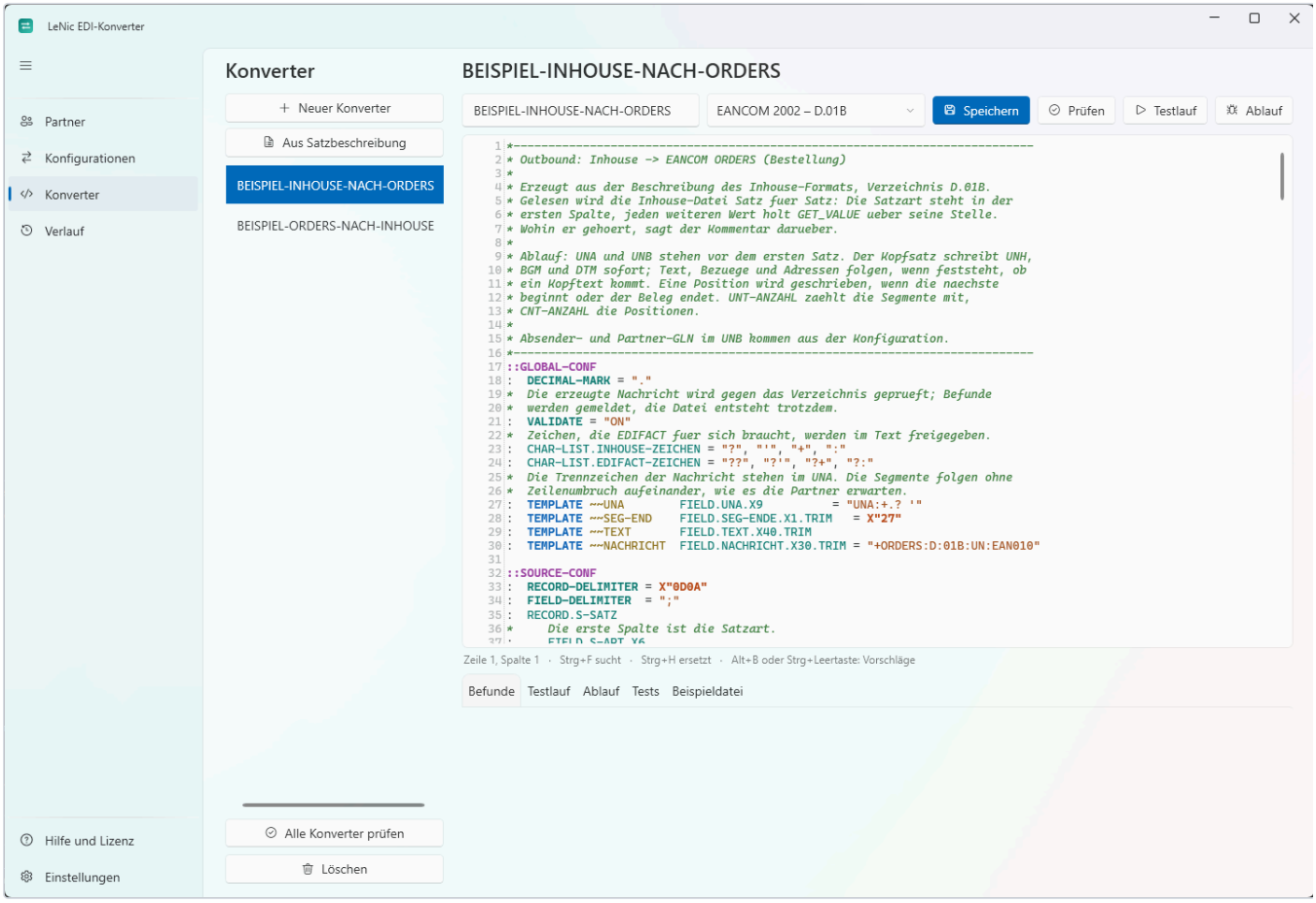
Reiter „Archiv & Log"

- **Archiv**
  - „Eingabedatei nach erfolgreicher Verarbeitung archivieren" – verschiebt die gelesene Datei nach dem Lauf ins Archiv, benannt wie `ORDERS_20260919_115102.edi`.
  - „Archivordner" – leer bedeutet: ein Ordner `Archiv` neben der Eingabedatei. Ein relativer Pfad gilt ab dort.
  - „Tage" – wie lange Archivdateien dieser Eingabe aufbewahrt werden; `0` heißt unbegrenzt.
  - „Archivdateien auch außerhalb des Basisverzeichnisses löschen" – siehe „Löschschutz". Ohne diesen Haken bleiben alte Archivdateien außerhalb des Installationsordners einfach liegen.
- **Log** – Ordner und Aufbewahrung in Tagen für die `.LOG`-Datei je Lauf. Ohne Angabe schreibt der Konverter sie nach `temp\` im Installationsordner.

Reiter „E-Mail"

- **E-Mail-Benachrichtigung** – „E-Mail senden" *Nie*, *Immer* oder *Bei Fehler*, dazu die Empfänger, getrennt mit `;`. *Immer* meldet jeden Lauf, *Bei Fehler* nur einen, der nicht erfolgreich war (Rückgabewert weder 0 noch 11). Ein Lauf, der keine Eingabedatei vorfindet (Rückgabewert 5), schickt nie eine E-Mail – sonst käme bei jedem Aufruf der Aufgabenplanung eine. Die E-Mail nennt Konfiguration, Partner, Rechner, Rückgabewert, Ein- und Ausgabedatei und die Meldung des Laufs. Versandt wird, wie unter Einstellungen → E-Mail-Versand eingestellt; ob die E-Mail hinausging, steht in der Meldung und im Verlauf. Ein gescheiterter Versand ändert den Rückgabewert nicht.

Seite „Konverter“



Die Konverterseite mit einer Definition. Die Farben zeigen Sektionen, Befehle, Felder und Muster.

Hier schreiben Sie die Definition, also die Vorschrift, wie gewandelt wird. Die Sprache ist in `docs/Sprachbeschreibung.md` beschrieben.

- **Neuer Konverter** beginnt mit einer kleinen, lauffähigen Vorlage.
- **Aus Satzbeschreibung** erzeugt einen **Entwurf**, ohne dass Sie eine Zeile der Definitionssprache schreiben müssen. Sie beschreiben nur Ihr Inhouse-Format – je Satzart eine Zeile mit der Kennung und den Namen der Spalten:

  ```
  K;Bestellnummer;Bestelldatum;Lieferdatum;Käufer;Warenempfänger;Name;Strasse;PLZ;Ort;Land;
  P;Positionsnummer;GTIN;Artikelnummer;Artikelbezeichnung;Menge;Einheit;
  E;Anzahl Positionen;
  ```

  Für **Bestellung, Lieferavis und Rechnung** steht beim Öffnen eine solche Vorlage im Feld; wechseln Sie den Nachrichtentyp, wechselt sie mit, solange Sie noch nichts Eigenes eingetragen haben. Mit **Datei wählen …** lesen Sie eine Beschreibung aus einer Datei.

  Woher die Werte kommen, weiß der Editor selbst: Die Segmente und Qualifier einer EANCOM-Nachricht sind genormt, und Ihr Spaltenname sagt, was gemeint ist. „Käufer", „Besteller" und „GLN Besteller" meinen dasselbe. Er schreibt daraus die vollständige Definition – Kopf, Positionen, Textsätze, Endsatz, samt Abgleich der Positionszahl mit dem `CNT`-Segment.

  Oben wählen Sie in drei Auswahlfeldern **Richtung**, **Directory** (die UN/EDIFACT-Ausgabe, etwa EANCOM 2002 – D.01B) und **Nachrichtentyp** (die Nachrichtenarten des gewählten Verzeichnisses).

  Dieselbe Beschreibung trägt beide Richtungen:

  - **Inbound** – liest die EDIFACT-Nachricht und schreibt Ihre Inhouse-Datei.
  - **Outbound** – liest Ihre Inhouse-Datei und schreibt die Nachricht: Umschlag (`UNA` , `UNB` , `UNZ`), Nachricht (`UNH` … `UNT`), Kopf, Positionen und Summenteil. Die Zahl der Segmente im `UNT` und die der Positionen im `CNT` zählt die

Definition beim Schreiben mit; Absender- und Partner-GLN im `UNB` kommen aus der Konfiguration. Freie Texte werden freigegeben, sodass ein Apostroph in einer Artikelbezeichnung das Segment nicht beendet.

Was der Name nicht verrät, wird nicht geraten: Das Feld entsteht, seine Zuweisung steht auskommentiert da, und die Meldung nennt die Spalte. Der Entwurf kommt **ungespeichert** in den Editor – Namen geben, ergänzen, prüfen, speichern.

Einen Vorschlag gibt es für diese Nachrichtentypen, in jedem beiliegenden Verzeichnis, das sie führt:

Nachrichtentyp	Beleg	Menge	Besonderheiten
ORDERS – Bestellung	`BGM+220`	`QTY+21`	Die Bestellnummer ist die Belegnummer.
INVOIC – Rechnung	`BGM+380`	`QTY+47`	Bezug auf Bestellung (`RFF+ON` , `DTM+171`) und Lieferschein (`RFF+DQ`), als Sammelrechnung auch je Position (`RFF+DQ` , `DTM+171`); Käufer, Lieferanschrift, Lieferant und Rechnungsempfänger je mit Anschrift und USt-IdNr. (`NAD` , `RFF+VA`); Währung, Preis (`PRI`), Positionsbetrag (`MOA+203`), Steuersatz (`TAX`), ein Steuersatz im Kopf als Vorgabe; Rabatte auf Beleg und Position (`ALC` , `PCD` , `MOA+204`); im Summenteil Rechnungsbetrag, Warenwert, Steuerbasis und Steuerbetrag (`MOA+77` , `79` , `125` , `124`) und je Steuersatz eine Steuerzeile.
DESADV – Lieferavis	`BGM+351`	`QTY+12`	Versand- und Lieferdatum (`DTM+11` , `DTM+17`); die Sendung (`CPS+1`) mit Zahl der Paletten und Gesamtgewicht, je Palette eine eigene Stufe mit Gewicht und SSCC (`GIN+BJ`), darunter die Positionen mit Charge und Mindesthaltbarkeitsdatum (`GIN+BX` , `DTM+361`). Ohne Preise und ohne `UNS` .
ORDRSP – Bestellbestätigung	`BGM+231` , Funktion 29	`QTY+113`	Jede Position wird mit `LIN+…+5` angenommen.

Hat eine Spalte in der gewählten Nachricht keinen Platz – etwa ein Preis im Lieferavis –, entsteht das Feld, und der Entwurf vermerkt „in DESADV nicht vorgesehen“. Die Reihenfolge der Segmente kommt aus dem gewählten Verzeichnis. Im `UNH` steht bei D.96A und D.01B das EANCOM-Kennzeichen (Bestellung `EAN008` / `EAN010` , Rechnung `EAN008` / `EAN011` , Lieferavis `EAN005` / `EAN007`), bei den übrigen Ausgaben und bei der Bestellbestätigung keines. In D.96A wird die GTIN als `EN` gekennzeichnet und die Artikelbezeichnung nach dem Aufbau dieser Ausgabe geschrieben (`IMD+F++:::…`).

Die erste Spalte jeder Inhouse-Zeile ist die **Satzart** (im Beispiel `K` , `P` , `E`). Der Entwurf legt dafür die Vorlage `~~SATZART` an: Jeder Satz beginnt mit ihr, und sie darf bis zu sechs Zeichen lang sein. Eine Satzart darf in der Beschreibung mehrmals stehen, um die Verschachtelung zu zeigen (Palette, Positionen, Palette, Positionen).

Manches entscheidet der **Zusammenhang**: `Name` , `Strasse` … und eine Spalte `MwST` gehören im Kopf zu der Partei, deren GLN davorsteht; ein Steuersatz im Kopf gilt für Positionen ohne eigenen; ein Rabattsatz hinter einer Position betrifft diese Position. Einzelheiten stehen im Teil „Das Mapping“.

Datumsangaben: Bestell-, Liefer- und Mindesthaltbarkeitsdatum stehen in Ihrer Inhouse-Datei beim Lesen einer Nachricht als `TT.MM.JJJJ` ; beim Schreiben einer Nachricht liest der Entwurf sie als `TTMMJJJJ` – mit oder ohne Punkte – und stellt sie mit `FORMAT-DATE` auf das EDIFACT-Format um. Andere Formate stellen Sie in der Definition selbst ein, etwa `FORMAT-DATE (GET_VALUE (3), "CCYY-MM-DD", "102")` .

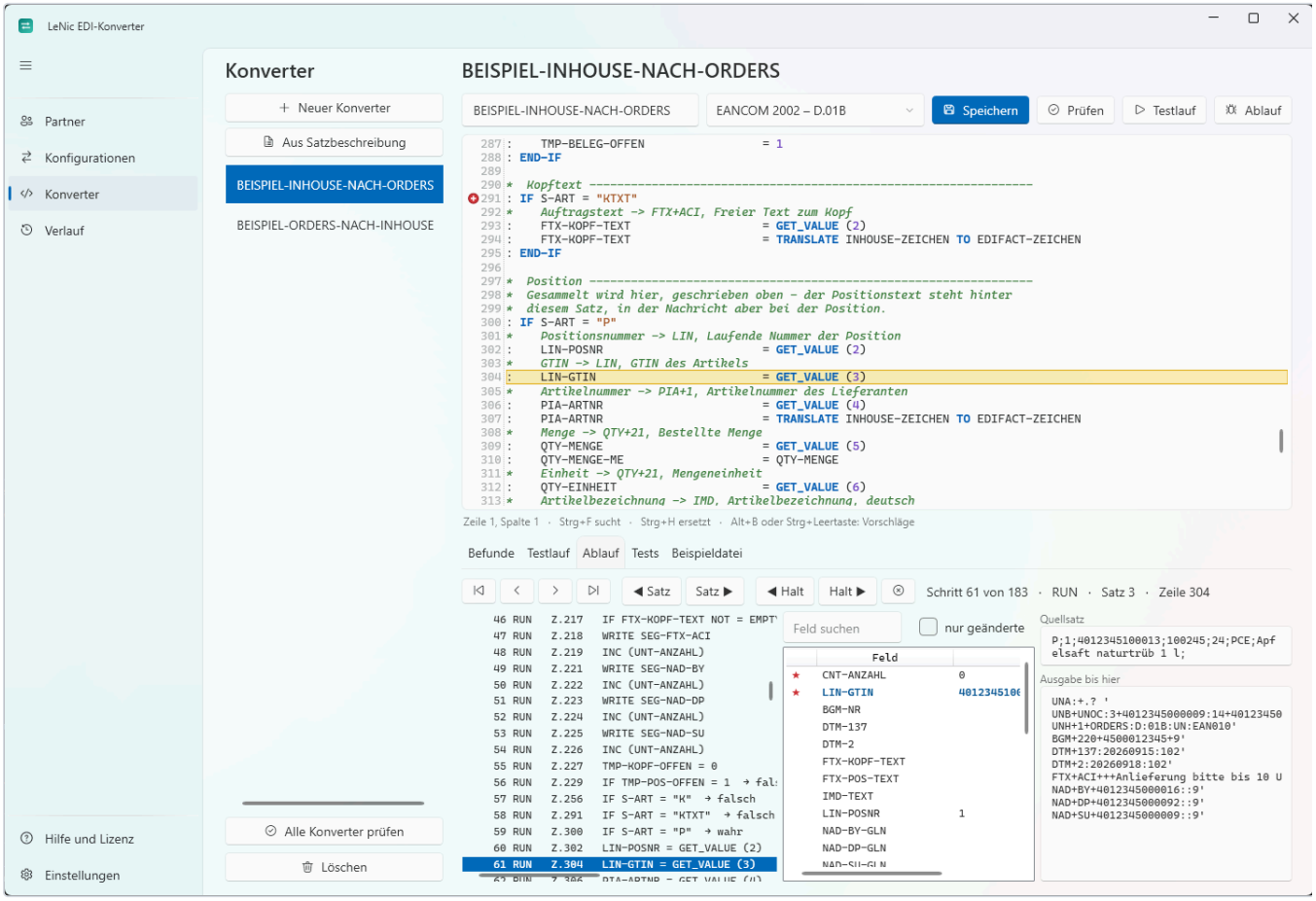
- **Prüfen** liest die Definition und zeigt alle Befunde auf einmal, mit Zeilennummer. Doppelklick auf einen Befund springt zur Zeile – oder setzt einen Vorschlag ein, wenn es einen gibt.
- **Testlauf** wandelt eine Beispieldatei und zeigt das Ergebnis, wahlweise mit einem Umbruch nach jedem Segment. Dabei wird **keine Datei geschrieben** und nichts archiviert. Für `~~PARTNER-GLN` und `~~PARTNER-NAME` wählen Sie einen Partner, die Absender-GLN tragen Sie daneben ein.
- **Ablauf** verfolgt den Testlauf Schritt für Schritt wie ein Debugger, mit Haltepunkten und beobachteten Feldern – siehe „Der Ablauf“ unten.
- **Tests** hält Eingaben mit dem Ergebnis fest, das der Konverter dafür liefern soll, und prüft, ob er es nach einer Änderung noch tut – siehe „Regressionstests“ unten.

- **Beispieldatei** zeigt die Testdatei Satz für Satz. Setzen Sie die Schreibmarke auf den Wert, den Sie lesen wollen: Der Editor schlägt das Muster vor (etwa `NAD+BY+VALUE`) und setzt es auf Knopfdruck oder mit einem Doppelklick in die Definition ein.
- **Erklärung beim Zeigen:** Fahren Sie mit der Maus über ein Muster oder eine Segmentkennung, erklärt Ihnen der Editor das Segment, zeigt farbig, welche Stelle gelesen wird, und nennt die Bedeutung der Qualifier aus dem gewählten UN/EDIFACT-Verzeichnis.
- **Verzeichnis** (oben rechts neben dem Namen) – welche Verzeichnisausgabe für Erklärungen und Muster gilt: automatisch aus dem UNH der Testdatei oder fest, etwa „EANCOM 1997 – D.96A". Mitgeliefert sind die Ausgaben D.96A bis D.06B.

Tastenkürzel im Definitionsfeld

- `Strg+F` suchen, `F3` weiter, `Umschalt+F3` zurück
- `Strg+Alt+↓` / `Strg+Alt+↑` – zum nächsten bzw. vorigen Vorkommen des Wortes an der Schreibmarke (Feld, Variable, Befehl) oder des markierten Textes; ein ganzes Wort wird nur als ganzes Wort gefunden. Am Ende bzw. Anfang ist Schluss, ein Hinweis sagt, dass nichts mehr kommt
- `Strg+H` ersetzen; „Alle" ersetzt in einem Schritt, der sich mit `Strg+Z` zurücknehmen lässt
- `Alt+B` oder `Strg+Leertaste` – Vorschläge für Befehle und Namen der Definition; `Esc` schließt sie
- `Strg+Z` / `Strg+Y` – rückgängig und wiederherstellen

Der Ablauf – Fehlersuche Schritt für Schritt



Der Ablauf eines Testlaufs: links die Schritte, in der Mitte die Felder – beobachtete mit ★ oben –, rechts Quellsatz und Ausgabe. Gelb die Zeile des Schritts, der rote Punkt mit Plus ein bedingter Haltepunkt.

Der Knopf **Ablauf** verfolgt den Testlauf Schritt für Schritt – wie ein Debugger. Aufgezeichnet wird mit derselben Eingabedatei, demselben Partner und derselben Absender-GLN wie beim Testlauf; danach zeigt der Reiter „Ablauf":

- links jeden Schritt mit Abschnitt (START, RUN, END), Zeile, Anweisung und Ergebnis – bei einer Bedingung „wahr“ oder „falsch“; ein neuer Quellsatz steht fett,
- in der Mitte alle Felder mit ihrem Wert **nach** diesem Schritt; was der Schritt gesetzt hat, ist hervorgehoben („nur geänderte“ zeigt nur diese),
- rechts den Quellsatz und die Ausgabe bis hierher.

Die Zeile des Schritts ist in der Definition gelb hinterlegt. Mit den Pfeilen oder `F10` geht es einen Schritt weiter, mit `Umschalt+F10` zurück; `F11` und `Umschalt+F11` springen zum nächsten oder vorigen Quellsatz. Bricht die Wandlung ab, steht der Ablauf gleich auf dem Schritt, der den Fehler auslöste. Es wird nichts geschrieben und nichts archiviert. Aufgezeichnet werden höchstens 200.000 Schritte; bei sehr großen Testdateien sagt der Editor, dass er gekürzt hat.

Haltepunkte: Ein Klick links neben die Zeilennummer setzt einen roten Punkt, ein zweiter entfernt ihn; `F9` tut dasselbe für die Zeile mit der Schreibmarke (im Reiter „Ablauf“ für die Zeile des aktuellen Schritts). Die Punkte wandern beim Bearbeiten mit ihrer Zeile; wird die Zeile gelöscht, verschwindet der Punkt. `F5` oder „Halt ▶“ springt zum nächsten Schritt in einer Zeile mit Haltepunkt, `Umschalt+F5` oder „◀ Halt“ zum vorigen. Gibt es Haltepunkte, beginnt ein neuer Ablauf gleich beim ersten Treffer. Der Knopf mit dem Kreuz entfernt alle Haltepunkte. Weil der Ablauf vollständig aufgezeichnet ist, darf man Haltepunkte auch nachträglich setzen und vor- wie rückwärts springen.

Bedingte Haltepunkte: Ein Rechtsklick auf den Rand öffnet ein Feld für eine Bedingung – in derselben Form wie hinter `IF`, auch mit `AND` und `OR`:

```
S-ART = "P"
LIN-GTIN = "4012345100013"
MENGE > 10 AND NAD-DP-GLN <> EMPTY
```

`Enter` übernimmt, `Esc` bricht ab, ein leeres Feld hält wieder immer. Geprüft wird **vor** der Zeile, mit dem Stand der Felder davor – wie in Visual Studio. Ein Punkt mit Bedingung trägt ein weißes Plus, der Tipp über dem Punkt nennt die Bedingung. Kommt ein Feldname im Ablauf nicht vor (etwa ein Tippfehler), warnt der Editor beim Springen.

Felder beobachten: Ein Rechtsklick auf ein Feld in der Feldliste bietet

- **Beobachten** – das Feld steht mit ★ immer oben, auch wenn der Filter oder „nur geänderte“ es sonst ausblenden würde;
- **Bei Änderung anhalten** – `F5` hält bei jedem Schritt, der dem Feld einen anderen Wert gibt (●); auch das Leeren nach einem `WRITE` zählt. Die Statuszeile nennt dann „ändert …“. Wer denselben Wert noch einmal setzt, hält nicht.

Der Knopf mit dem Kreuz entfernt auch die Haltepunkte bei Feldern.

Haltepunkte bleiben: Haltepunkte samt Bedingungen, beobachtete Felder und Haltepunkte bei Feldern gehören zum Konverter. Sie werden bei jeder Änderung gespeichert und sind beim nächsten Öffnen des Konverters wieder da – auch nach einem Neustart des Editors. Ein Haltepunkt merkt sich den Text seiner Zeile: Ist die Zeile durch Änderungen an der Definition gewandert, findet er sie wieder; gibt es sie nicht mehr, fällt er weg.

Einen Lauf aus dem Betrieb nachstellen: Auf der Seite „Verlauf“ stellt **Im Ablauf nachstellen** (Knopf oder Rechtsklick) einen Lauf im Debugger nach – mit seiner Eingabedatei, und zwar der archivierten, sonst der ursprünglichen, mit Partner, Absender-GLN und Zeichensatz seiner Konfiguration. Bricht die Wandlung ab, steht der Ablauf gleich auf dem Schritt, der den Fehler auslöste. Nachgestellt wird mit der Definition, wie sie **heute** ist; wurde der Konverter seit dem Lauf geändert, kann das Ergebnis abweichen. Scheiterte der Lauf an etwas anderem als der Wandlung – etwa einer fehlenden Ausgabeordner-Berechtigung –, sagt der Hinweis das. Gibt es die Eingabedatei nicht mehr, ist der Knopf ausgegraut.

Regressionstests

Ein Regressionstest hält eine Eingabe fest und das Ergebnis, das der Konverter dafür liefern soll. Nach jeder Änderung – am Konverter oder durch eine neue Version des Programms – zeigt ein Vergleich, ob er noch dasselbe liefert. Die Tests stehen im Reiter **Tests** der Konverterseite:

- **Als Test merken** legt einen Test an: mit der Eingabedatei, dem Partner und der Absender-GLN aus dem Reiter „Testlauf“. Was der gespeicherte Konverter jetzt daraus macht, gilt fortan als erwartet – auch ein Wandlungsfehler oder ein Anwenderfehler kann ein erwartetes Ergebnis sein. Die Eingabe wird im Test selbst gespeichert; die Datei darf danach verschwinden.

- **Prüfen** wandelt jede Eingabe mit der Definition im Editor und vergleicht Zeichen für Zeichen. Das geht auch vor dem Speichern: So sehen Sie, was eine Änderung bewirkt, bevor sie im Betrieb ankommt. ✓ heißt bestanden, ✗ abweichend; rechts steht die erste abweichende Zeile – bei EDIFACT das Segment – mit erwartetem und jetzigem Inhalt, darunter beide Ergebnisse ganz.
- **Erwartung übernehmen** macht das jetzige Ergebnis eines abweichenden Tests zum erwarteten – für eine gewollte Änderung. Das geht nur bei gespeichertem Konverter.
- Der Papierkorb löscht den gewählten Test.

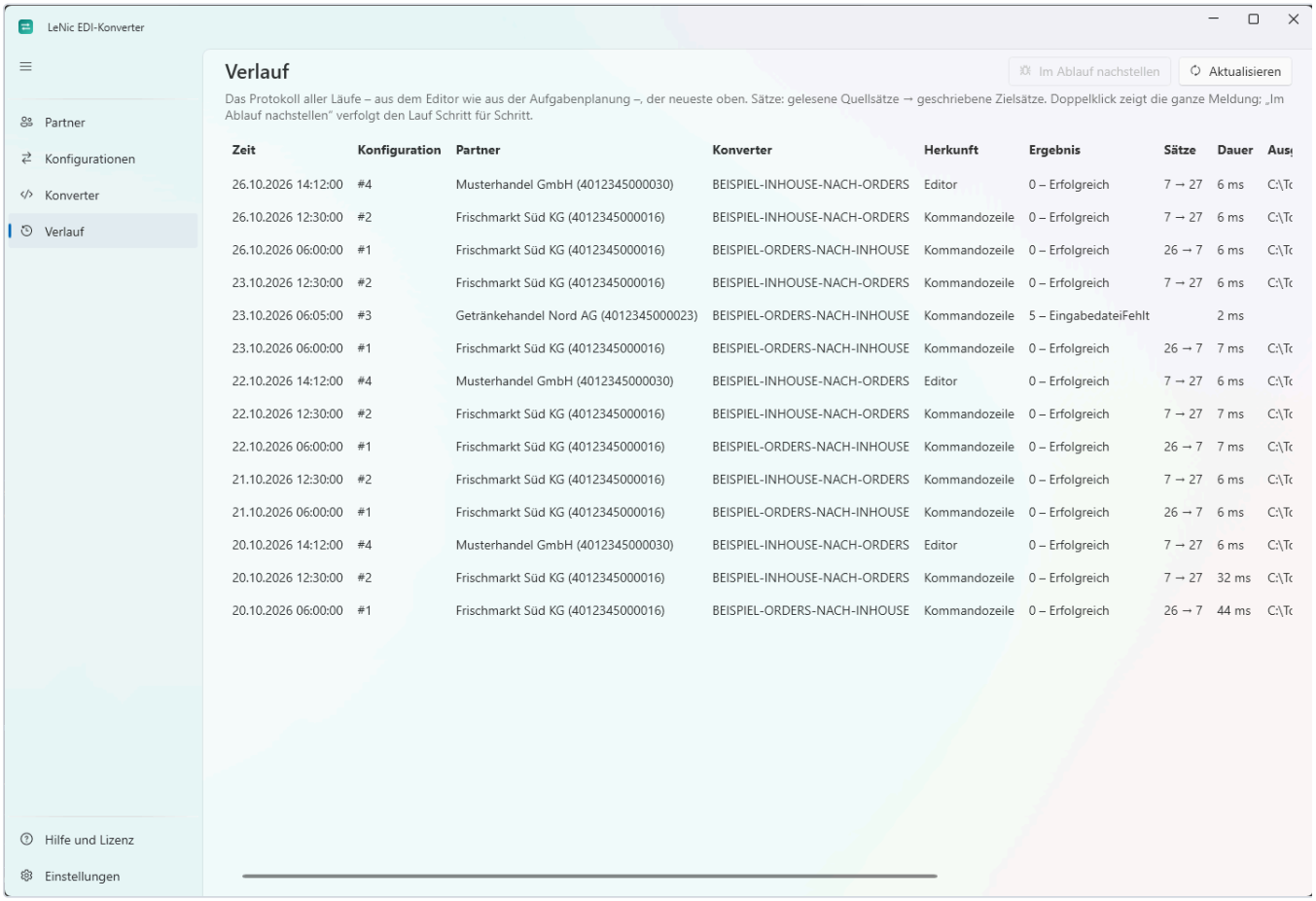
Datum und Uhrzeit: Ein Test hält beim Anlegen Datum und Uhrzeit fest („Stichtag") und wandelt immer damit. `DATE` und `TIME` – etwa im UNB – ergeben so bei jeder Prüfung dasselbe, statt jeden Test rot zu färben.

Alle Konverter prüfen (links unter der Konverterliste) prüft die Tests aller Konverter mit ihrer **gespeicherten** Definition, die Abweichungen zuerst – etwa vor einem Update beim Kunden. Ein Doppelklick öffnet den Konverter im Reiter „Tests".

Tests gehören zum Konverter: Wird er umbenannt, ziehen sie mit, wird er gelöscht, gehen sie mit. Sie liegen in der Datenbank und werden mit ihr gesichert.

Meldungen in einem Testergebnis folgen der Sprache, die beim Anlegen eingestellt war. Wird die Sprache umgestellt, kann deshalb ein Test abweichen, der einen Wandlungsfehler erwartet – „Erwartung übernehmen" übernimmt die Meldung in der neuen Sprache.

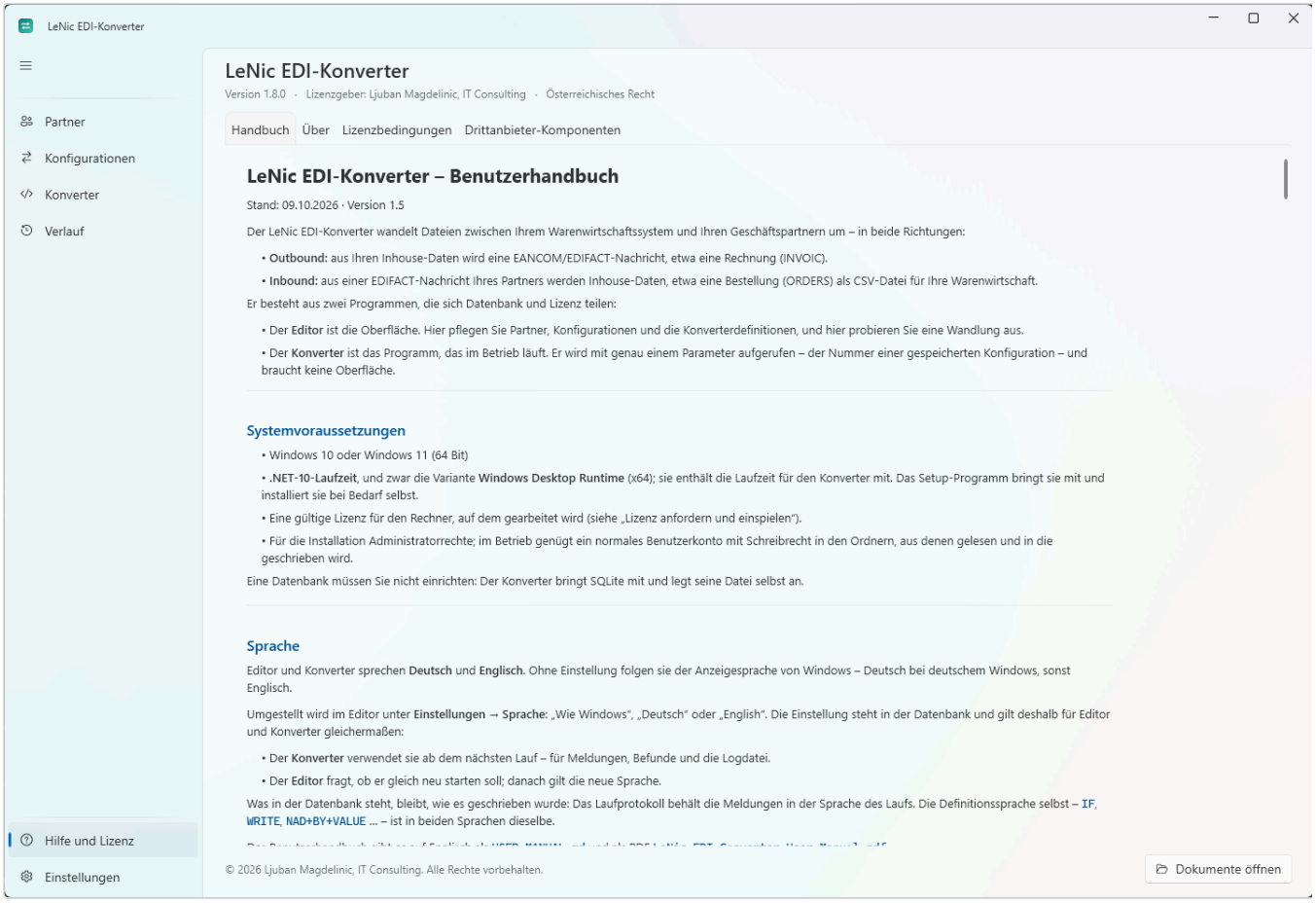
Seite „Verlauf"



Der Verlauf: jeder Lauf mit Zeit, Konfiguration, Partner, Konverter, Herkunft und Rückgabewert.

Das Protokoll aller Läufe – aus dem Editor wie aus der Aufgabenplanung –, der neueste oben, mit Zeit, Konfiguration, Partner, Konverter, Herkunft, Ergebnis, Anzahl der gelesenen und geschriebenen Sätze, Dauer, Ausgabe- und Archivdatei. Ein Doppelklick zeigt die ganze Meldung. **Im Ablauf nachstellen** verfolgt den gewählten Lauf im Debugger – siehe „Der Ablauf".

Seite „Einstellungen“



> Dieses Handbuch steckt auch im Editor: unter „Hilfe und Lizenz“, Reiter „Handbuch“.

Die Einstellungen stehen auf drei Reitern.

Reiter „Allgemein“

- **Darstellung** – Wie Windows, Hell oder Dunkel.
- **Sprache** – Wie Windows, Deutsch oder English; siehe „Sprache“.
- **Globale Konfiguration** – „Laufprotokoll löschen nach (Tagen)“. Vorgabe 90, `0` heißt nie, höchstens 3650. Gilt für alle Konfigurationen; Speichern räumt sofort auf.
- **Über** – die Version des Editors.

Reiter „Daten & Lizenz“

- **Datenbank** – wo sie liegt, mit Knopf „Im Explorer zeigen“.
- **Beispiele** – „Beispiele übernehmen …“ ergänzt die mitgelieferten Beispiele dieser Fassung, soweit sie in Ihrer Datenbank fehlen.
- **Lizenz** – Stand, Rechnername und das Einspielen einer neuen Lizenzdatei.

Reiter „E-Mail-Versand“

- **E-Mail-Versand** – wie die Benachrichtigungen hinausgehen, für alle Konfigurationen gleich, dazu die **Absenderadresse**:
  - *SMTP-Server* – Server, Port (Vorgabe 587), Verschlüsselung mit STARTTLS, Benutzer und Kennwort (Benutzer leer = ohne Anmeldung). Port 465 (sofort verschlüsselt) wird nicht unterstützt – nehmen Sie 587.
  - *Microsoft 365 (Graph)* – Tenant-ID, Anwendungs-ID und geheimer Clientschlüssel einer App-Registrierung; siehe „E-Mail über Microsoft 365“. Microsoft 365 nimmt über SMTP keine Anmeldung mit Kennwort mehr an.

  Kennwort und Clientschlüssel werden verschlüsselt und an diesen Rechner gebunden gespeichert; auf einem anderen Rechner – etwa nach dem Umzug der Datenbank – geben Sie sie neu ein. Ein leeres Feld behält den gespeicherten Wert.

„Test-E-Mail senden“ schickt mit den eingetragenen Werten eine E-Mail an die Absenderadresse, auch vor dem Speichern.

1.9 ZUGFeRD-Rechnungen

Der Konverter erzeugt aus einer Inhouse-Datei Ihrer Warenwirtschaft eine **ZUGFeRD-Rechnung** (auch Factur-X): ein PDF, das Menschen lesen und in dem dieselbe Rechnung maschinenlesbar als XML steckt. Das PDF erfüllt **PDF/A-3**, die XML folgt dem Profil **EN 16931** nach UN/CEFACT CII.

Zum Ausprobieren gibt es Konfiguration 7 mit dem Konverter **ZUGFERD-CSV** (siehe „Mitgelieferte Beispiele“): beim Beispielpartner **Run** drücken, und in `Export\` liegt `ZUGFERD_OUT_1.PDF`. Fehlt sie in einer älteren Installation, holen Sie sie unter **Einstellungen → Daten & Lizenz → Beispiele → „Beispiele übernehmen …“**.

Ein ZUGFeRD-Konverter enthält alles, was er braucht:

- die **Ausgabevorlage** in LAYOUT-CONF – die Rechnung selbst, fest für ein Profil, mit ihren Feldern in TARGET-CONF (siehe „Ausgabevorlagen“),
- **Quelle und Zuordnung** in SOURCE-CONF und RUN-CONF – sie lesen Ihre Datei und füllen die Felder. Nur diesen Teil passen Sie an.

Für Ihre eigenen Rechnungen:

1. **Definition:** Beim Konverter liegt ein Muster bei, `Konverter\Samples\ZUGFERD-CSV.DEF`, mit der Beispielrechnung `ZUGFERD-EN16931.csv`; daneben `ZUGFERD-EN16931-RABATTE.csv` mit Preisnachlass, mehreren Rabatten und einem Zuschlag auf eine Position, Treuerabatt und Fracht und `ZUGFERD-EN16931-KORREKTUR.csv` mit einem Storno. Legen Sie damit auf der Seite „Konverter“ einen neuen Konverter an und passen Sie SOURCE-CONF und RUN-CONF an den Aufbau Ihrer Datei an. Die Kommentare am Anfang beschreiben, welche Angaben die Datei braucht.
2. **Konfiguration: Ausgabe als** auf „ZUGFeRD-PDF“ stellen und eine Ausgabedatei auf `.pdf` angeben, etwa `{BASIS}\Export\RECHNUNG_{ZAEHLER}.pdf`.

Was dabei geschieht:

- Die Definition füllt die Vorlage, daraus wird die XML-Rechnung. Aus **dieser XML** entsteht die sichtbare Seite – Kopf, Anschrift, Positionen, Steuer je Satz, Summen und Zahlung. PDF und XML zeigen deshalb sicher dieselbe Rechnung.
- Die XML wird als `factur-x.xml` eingebettet, **immer in UTF-8**, gleich welcher Zeichensatz der Ausgabe eingestellt ist.
- **Eine Rechnung je Datei.** Stehen zwei in einer Eingabedatei, meldet das Muster einen Fehler und schreibt nichts.
- Die **Summen** kommen aus Ihrer Warenwirtschaft. Der Konverter rechnet nur die Summen der Nachlässe und Zuschläge aus den A-Sätzen und prüft, dass die Steuerbasis zu Positionen, Nachlässen und Zuschlägen passt; sonst meldet er einen Fehler und schreibt nichts.
- Ist die Ausgabe der Definition keine CII-Rechnung, endet der Lauf mit Rückgabewert 7 (Wandlungsfehler) und einer Meldung, was fehlt.

Das Muster kennt:

- **Rechnung, Gutschrift und Rechnungskorrektur:** Die Rechnungsart im K-Satz ist 380, 381 oder 384. Eine Korrektur oder ein Storno nennt im **R-Satz** Nummer und Datum der berichtigten Rechnung; ein Storno trägt negative Mengen und Beträge. Das PDF heißt dann „Gutschrift“ bzw. „Rechnungskorrektur“ und nennt die berichtigte Rechnung.
- **Rabatte und Zuschläge auf eine Position:** ein **Preisnachlass** je Einheit am P-Satz (Bruttopreis und Nachlass, der Nettopreis ist der Preis danach) und **beliebig viele Nachlässe und Zuschläge** als **N-Sätze** direkt hinter dem P-Satz – Betrag und wahlweise Prozent, mit Grund. Der Nettowert der Position ist der Wert nach allen N-Sätzen.
- **Nachlässe und Zuschläge auf die ganze Rechnung** – Treuerabatt, Fracht, Verpackung – als **A-Sätze**, je mit Steuersatz und Grund.

Rabatte und Zuschläge stehen auch auf der sichtbaren Rechnung: unter der Position bzw. zwischen „Summe netto“ und der Steuer.

Das Muster deckt Steuerkategorie „S“ ab (Regelsteuersatz, auch mehrere Sätze) und Zahlung per Überweisung. Steuerbefreite Rechnungen sowie die **XRechnung** für öffentliche Auftraggeber (Leitweg-ID, Ansprechpartner, elektronische Adressen) brauchen Ergänzungen von Vorlage und Definition.

Prüfen lässt sich eine erzeugte Rechnung mit frei verfügbaren Werkzeugen wie dem Mustang-Validator; er prüft PDF/A und die Regeln der EN 16931.

Ausgabevorlagen

Liefert mehr als eine Warenwirtschaft Rechnungen – oder kommen sie als EANCOM-Nachricht –, muss die ZUGFeRD-Rechnung nicht für jede Quelle neu geschrieben werden. Eine **Ausgabevorlage** ist die Rechnung selbst, als XML mit Platzhaltern wie `{KO-NUMMER}`. Sie steht in der Definition selbst, in LAYOUT-CONF, zusammen mit `FORMAT = LAYOUT` und ihren Feldern in TARGET-CONF; RUN-CONF füllt nur diese Felder. In welcher Reihenfolge Ihre Datei Kopf, Positionen und Summen liefert, spielt dabei keine Rolle – die Vorlage ordnet. Was eine Vorlage kann und wie man eine schreibt, steht in der Sprachbeschreibung unter „Ausgabevorlage“.

Mitgeliefert werden zwei Vorlagen mit denselben Feldern, als **Kopiervorlagen** unter `Konverter\Samples` – sie sind keine Konverter, sondern werden in eine Definition übernommen:

Kopiervorlage	Profil	Lieferscheine
`ZUGFERD-VORLAGE-EN16931.LAYOUT`	EN 16931 – auch Grundlage der XRechnung	jeder Lieferschein einmal als Beleg im Kopf
`ZUGFERD-VORLAGE-EXTENDED.LAYOUT`	EXTENDED	an **jeder Position**, mit Bestellung, Position darin und Datum – die Sammelrechnung über mehrere Lieferscheine

EN 16931 kennt keinen Lieferschein an der Position; für eine Sammelrechnung, die jede Position ihrem Lieferschein zuordnet, braucht es EXTENDED. Das nehmen aber nicht alle Empfänger an – öffentliche Auftraggeber verlangen meist XRechnung. Weil beide Vorlagen dieselben Felder haben, wechseln Sie das Profil, indem Sie LAYOUT-CONF durch die der anderen Kopiervorlage ersetzen. Die sichtbare Rechnung zeigt Bestellung und Lieferschein unter der Position bzw. die Lieferscheine im Kopf.

Dazu zwei Definitionen als Muster, beide auch als Beispielkonfiguration:

- `Konverter\Samples\ZUGFERD-CSV.DEF` (EN 16931) liest das Inhouse-CSV – mit drei wahlweisen Spalten am P-Satz für Lieferschein, Position und Datum. Ein N-Satz gehört zum P-Satz davor; die übrigen Sätze dürfen in beliebiger Reihenfolge kommen, die Vorlage ordnet.
- `Konverter\Samples\ZUGFERD-INVOIC.DEF` (EXTENDED) liest eine EANCOM-Rechnung (D.01B) mit Bestellung (`RFF+ON`) und Lieferschein (`RFF+DQ`) je Position.

1.10 Daten aus dem SQL Server (SQL-LINK)

Fehlt in der Eingabedatei etwas, das in Ihrer Warenwirtschaft steht – die Bezeichnung zu einem Artikel, die Lieferanschrift zu einer Kundennummer –, holt der Konverter es mit `SQL-LINK` direkt aus einem **Microsoft SQL Server**. Das Ergebnis steht danach als temporäre Tabelle bereit und wird mit `WHILE NEXT` gelesen. Die genaue Schreibweise steht in der Sprachbeschreibung (`docs/Sprachbeschreibung.md` , Abschnitt 4).

1. Die Verbindung eintragen. Jeder Konverter trägt seine Verbindung selbst, in einer eigenen Sektion `::LINK-CONF` :

```
::LINK-CONF
: SQL-CONNECTION = "Server=sql01;Database=WaWi;User Id=edi_leser;TrustServerCertificate=True"
: SQL-PASSWORD   = "Ihr Kennwort"
```

Schreiben Sie das Kennwort ruhig im Klartext hin: Beim **Speichern** – und vor einem Testlauf – ersetzt der Editor es durch eine verschlüsselte Form (`"dpapi:…"`). In der Datenbank steht es nie lesbar. Die Verschlüsselung ist an den Rechner gebunden; kopieren Sie den Konverter auf einen anderen Rechner, geben Sie das Kennwort dort neu ein und speichern. Ohne `SQL-PASSWORD` meldet sich der Konverter mit dem Windows-Konto an, unter dem er läuft – dann gehört `Integrated Security=True` in die Verbindung.

2. Die Tabelle anlegen. In SOURCE-CONF, je Spalte des Ergebnisses ein Feld, in derselben Reihenfolge:

```
: TEMP-TABLE.ARTIKEL
:    FIELD.A-EAN.X13
:    FIELD.A-TEXT.X35
```

3. Abfragen. In RUN-CONF (oder START-/END-CONF); Werte aus der Definition stehen in geschweiften Klammern:

```
: SQL-LINK ARTIKEL = "SELECT EAN, Bezeichnung FROM dbo.Artikel WHERE KundenNr = {KDNR}"
: WHILE NEXT ARTIKEL
:    …
: END-WHILE
```

Gut zu wissen:

- **SQL-LINK liest nur.** Erlaubt ist genau ein `SELECT` – kein `INSERT`, `UPDATE`, `DELETE`, kein `EXEC`, kein Semikolon, keine Kommentare. Der Editor meldet alles andere schon beim Speichern als Befund. Im Betrieb läuft jede Abfrage zudem in einer Transaktion, die immer zurückgenommen wird.
- **Richten Sie trotzdem einen eigenen Datenbankbenutzer ein, der nur lesen darf** (Rolle `db_datareader`, oder nur `SELECT` auf die nötigen Tabellen und Sichten). Das ist der wirksamste Schutz.
- Ein Wert in `{…}` geht als Parameter an den Server; Zeichen wie `'` oder `;` in Ihren Daten können die Abfrage nicht verändern.
- Heißt eine Spalte wie ein SQL-Befehl, etwa `Update`, schreiben Sie sie in eckigen Klammern: `[Update]`.
- Jedes `SQL-LINK` in RUN-CONF fragt einmal je Quellsatz. Was für alle Sätze gleich ist, holen Sie besser einmal in START-CONF.
- Ist der Server nicht erreichbar oder scheitert die Abfrage, bricht der Lauf mit Rückgabewert **7** ab; die Meldung im Log nennt die Tabelle und den Text des Servers.

1.11 Der Aufruf im Betrieb

Der Konverter wird mit **genau einem Parameter** aufgerufen: der Nummer der Konfiguration. Alles Weitere steht in der Konfiguration.

```
C:\Tools\LeNicEdiKonverter\Konverter\LeNic.EdiKonverter.App.exe 1
```

Die Nummer steht im Editor vor jeder Konfiguration (`#1`), die fertige Zeile auf der Seite „Konfigurationen“ oben rechts, mit einem Knopf zum Kopieren.

Automatischer Aufruf

Empfohlen ist das Produkt **LeNicScheduler**: Dort legen Sie eine Aufgabe mit dem Programm und der Nummer als Parameter an und bekommen Zeitsteuerung, Überwachung und Benachrichtigung dazu.

Ebenso geht die **Windows-Aufgabenplanung**:

- Programm/Skript: `C:\Tools\LeNicEdiKonverter\Konverter\LeNic.EdiKonverter.App.exe`
- Argumente: `1`
- Ausführen mit einem Konto, das die Ein- und Ausgabeordner erreicht.

Der Rückgabewert des Programms ist zugleich sein Exit-Code – beide Werkzeuge erkennen daran, ob der Lauf gelungen ist.

1.12 Rückgabewerte

- **0 – erfolgreich.** Die Ausgabedatei ist geschrieben.

- **1 – falscher Aufruf.** Es fehlt die Nummer der Konfiguration, oder sie ist keine Zahl.
- **2 – Konfiguration nicht gefunden.** Nummer prüfen; im Editor steht sie vor jeder Konfiguration.
- **3 – Konverter nicht gefunden.** Die Konfiguration verweist auf eine Definition, die es nicht mehr gibt.
- **4 – Fehler in der Definition.** Im Editor „Prüfen" drücken; die Befunde nennen die Zeile.
- **5 – Eingabedatei fehlt.** Pfad prüfen; kam die Datei des Partners an? Bei einem Muster heißt der Wert: Im Ordner liegt zurzeit nichts Passendes mehr – bei einem geplanten Aufruf ist das der Normalfall und kein Grund zur Sorge.
- **6 – Ausgabedatei bereits vorhanden.** Nur bei einem festen Namen ohne `{ZAEHLER}`. Alte Datei wegräumen oder das Muster um `{ZAEHLER}` ergänzen.
- **7 – Wandlungsfehler.** Etwa ein Wert, der länger ist als sein Feld, oder ein Zeichen, das der Zeichensatz der Ausgabe nicht kennt. Die Meldung sagt, wo. Mit „Testlauf" im Editor nachstellen.
- **8 – Datei nicht lesbar oder nicht schreibbar.** Rechte prüfen, oder ob die Datei gerade von einem anderen Programm gehalten wird.
- **9 – Einstellungen oder Datenbank nicht zu öffnen.** Liegt `data\` am erwarteten Ort? Stammt die Datenbank von einer neueren Programmfassung?
- **10 – Anwenderfehler.** Die Definition hat `ERROR` gemeldet; es wurde nichts geschrieben. Die Meldungen stehen im Log.
- **11 – erfolgreich mit Warnungen.** Geschrieben wurde; die Definition hat `WARNING` gemeldet oder die Prüfung gegen das EDIFACT-Verzeichnis hat etwas gefunden.
- **12 – Archivfehler.** Die Ausgabe ist geschrieben, aber die Eingabedatei ließ sich nicht ins Archiv verschieben; sie liegt noch am alten Ort.
- **13 – Lizenzfehler.** Keine gültige Lizenz für diesen Rechner, oder sie umfasst ein Modul nicht, das der Lauf braucht (siehe „Module"); es wurde nichts gewandelt.
- **14 – läuft bereits.** Dieselbe Konfiguration oder dieselbe Eingabedatei war länger als zehn Minuten belegt.

1.13 Log, Laufprotokoll und Verlauf

Je Lauf eine `.LOG`-Datei im Logordner der Konfiguration (ohne Angabe `temp\` im Installationsordner). Darin stehen die Angaben des Laufs, der Rückgabewert, die Meldung, alle `WARNING`- und `ERROR`-Meldungen und eine Statistik, wie oft jede `IF`-Bedingung wahr und falsch war – hilfreich, wenn ein Feld leer bleibt und Sie wissen wollen, welcher Zweig gelaufen ist. Enthält die Ausgabedatei `{ZAEHLER}`, heißt das Log wie die Ausgabe und wird bei derselben Nummer überschrieben, sonst trägt es einen Zeitstempel. Auch ein abgebrochener Lauf hinterlässt ein Log. Geschrieben wird es in der eingestellten Sprache.

Das Laufprotokoll in der Datenbank hält jeden Lauf fest, auch einen gescheiterten. Sie lesen es im Editor unter „Verlauf"; der letzte Lauf steht außerdem bei jeder Konfiguration, als farbiger Punkt und mit Uhrzeit. Wie lange es aufbewahrt wird, steht unter Einstellungen → Globale Konfiguration.

1.14 Archiv, Wiederherstellen und Löschschutz

Ist das Archivieren eingeschaltet, wird die Eingabedatei **nach einem erfolgreichen Lauf** ins Archiv verschoben und bekommt Datum und Uhrzeit in den Namen. Nach einem Fehlschlag bleibt sie liegen – der Lauf lässt sich also wiederholen, sobald die Ursache behoben ist.

Im Editor öffnet der Knopf **Archiv** auf der Konfiguration die Liste:

- Sie sehen zu jeder Datei den Lauf, der sie archiviert hat, und eine Vorschau.
- **Wiederherstellen** kopiert die Datei an den Ort der Eingabedatei zurück, sodass sie beim nächsten Lauf erneut verarbeitet wird. Eine dort vorhandene Datei wird nur nach Rückfrage überschrieben. **Das Archiv selbst bleibt unverändert** – es ist der Nachweis, was angeliefert wurde. Ist die Eingabedatei ein Muster, kommt die Datei unter ihrem ursprünglichen Namen zurück: aus `INVOIC_7_20260920_101010.csv` wieder `INVOIC_7.csv`.
- **Als Testdatei verwenden** öffnet den Konverter mit dieser Datei.

Löschschutz: Das Programm löscht von sich aus nur innerhalb seines Installationsordners. Liegt Ihr Archiv woanders – und das ist der Normalfall –, werden alte Archivdateien nur gelöscht, wenn Sie das in der Konfiguration ausdrücklich freigeben. Jede Freigabe, jeder Entzug und jede Löschung außerhalb wird mit Zeitpunkt und Windows-Benutzer in der Datenbank festgehalten; die letzte Freigabe steht im Formular. Logdateien außerhalb des Installationsordners werden nie gelöscht.

1.15 Gleichzeitige Läufe

Jeder Aufruf ist ein eigener Prozess; verschiedene Konfigurationen laufen nebeneinander. Treffen zwei Läufe **derselben** Konfiguration oder derselben Eingabedatei aufeinander, wartet der zweite – über Sperrdateien in `temp\Sperren`, höchstens zehn Minuten. Danach endet er mit Rückgabewert 14, ohne etwas zu wandeln. Das gilt auch zwischen Aufgabenplanung und Editor.

1.16 Datensicherung

Sichern Sie regelmäßig:

- `data\` – die Datenbank mit Partnern, Konfigurationen, Definitionen und dem Laufprotokoll. Das ist der wichtigste Ordner.
- `Lizenz\` – Ihre Lizenzdatei.
- Ihre eigenen Ein-, Ausgabe- und Archivordner, soweit sie nicht ohnehin gesichert werden.

Die Programmordner müssen Sie nicht sichern; sie kommen aus der Auslieferung.

Zurück zu einer Sicherung: Beenden Sie Editor und Konverter, benennen Sie die gewünschte Datei `LeNic.EdiKonverter.vor-Schema…db` in `LeNic.EdiKonverter.db` um (die bisherige vorher beiseitelegen) und starten Sie den Editor. Stammt die Sicherung von einer älteren Fassung, wird sie beim Öffnen wieder auf den aktuellen Stand gebracht.

1.17 Abweichende Pfade

Sollen Datenbank oder Lizenz woanders liegen – etwa auf einem Netzlaufwerk –, legen Sie im Installationsordner die Datei `LeNic.EdiKonverter.json` an:

```
{ "Datenbank": "data\\LeNic.EdiKonverter.db", "Lizenzordner": "Lizenz" }
```

Ein relativer Pfad gilt ab dem Installationsordner, ein vollständiger Pfad ersetzt ihn. In den Pfaden einer Konfiguration steht der Installationsordner als `{BASIS}`; so überstehen sie auch einen Umzug der Installation. Den Installationsordner selbst können Sie mit der Umgebungsvariablen `LENIC_EDIKONVERTER_HOME` vorgeben; sie hat Vorrang vor der Ordnerkonvention.

1.18 E-Mail über Microsoft 365

Microsoft 365 (Exchange Online) nimmt über SMTP keine Anmeldung mit Benutzer und Kennwort mehr an. Der Konverter sendet dort über **Microsoft Graph**: Er meldet sich ohne Benutzer als App an und sendet im Namen eines Postfachs. Verbunden wird über HTTPS (Port 443) mit `login.microsoftonline.com` und `graph.microsoft.com`. Einrichten muss das ein Administrator von Microsoft 365.

1. App registrieren – im Microsoft Entra Admin Center unter *Identität → Anwendungen → App-Registrierungen → Neue Registrierung*:

- Name, etwa „LeNic EDI-Konverter“; *Nur Konten in diesem Organisationsverzeichnis*; keine Umleitungs-URI.
- Auf der Übersichtsseite **Anwendungs-ID (Client-ID)** und **Verzeichnis-ID (Tenant-ID)** notieren.

2. Geheimen Clientschlüssel anlegen – *Zertifikate & Geheimnisse → Neuer geheimer Clientschlüssel*. Kopieren Sie sofort den **Wert**, nicht die Geheimnis-ID; später wird er nicht mehr angezeigt. Notieren Sie das Ablaufdatum: Danach schlägt der Versand fehl, bis ein neuer Schlüssel eingetragen ist.

3. Berechtigung erteilen – eine von zwei Möglichkeiten:

- *Einfach: API-Berechtigungen → Berechtigung hinzufügen → Microsoft Graph → Anwendungsberechtigungen →* `Mail.Send`, dann **Administratorzustimmung erteilen**. Die App darf damit im Namen **jedes** Postfachs senden.
- *Auf ein Postfach beschränkt (empfohlen):* `Mail.Send` **nicht** in Entra erteilen, sondern in Exchange Online PowerShell nur für das Absenderpostfach. Die Objekt-ID steht in Entra unter *Unternehmensanwendungen* bei der App:

  ```
  Connect-ExchangeOnline
  New-ServicePrincipal -AppId <Anwendungs-ID> -ObjectId <Objekt-ID> -DisplayName "LeNic EDI-Konverter"
  New-ManagementScope -Name "LeNic Absender" -RecipientRestrictionFilter "PrimarySmtpAddress -eq 'edi-konverter@firma.de'"
  New-ManagementRoleAssignment -App <Anwendungs-ID> -Role "Application Mail.Send" -CustomResourceScope "LeNic Absender"
  ```

  Ist `Mail.Send` zusätzlich in Entra erteilt, gilt wieder jedes Postfach.

4. Im Editor eintragen – *Einstellungen → E-Mail-Versand → Microsoft 365 (Graph)*: Tenant-ID (oder `firma.onmicrosoft.com`), Anwendungs-ID, Clientschlüssel und als **Absenderadresse** das Postfach, aus dem gesendet wird – ein Benutzer- oder freigegebenes Postfach in Exchange Online. Speichern, dann „Test-E-Mail senden". Bis eine neue Berechtigung greift, können einige Minuten vergehen.

Häufige Meldungen:

Meldung	Ursache
`AADSTS7000215: Invalid client secret`	Statt des Werts die Geheimnis-ID eingetragen, oder ein Tippfehler
`AADSTS7000222`	Der Clientschlüssel ist abgelaufen – neuen anlegen und eintragen
`AADSTS700016`	Die Anwendungs-ID gibt es in diesem Tenant nicht
`AADSTS90002`	Den Tenant gibt es nicht – Tenant-ID prüfen
`403 ErrorAccessDenied`	`Mail.Send` fehlt, die Zustimmung fehlt, oder der Absender liegt außerhalb des Bereichs
`404 ErrorInvalidUser`, `MailboxNotEnabledForRESTAPI`	Die Absenderadresse ist kein Postfach in Exchange Online

---

1.19 Fehlerbehebung und häufige Fragen

Der Editor startet nicht und meldet, eine Komponente fehle. Die .NET-10-Laufzeit („Windows Desktop Runtime") fehlt. Nachinstallieren und den Editor neu starten.

Der Editor zeigt beim Start das Lizenzfenster. Die Lizenz fehlt, ist abgelaufen oder für einen anderen Rechner ausgestellt. Das Fenster nennt den Grund und den Rechnernamen für die Anfrage.

Der Lauf endet mit 5, obwohl die Datei da ist. Prüfen Sie den Pfad in der Konfiguration auf Tippfehler und ob das Konto der Aufgabenplanung das Laufwerk erreicht – ein verbundenes Netzlaufwerk (`X:\`) kennt ein Dienstkonto oft nicht; nehmen Sie den vollständigen Netzpfad (`\\Server\Freigabe\…`).

Der Lauf endet mit 6. Die Ausgabedatei gibt es schon. Ergänzen Sie das Muster um `{ZAEHLER}` oder `{DATUM}`, dann entsteht jedes Mal ein neuer Name.

Ein Feld in der Ausgabe bleibt leer. Sehen Sie im Log die `IF`-Statistik an: War die Bedingung nie wahr, trifft das Muster nicht. Im Editor mit „Beispieldatei" das Muster an der Datei prüfen – trifft es nicht, bleibt das Feld unverändert.

Der Lauf endet mit 7: „Segment 1 ist kein EDIFACT-Segment". Die Datei ist anders getrennt, als die Definition annimmt – etwa ein falsches Segmentende im `UNA`, oder es ist gar keine EDIFACT-Datei. Die Meldung zeigt den Anfang des

Segments; Steuerzeichen erscheinen darin als `<CR>` und `<LF>`. Zeilenumbrüche allein sind kein Problem: Nach jedem Segment, vor dem Segmentende oder mitten im Segment (eine nach 80 Zeichen umbrochene Datei) werden sie überlesen.

Der Lauf endet mit 7: „FORMAT-DATE: … passt nicht zum Format …". Ein Datum in der Eingabe hat nicht das erwartete Format, etwa `7.8.2026` statt `07.08.2026`, oder ist kein gültiges Datum wie `31.02.2026`. Punkte oder keine Punkte spielen keine Rolle – gezählt werden die Ziffern. Die Meldung nennt den Wert; korrigieren Sie ihn in der Warenwirtschaft oder das Format in der Definition.

Im Lieferavis steht eine Palette zu viel oder zu wenig. Jeder Satz mit SSCC beginnt eine neue Palette; die Positionen dahinter gehören zu ihr. Prüfen Sie die Reihenfolge der Sätze in Ihrer Datei.

In der Rechnung fehlt eine Steuerzeile. Steuerzeilen werden je Rechnung gesammelt und hinter den Summen geschrieben – aber nur, wenn die Satzart der Steuerzeile vor dem Summensatz dieser Rechnung steht. Eine Steuerzeile hinter der Summe gehört schon zur nächsten.

Umlaute kommen falsch an. Stellen Sie den Zeichensatz der Ein- und Ausgabe in der Konfiguration ausdrücklich ein, statt es der Erkennung zu überlassen.

Der Lauf endet mit 14. Ein anderer Lauf derselben Konfiguration ist noch beschäftigt oder wurde hart beendet. Läuft nichts mehr, können die Dateien in `temp\Sperren` entfernt werden.

Kann dieselbe Definition für mehrere Partner gelten? Ja. Die Partnerdaten kommen aus der Konfiguration und stehen in der Definition als `~~PARTNER-GLN`, `~~PARTNER-NAME` und `~~ABSENDER-GLN`. Legen Sie je Partner eine Konfiguration an, die denselben Konverter verwendet.

Woher kommt die Nummer in der Ausgabedatei? Der Zähler wird je Konfiguration in der Datenbank fortgeschrieben und erst nach einer gelungenen Wandlung vergeben. Im Formular lässt er sich mit „Setzen" ausdrücklich neu setzen.

Editor oder Konverter sprechen die falsche Sprache. Prüfen Sie **Einstellungen → Sprache**. „Wie Windows" nimmt Deutsch nur bei deutscher Anzeigesprache von Windows. Der Editor übernimmt eine Änderung nach einem Neustart, der Konverter beim nächsten Lauf.

1.20 Deinstallation

- Beenden Sie Editor und Konverter sowie geplante Aufgaben, die den Konverter aufrufen.
- Entfernen Sie „LeNic EDI-Konverter" unter **Einstellungen → Apps → Installierte Apps** (Windows 10: „Apps und Features").
- **Ihre Daten bleiben stehen:** `data\`, `Lizenz\`, `temp\` und die Ein- und Ausgabeordner samt Inhalt. Werden sie nicht mehr gebraucht, löschen Sie den Installationsordner danach von Hand – vorher `data\` und `Lizenz\` sichern, wenn Sie die Daten behalten wollen.
- In der Registrierung bleibt nichts zurück; persönliche Einstellungen des Editors (nur das Farbschema) stehen unter `%APPDATA%\LeNic.EdiKonverter`.

1.21 Kontakt

Ljuban Magdelinic, IT Consulting — Lizenzgeber und Hersteller Franz Lehar-Gasse 43, A-2384 Breitenfurt bei Wien, Österreich **ljuban@magdelinic.consulting**

Die Lizenzbedingungen stehen in `LICENSE.md`, die verwendeten Fremdkomponenten in `THIRD-PARTY-NOTICES.md`; beide liegen neben dem Programm und im Editor unter „Hilfe und Lizenz". Die englischen Fassungen `LICENSE.en.md` und `THIRD-PARTY-NOTICES.en.md` sind Übersetzungen; maßgeblich ist die deutsche Fassung.

TEIL 2

Das Mapping

Wie aus Spaltennamen ein Konverter wird

Dieses Kapitel erklärt, was zwischen Ihrer Datei und der Nachricht Ihres Partners geschieht – und wie der Editor einen Konverter dafür schreibt, ohne dass Sie eine Zeile der Definitionssprache tippen müssen.

Es ist der Mittelteil des Handbuchs: Das vorige Kapitel zeigt die Oberfläche, das folgende beschreibt die Sprache im Einzelnen. Hier steht dazwischen, was beide verbindet.

2.1 Was ein Mapping ist

Ein Mapping ist die Antwort auf eine einzige Frage: **Welcher Wert aus der Eingabe gehört an welche Stelle der Ausgabe?**

Beide Seiten sind verschieden gebaut. Ihre Warenwirtschaft denkt in Sätzen und Spalten – ein Kopfsatz, darunter Positionen, jede Spalte an ihrer Stelle. Eine EANCOM-Nachricht denkt in Segmenten und Qualifiern – `BGM` trägt die Belegnummer, `NAD+BY` den Besteller, `QTY+21` die bestellte Menge. Beides ist festgelegt, nur eben verschieden.

Der Konverter liest die Eingabe Satz für Satz und schreibt die Ausgabe mit `WRITE`. Was dazwischen geschieht, steht in der Definition:

```
: IF S-SEG = "BGM"
:    K-BESTELLNUMMER                  = BGM+*+VALUE
: END-IF
```

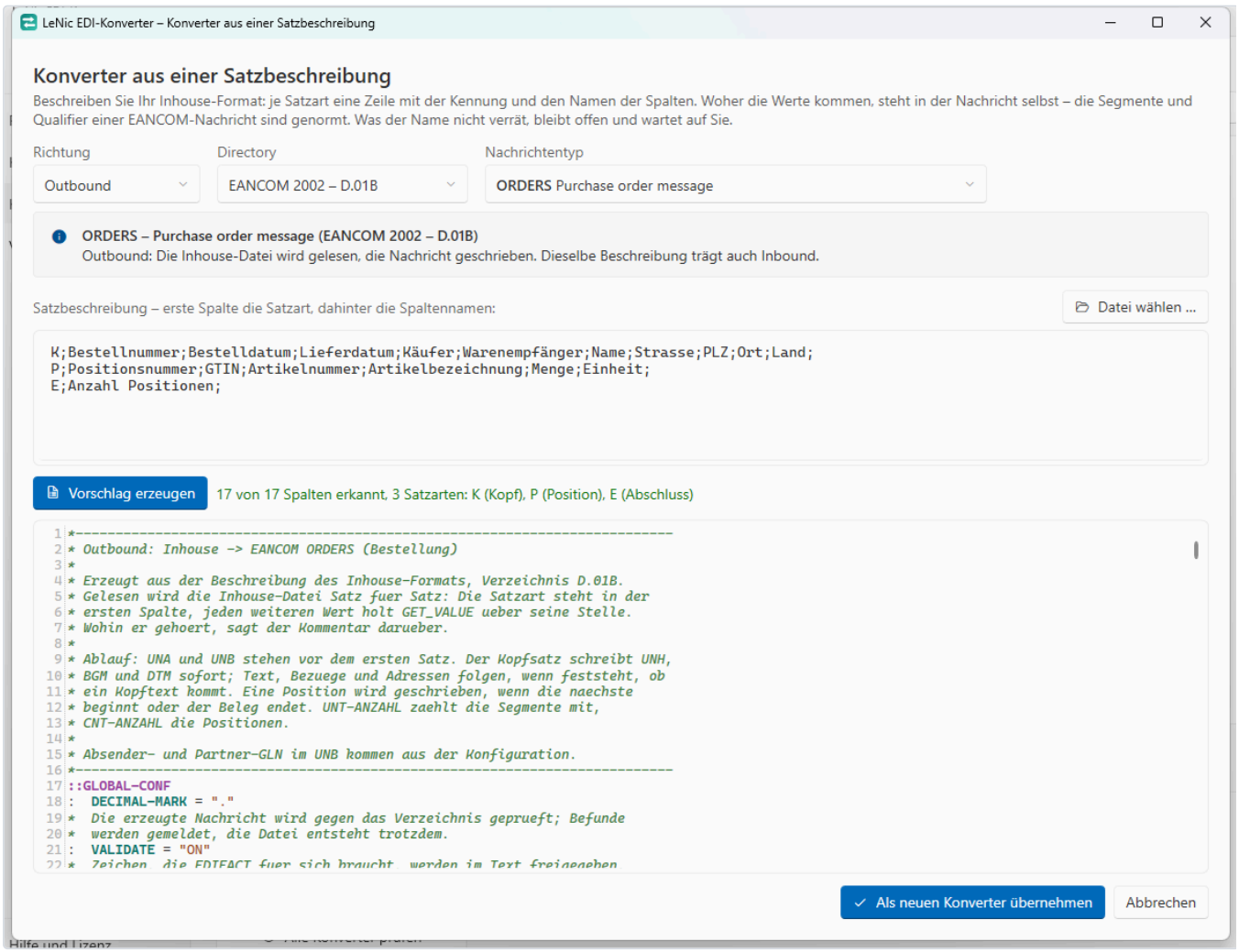
Gelesen: *Ist der gerade gelesene Satz ein `BGM`-Segment, dann steht die Bestellnummer in seiner zweiten Gruppe.* Dieselbe Zeile in der Gegenrichtung heißt:

```
:    BGM-NR                           = GET_VALUE (2)
```

Die Bestellnummer steht in der zweiten Spalte des Inhouse-Satzes.

Mehr ist ein Mapping nicht. Es sind viele solcher Zeilen, in der richtigen Reihenfolge, mit den Fällen dazwischen, die es im Alltag gibt: ein Datum mit Uhrzeit, ein Text in mehreren Sprachen, eine Position ohne Mengeneinheit.

2.2 Die Satzbeschreibung



> Aus der Satzbeschreibung entsteht der Entwurf – hier Outbound, EANCOM 2002 – D.01B, ORDERS. Die Zeile neben dem Knopf sagt, wie viele Spalten erkannt wurden.

Damit der Editor diese Zeilen schreiben kann, braucht er eine einzige Angabe: **wie Ihre Datei aussieht.** Das ist die Satzbeschreibung – je Satzart eine Zeile, zuerst die Kennung, dahinter die Namen der Spalten:

```
K;Bestellnummer;Bestelldatum;Lieferdatum;Käufer;Warenempfänger;Lieferant;
KTXT;Auftragstext;
P;Positionsnummer;GTIN;Artikelnummer;Menge;Einheit;Artikelbezeichnung;
PTXT;Positionstext;
E;Anzahl Positionen;
```

Das ist **keine Beispieldatei mit Werten**, sondern eine Beschreibung. Was in den Spalten steht, muss niemand vormachen: Die Bedeutung steht im Namen, und wo dieser Wert in der Nachricht steht, ist genormt.

Die Trennzeichen erkennt der Editor selbst (Strichpunkt, Tabulator, senkrechter Strich oder Komma); eine leere Spalte hinter dem letzten Trenner zählt nicht. Zeilen, die mit `*` beginnen, sind Kommentar.

Eine Satzart darf mehrmals dastehen, um zu zeigen, wie die Sätze ineinandergreifen – im Lieferavis etwa Palette, Positionen, Palette, Positionen. Gezählt wird sie einmal. Steht sie ein zweites Mal mit **anderen** Spalten da, ist das ein Widerspruch, und der Editor sagt es.

Für Bestellung, Lieferavis und Rechnung bringt der Editor eine Vorlage mit. Wählen Sie im Fenster den Nachrichtentyp, steht seine Vorlage im Feld – solange Sie dort noch nichts Eigenes geschrieben haben. Die Vorlagen liegen auch als Dateien bei (`ORDER_VORLAGE.csv` , `DESADV.txt` , `INVOIC.txt`).

Welche Rolle ein Satz hat

Aus der Beschreibung ergibt sich zuerst, **was jeder Satz ist**:

Rolle	Woran sie erkannt wird
Position	Der Satz führt, was eine Position ausmacht – Positionsnummer, GTIN, Artikelnummer, Menge oder Artikelbezeichnung. Preis, Betrag und Steuersatz allein machen noch keine Position; sie stehen auch im Kopf und in den Summen
Kopf	Ein Satz ohne solche Spalten, der keine der übrigen Rollen hat
Text zum Kopf, **Text zur Position**	Die Kennung setzt auf der eines anderen Satzes auf: `KTXT` gehört zu `K`, `PTXT` zu `P`
Belegrabatt, **Positionsrabatt**	Der Satz führt einen Rabatt (`Rabatt in %`, `Wert`). Er gehört zur Position, wenn sein Träger eine ist – `PR` zu `P` – oder, ohne passende Kennung, der Satz davor; sonst zum ganzen Beleg (`KR` zu `K`)
Belegzuschlag, **Positionszuschlag**	Wie ein Rabattsatz, aber mit einem Zuschlag (`Zuschlag in %`, `Zuschlagsbetrag`): `KZ` zu `K`, `PZ` zu `P`
Steuerzeile	Steuersatz und steuerpflichtiger Betrag, aber kein Rechnungsbetrag – je Steuersatz eine Zeile
Sendung	Im Lieferavis: Zahl der Packstücke oder Gesamtgewicht der Sendung
Packstück	Im Lieferavis: der Satz führt eine SSCC/NVE – etwa eine Palette. Die Positionen dahinter gehören zu ihm, bis das nächste beginnt
Abschluss	Der Satz führt die Zahl der Positionen oder eine Summe wie den Rechnungsbetrag, aber keine Position – oder heißt schlicht `E` und steht als letzter

In einem Textsatz gilt die Rolle vor dem Namen: Eine Spalte `Artikeltext` in `PTXT` ist der Positionstext, auch wenn sie nach einer Artikelbezeichnung klingt.

Welche Bedeutung eine Spalte hat

Danach bekommt jede Spalte ihre **Bedeutung**. Verglichen wird nachsichtig: Der Name wird auf Kleinbuchstaben gebracht, Umlaute werden aufgelöst (`ä` → `ae`), und alles, was kein Buchstabe und keine Ziffer ist, entfällt. Aus „GLN Besteller" wird so `glnbesteller`. Erst wird auf Gleichheit geprüft, dann auf Enthaltensein, längere Schlüsselwörter zuerst – damit `Positionstext` nicht an `Position` hängen bleibt.

Diese Bedeutungen kennt der Editor, mit ihrer Stelle in jedem Nachrichtentyp. Ein Strich heißt: Dort ist sie nicht vorgesehen – die Spalte entsteht, bleibt aber offen, wie eine nicht erkannte.

Bedeutung	Erkannt an (Auswahl)	ORDERS	INVOIC	DESADV	ORDRSP
Belegnummer	`belegnummer`, `belegnr`, `rechnungsnummer`	`BGM+220`	`BGM+380`	`BGM+351`	`BGM+231`
Bestellnummer	`bestellnummer`, `bestellnr`, `auftragsnummer`	`BGM+220`	`RFF+ON`	`RFF+ON`	`RFF+ON`
Belegdatum	`belegdatum`, `rechnungsdatum`, `lieferscheindatum`	`DTM+137`	`DTM+137`	`DTM+137`	`DTM+137`
Bestelldatum	`bestelldatum`, `auftragsdatum`, `datumbestellung`	`DTM+137`	`DTM+171`	`DTM+171`	`DTM+171`
Lieferdatum	`lieferdatum`, `liefertermin`, `lieferzeitpunkt`	`DTM+2`	`DTM+35`	`DTM+17`	`DTM+2`
Versanddatum	`versanddatum`, `versandtag`, `abgangsdatum`	–	–	`DTM+11`	–
GLN Käufer	`kaeufer`, `besteller`, `kunde`	`NAD+BY`	`NAD+BY`	`NAD+BY`	`NAD+BY`
GLN Lieferanschrift	`warenempfaenger`, `lieferanschrift`, `lieferadresse`	`NAD+DP`	`NAD+DP`	`NAD+DP`	`NAD+DP`
GLN Lieferant	`lieferant`, `absender`, `glnlieferant`	`NAD+SU`	`NAD+SU`	`NAD+SU`	`NAD+SU`
GLN Rechnungsempfänger	`rechnungsempfaenger`, `glnrechnungsempfaenger`, `rechnungsadresse`	`NAD+IV`	`NAD+IV`	`NAD+IV`	`NAD+IV`
Name Lieferanschrift	`name`, `name1`, `firmenname`	`NAD+DP`	`NAD+DP`	`NAD+DP`	`NAD+DP`
Straße Lieferanschrift	`strasse`, `str`, `strassehausnummer`	`NAD+DP`	`NAD+DP`	`NAD+DP`	`NAD+DP`
Ort Lieferanschrift	`ort`, `stadt`, `wohnort`	`NAD+DP`	`NAD+DP`	`NAD+DP`	`NAD+DP`
PLZ Lieferanschrift	`plz`, `postleitzahl`	`NAD+DP`	`NAD+DP`	`NAD+DP`	`NAD+DP`
Land Lieferanschrift	`land`, `laendercode`, `landescode`	`NAD+DP`	`NAD+DP`	`NAD+DP`	`NAD+DP`
Währung	`waehrung`, `waehrungscode`, `currency`	`CUX+2`	`CUX+2`	–	`CUX+2`
Kopftext	`auftragstext`, `kopftext`, `bestelltext`	`FTX+ACI`	`FTX+ACI`	`FTX+ACI`	`FTX+ACI`
Positionsnummer	`positionsnummer`, `positionsnr`, `posnummer`	`LIN`	`LIN`	`LIN`	`LIN`
Gtin	`gtin`, `ean`, `artikelean`	`LIN`	`LIN`	`LIN`	`LIN`
Artikelnummer	`artikelnummer`, `artikelnr`, `artnr`	`PIA+1`	`PIA+1`	`PIA+1`	`PIA+1`
Menge	`menge`, `bestellmenge`, `anzahlartikel`	`QTY+21`	`QTY+47`	`QTY+12`	`QTY+113`
Einheit	`einheit`, `mengeneinheit`, `me`	`QTY+21`	`QTY+47`	`QTY+12`	`QTY+113`
Artikelbezeichnung	`artikelbezeichnung`, `artikeltext`, `bezeichnung`	`IMD`	`IMD`	`IMD`	`IMD`
Positionstext	`positionstext`, `postext`, `zusatztext`	`FTX+LIN`	`FTX+LIN`	`FTX+LIN`	`FTX+LIN`
Preis	`preis`, `nettopreis`, `einzelpreis`	`PRI+AAA`	`PRI+AAA`	–	`PRI+AAA`
Positionsbetrag	`positionsbetrag`, `positionswert`, `zeilenbetrag`	`MOA+203`	`MOA+203`	`MOA+203`	`MOA+203`
Steuersatz	`steuersatz`, `mwstsatz`, `mwst`	`TAX+7`	`TAX+7`	–	`TAX+7`
Anzahl Positionen	`anzahlposten`, `anzahltposten`, `anzahlpositionen`	`CNT+2`	`CNT+2`	`CNT+2`	`CNT+2`
Warenwert	`warenwert`, `nettobetrag`, `summenetto`	`MOA+79`	`MOA+79`	–	`MOA+79`
Steuerbetrag	`steuerbetrag`, `mwstbetrag`, `ustbetrag`	–	`MOA+124`	–	–
Rechnungsbetrag	`rechnungsbetrag`, `gesamtbetrag`, `bruttobetrag`	–	`MOA+77`	–	–
Anzahl Packstücke	`anzahlpaletten`, `anzahlpackstuecke`, `anzahlcolli`	–	–	`PAC`	–
Bruttogewicht Sendung	`sendungsgewichtbrutto`, `bruttogewichtsendung`, `sendungbrutto`	–	–	`MEA+AAB`	–
Nettogewicht Sendung	`sendungsgewichtnetto`, `nettogewichtsendung`, `sendungnetto`	–	–	`MEA+AAA`	–
Nummer des Packstücks	`lfdnrderpalette`, `lfdnrpalette`, `palettennummer`	–	–	–	–
SSCC/NVE	`sscc`, `nve`, `nummerderversandeinheit`	–	–	`GIN+BJ`	–
Bruttogewicht	`bruttogewicht`, `brutto`, `gewichtbrutto`	–	–	`MEA+AAB`	–

Nettogewicht	`nettogewicht`, `netto`, `gewichtnetto`	–	–	`MEA+AAA`	–
Losnummer	`losnummer`, `losnr`, `charge`	–	–	`GIN+BX`	–
Mindesthaltbarkeitsdatum	`mindesthaltbarkeitsdatum`, `mindesthaltbarkeit`, `haltbarkeitsdatum`	–	–	`DTM+361`	–
Lieferscheinnummer	`lieferscheinnummer`, `lieferscheinnr`, `lsnummer`	`RFF+DQ`	`RFF+DQ`	`BGM+351`	`RFF+DQ`
UStIdLieferant	`ustid`, `ustidnr`, `ustidnummer`	`RFF+VA`	`RFF+VA`	`RFF+VA`	`RFF+VA`
Steuerbasis	`basissteuerwert`, `steuerbasis`, `steuerpflichtigerbetrag`	–	`MOA+125`	–	–
RabattProzent	`rabatt`, `rabattin`, `rabattinprozent`	`PCD+1`	`PCD+1`	–	`PCD+1`
RabattBetrag	`rabattbetrag`, `rabattwert`, `nachlass`	`MOA+204`	`MOA+204`	–	`MOA+204`
ZuschlagProzent	`zuschlag`, `zuschlagin`, `zuschlaginprozent`	`PCD+2`	`PCD+2`	–	`PCD+2`
ZuschlagBetrag	`zuschlagbetrag`, `zuschlagsbetrag`, `zuschlagwert`	`MOA+23`	`MOA+23`	–	`MOA+23`
Lieferbedingung	`lieferbedingung`, `lieferbedingungen`, `lieferkondition`	`TOD+6`	`TOD+6`	`TOD+6`	`TOD+6`

Wenn der Zusammenhang entscheidet

Manche Namen sagen allein nicht genug. Dann entscheidet, **wo** die Spalte steht:

- **Anschrift und USt-IdNr. gehören zur Partei davor.** Im Kopf gehören `Name`, `Strasse`, `PLZ`, `Ort`, `Land` und eine USt-IdNr. – auch eine Spalte, die schlicht `MwST` heißt – zu der Partei, deren GLN zuletzt davorsteht. Hinter `Lieferant` ist es also die Anschrift des Lieferanten (`NAD+SU`), hinter `Warenempfänger` die Lieferanschrift (`NAD+DP`). Die USt-IdNr. steht in der Nachricht als `RFF+VA` gleich hinter ihrem `NAD`.
- **Ein Steuersatz im Kopf** gilt für jede Position, die keinen eigenen führt.
- **`Rabatt in %` und `Wert`** sind im Belegrabatt ein Rabatt auf den ganzen Beleg, im Positionsrabatt einer auf die Position davor. In einem Zuschlagssatz ist `Wert` der Zuschlagsbetrag.
- **In einer Steuerzeile** sind Satz, Basis und Steuerbetrag die dieses Steuersatzes; im Abschluss dieselben Namen die Summen des Belegs.
- **`Netto Wert`** ist in der Position ihr Betrag (`MOA+203`), im Abschluss die Summe der Positionen (`MOA+79`).
- **Die Lieferscheinnummer** ist im Lieferavis der Beleg selbst (`BGM`), in der Rechnung ein Bezug (`RFF+DQ`). Steht sie im Positionssatz, gehört sie zur Position: In der Rechnung trägt dann jede Position ihren Lieferschein (`RFF+DQ` hinter dem Preis), ein `Lieferdatum` im Positionssatz ist das Datum dieses Lieferscheins (`DTM+171`). So entsteht eine Sammelrechnung.

Was kein bekannter Name ist, wird nicht geraten. Die Spalte entsteht trotzdem, ihre Zuweisung bleibt offen, und die Meldung unter dem Knopf nennt sie. In der Definition steht an ihrer Stelle ein Hinweis:

```
*    Spalte 8 "Sachbearbeiter": nicht erkannt – bitte die Zuweisung von Hand ergaenzen
```

Das ist Absicht. Ein geratener Wert an der falschen Stelle fällt erst beim Partner auf; eine offene Zeile fällt sofort auf.

2.3 Beide Richtungen aus einer Beschreibung

Dieselbe Beschreibung trägt beide Richtungen. Was „Lieferdatum" bedeutet und wozu `PTXT` gehört, ändert sich nicht dadurch, ob die Datei gelesen oder geschrieben wird – nur was daraus entsteht.

Die Richtung wählen Sie oben im Fenster im Feld **Richtung** (Inbound oder Outbound), daneben das **Directory** und den **Nachrichtentyp**: Bestellung (ORDERS), Rechnung (INVOIC), Lieferavis (DESADV) oder Bestellbestätigung (ORDRSP). Die Typen unterscheiden sich weniger, als man denkt – Kopf mit `BGM`, `DTM` und Adressen, Positionen mit `LIN`, `PIA`, `IMD` und `QTY`, am Ende `CNT`. Was einer darüber hinaus trägt, etwa Preise und Summen in der Rechnung oder die Versandstufe `CPS` im Lieferavis, kennt der Editor. Die **Reihenfolge der Segmente** nimmt er aus dem gewählten Verzeichnis, denn sie unterscheidet sich zwischen den Typen und den Ausgaben.

Die Beispiele in diesem Kapitel zeigen die Bestellung.

Inbound: Nachricht → Inhouse

Gelesen wird die Nachricht, geschrieben Ihre Datei. Die Reihenfolge gibt die Nachricht vor, und sie ist der Grund für die einzige Feinheit dieser Richtung:

- Die Kopfsegmente – `BGM`, `DTM`, `NAD`, `FTX` – füllen die Felder des Kopfsatzes.
- Geschrieben wird der Kopfsatz aber erst beim **ersten** `LIN`. Vorher ist nicht sicher, dass alle Kopfsegmente gelesen sind.
- Eine Position wird gesammelt und beim **nächsten** `LIN` **oder bei** `UNS` geschrieben – `QTY` und `IMD` folgen erst hinter `LIN`.
- Bei `UNT` kommen die letzte Position, der Endsatz und der Abgleich mit `CNT`: Meldet die Nachricht eine andere Zahl von Positionen, als gelesen wurden, steht eine Warnung im Protokoll.

Der Stand der Verarbeitung steht in einem Satz `TEMP`, der nie geschrieben wird: der gerade gelesene Qualifier, die Sprache, die Zähler und zwei Merker.

Zwei Einzelheiten dieser Richtung:

- **Die Artikelbezeichnung** wird aus dem `IMD` in deutscher Sprache gelesen – oder aus einem `IMD` ganz ohne Sprache (`IMD+F++:::Text`), wie ihn viele Partner schicken.
- **Bestell-, Liefer- und Mindesthaltbarkeitsdatum** schreibt der Entwurf als `TT.MM.JJJJ` in Ihre Datei, also `04.08.2026` statt `20260804`.

Outbound: Inhouse → Nachricht

Gelesen wird Ihre Datei, geschrieben die Nachricht. Die Satzart steht in der ersten Spalte, jeden weiteren Wert holt `GET_VALUE` über seine Stelle. In der Gegenrichtung schreibt der Entwurf die Satzart über die Vorlage `~~SATZART` an den Anfang jedes Satzes.

- `UNA` und `UNB` stehen vor dem ersten Satz und entstehen in START-CONF. **Absender- und Partner-GLN kommen aus der Konfiguration**, nicht aus der Definition – so dient ein Konverter mehreren Partnern.
- Der Kopfsatz schreibt `UNH`, `BGM` und die `DTM`-Segmente sofort. `FTX` und die `NAD`-Gruppe folgen erst, wenn feststeht, ob ein Kopftext kommt: In der Nachricht steht der freie Text **vor** der `NAD`-Gruppe, in Ihrer Datei aber hinter dem Kopfsatz.
- Eine Position wird gesammelt und geschrieben, wenn die nächste beginnt oder der Beleg endet – aus demselben Grund: Der Positionstext steht hinter dem Positionssatz, in der Nachricht aber hinter der Menge.
- Der Endsatz schließt mit `UNS` (wo die Nachricht eines kennt), `CNT`, den Summen und `UNT` ab, `UNZ` steht in END-CONF.
- **Steuerzeilen** stehen in Ihrer Datei vor der Summe, in der Nachricht aber dahinter. Sie werden deshalb in einer temporären Tabelle gesammelt – jede mit der Nummer ihrer Nachricht, sodass bei zwei Rechnungen in einer Datei jede nur ihre eigenen bekommt – und im Abschluss geschrieben.
- **Datum:** Bestell-, Liefer- und Mindesthaltbarkeitsdatum liest der Entwurf als `TTMMJJJJ` und schreibt sie im Format 102 (`CCYYMMDD`) mit `FORMAT-DATE`. Ein Datum mit Punkten (`06.10.2026`) wird ebenso gelesen. Führt Ihre Datei kein Belegdatum, wird `DTM+137` mit dem Tag der Wandlung geschrieben – EANCOM verlangt es.
- **Die Segmente folgen ohne Zeilenumbruch** aufeinander, so wie Partner sie üblicherweise erwarten.

Gezählt wird mitgeschrieben. `UNT-ANZAHL` beginnt bei 1 – das `UNT` zählt sich selbst mit – und wird nach jedem geschriebenen Segment erhöht. Nach dem `WRITE` steht es wieder auf 1, weil ein Satz nach dem Schreiben seinen Anfangszustand annimmt. Ebenso zählen `CNT-ANZAHL` die Positionen und `UNH-NR` die Nachrichten.

Drei Dinge nimmt diese Richtung Ihnen ab, die von Hand leicht vergessen werden:

- **Freie Texte werden freigegeben.** Ein Apostroph in einer Artikelbezeichnung beendete sonst das Segment; `TRANSLATE INHOUSE-ZEICHEN TO EDIFACT-ZEICHEN` macht daraus `?'`.
- **Die erzeugte Nachricht wird geprüft.** `VALIDATE = "ON"` hält sie gegen das UN/EDIFACT-Verzeichnis – Segmentaufbau, Formate, Codelisten, dazu der Umschlag und die Zähler in `UNT` und `UNZ` – und meldet, was nicht passt, bevor die Datei den Partner erreicht.

- **Eine Datei ohne Endsatz ist ein Fehler.** Eine Nachricht ohne `UNT` soll nicht hinausgehen; der Lauf meldet es und schreibt keine Ausgabedatei.

Ein Zahlenfeld, das nur bei Bedarf geschrieben wird – ein Rabatt, ein Gewicht, ein Steuersatz –, beginnt **leer** statt mit 0. So entsteht ein Segment nur, wenn Ihre Datei dort wirklich einen Wert hat, und nicht in jeder Position ein Rabatt von 0.

2.4 Ein Durchlauf zum Mitlesen

Aus der Beschreibung oben entstehen beide Konverter. Die Bestellung des Partners …

```
UNA:+.? '
UNB+UNOC:3+4333685000000:14+5990038744005:14+260804:1050+37626138+++++EANCOM'
UNH+1+ORDERS:D:01B:UN:EAN010'
BGM+220::9+780307082681+9'
DTM+137:20260804:102'
DTM+2:20260807:102'
NAD+BY+4335815000003::9'
NAD+DP+4335815000089::9'
NAD+SU+5990038744005::9'
RFF+YC1:0007803'
LIN+1++5996227007944:SRV'
PIA+1+50091040010:IN::92'
… (29 weitere Zeilen)
```

… wird eingehend zu dieser Inhouse-Datei:

```
K;780307082681;04.08.2026;07.08.2026;4335815000003;4335815000089;5990038744005;
P;1;5996227007944;50091040010;24;;Wiesbauer Grillmix;
P;2;5996227003656;50091090010;108;;Wiesbauer Tura SK1;
P;3;4052917470663;50091460010;60;;Wiener Wuerstchen;
P;4;4052917470687;50091500014;12;;Stuttgarter/ Tiroler SK1;
P;5;4052917470724;50091540014;12;;Griller SK1;
E;5;
```

Und dieselbe Datei wird ausgehend wieder zur Bestellung:

```
UNA:+.?
'UNB+UNOC:3+4333685000000:14+4335815000003:14+261102:2220+2611022220'UNH+1+ORDERS:D:01B:UN:EAN010'BGM+220+780307
082681+9'DTM+137:20260804:102'DTM+2:20260807:102'NAD+BY+4335815000003::9'NAD+DP+4335815000089::9'NAD+SU+59900387
44005::9'LIN+1++5996227007944:SRV'PIA+1+50091040010:SA::91'IMD+F++:::Wiesbauer
Grillmix'QTY+21:24'LIN+2++5996227003656:SRV'PIA+1+50091090010:SA::91'IMD+F++:::Wiesbauer Tura
SK1'QTY+21:108'LIN+3++4052917470663:SRV'PIA+1+50091460010:SA::91'IMD+F++:::Wiener
Wuerstchen'QTY+21:60'LIN+4++4052917470687:SRV'PIA+1+50091500014:SA::91'IMD+F++:::Stuttgarter/ Tiroler
SK1'QTY+21:12'LIN+5++4052917470724:SRV'PIA+1+50091540014:SA::91'IMD+F++:::Griller
SK1'QTY+21:12'UNS+S'CNT+2:5'UNT+30+1'UNZ+1+2611022220'
```

Was dabei auffällt, ist gewollt: Die zweite Bestellung ist nicht Byte für Byte die erste. Der Umschlag trägt Datum, Uhrzeit und Referenz **dieses** Laufes, die GLN kommen aus der Konfiguration, und Segmente, die das Inhouse-Format nicht führt – im Beispiel `RFF`, `MEA` und die ungarische Artikelbezeichnung – sind nicht wieder da. Ein Konverter gibt nur weiter, was in der Beschreibung steht.

2.5 Lieferavis: Sendung, Paletten, Positionen

Das Lieferavis (DESADV) sagt dem Kunden, was auf welcher Palette kommt. Die mitgelieferte Vorlage beschreibt das so: Kopf `K`, Sendung `S`, je Palette ein Packstück `SSCC` und darunter seine Positionen `P`:

```
K;Bestellnummer;Lieferscheinnummer;Bestelldatum;Lieferdatum;Käufer;Warenempfänger;Name;Strasse;PLZ;Ort;Land;
S;Anzahl Paletten;Sendungsgewicht Brutto;Sendungsgewicht Netto;
SSCC;LfdNr der Palette;Nummer der Versandeinheit (NVE oder SSCC);Bruttogewicht;Nettogewicht;
P;Positionsnummer;GTIN;Artikelnummer;Artikelbezeichnung;Liefermenge;Einheit;Losnummer;Mindesthaltbarkeitsdatum;
P;Positionsnummer;GTIN;Artikelnummer;Artikelbezeichnung;Liefermenge;Einheit;Losnummer;Mindesthaltbarkeitsdatum;
SSCC;LfdNr der Palette;Nummer der Versandeinheit (NVE oder SSCC);Bruttogewicht;Nettogewicht;
P;Positionsnummer;GTIN;Artikelnummer;Artikelbezeichnung;Liefermenge;Einheit;Losnummer;Mindesthaltbarkeitsdatum;
```

Eine Lieferung aus den mitgelieferten Testdaten (`INHOUSE\DESADV_OUT.CSV`) – zwei Paletten mit fünf Positionen, teils mit Charge und Mindesthaltbarkeitsdatum:

```
K;PO-2026-100234;LS-2026-500123;06.10.2026;14.10.2026;4012345000009;4012345111118;Muster Handels GmbH Lager
Nord;Lagerstrasse 7;4020;Linz;AT;
S;2;710.5;662.5;
SSCC;1;340123450000000017;412.5;387.5;
P;1;4001234567890;10001;Schrauben M8x40 verzinkt;500;PCE;;;
P;2;4001234567891;10002;Kabel NYM-J 3x1,5 mm2;250;MTR;L2026-0815;;
P;3;4001234567892;10003;Wandfarbe weiss matt 10 Liter;24;PCE;L2026-0921;30.09.2028;
SSCC;2;340123450000000024;298.0;275.0;
P;4;4001234567893;10004;Industriereiniger Konzentrat;60;LTR;L2026-1002;31.03.2029;
P;5;4001234567894;10005;Handschuhe Nitril Gr. L (Karton a 100 Stk);12;CT;;;
```

Daraus wird diese Nachricht (gezeigt ohne Umschlag, ein Segment je Zeile):

```
UNH+1+DESADV:D:01B:UN:EAN007'
BGM+351+LS-2026-500123+9'
DTM+137:20261102:102'
DTM+17:20261014:102'
RFF+ON:PO-2026-100234'
DTM+171:20261006:102'
NAD+BY+4012345000009::9'
NAD+DP+4012345111118::9++Muster Handels GmbH Lager Nord+Lagerstrasse 7+Linz++4020+AT'
CPS+1'
PAC+2++201'
MEA+PD+AAB+KGM:710.5'
MEA+PD+AAA+KGM:662.5'
CPS+2+1'
PAC+1++201'
MEA+PD+AAB+KGM:412.5'
MEA+PD+AAA+KGM:387.5'
PCI+33E'
GIN+BJ+340123450000000017'
LIN+1++4001234567890:SRV'
PIA+1+10001:SA::91'
IMD+F++:::Schrauben M8x40 verzinkt'
QTY+12:500:PCE'
LIN+2++4001234567891:SRV'
PIA+1+10002:SA::91'
IMD+F++:::Kabel NYM-J 3x1,5 mm2'
QTY+12:250:MTR'
PCI+36E'
GIN+BX+L2026-0815'
LIN+3++4001234567892:SRV'
PIA+1+10003:SA::91'
IMD+F++:::Wandfarbe weiss matt 10 Liter'
QTY+12:24:PCE'
PCI+36E'
DTM+361:20280930:102'
GIN+BX+L2026-0921'
CPS+3+1'
PAC+1++201'
MEA+PD+AAB+KGM:298'
MEA+PD+AAA+KGM:275'
PCI+33E'
GIN+BJ+340123450000000024'
LIN+4++4001234567893:SRV'
PIA+1+10004:SA::91'
IMD+F++:::Industriereiniger Konzentrat'
QTY+12:60:LTR'
PCI+36E'
DTM+361:20290331:102'
GIN+BX+L2026-1002'
LIN+5++4001234567894:SRV'
PIA+1+10005:SA::91'
IMD+F++:::Handschuhe Nitril Gr. L (Karton a 100 Stk)'
QTY+12:12:CT'
CNT+2:5'
UNT+54+1'
```

Was der Konverter dabei selbst tut:

- **Die Versandstufen zählt er:** `CPS+1` ist die Sendung, jede Palette bekommt die nächste Stufe darunter (`CPS+2+1`, `CPS+3+1` …). Die Spalte „LfdNr der Palette" wird dafür nicht gebraucht.
- **Paletten sind EUR-Paletten** (`PAC+…++201`), Gewichte in Kilogramm (`MEA+PD+AAB` brutto, `MEA+PD+AAA` netto).
- **Die SSCC** steht als `GIN+BJ` hinter `PCI+33E`, **Charge und Mindesthaltbarkeit** als `GIN+BX` und `DTM+361` hinter `PCI+36E` – nur bei Positionen, die eines davon führen. `33E` und `36E` sind Codes von EANCOM; die Prüfung gegen das UN-Verzeichnis meldet sie nicht als Abweichung.

2.6 Rechnung: Parteien, Rabatte, Steuern

Die Rechnung (INVOIC) trägt mehr als die anderen Nachrichten: drei Parteien mit USt-IdNr., Rabatte auf Beleg und Positionen und je Steuersatz eine Steuerzeile. Die mitgelieferte Vorlage:

```
K;Rechnungsnummer;Bestellnummer;Lieferscheinnummer;Bestelldatum;Lieferdatum;Käufer;MwST;Warenempfänger;MwST;Name
;Strasse;PLZ;Ort;Land;Lieferant;MwST;Name;Strasse;PLZ;Ort;Land;Währung;Steuersatz;
KR;Rabatt in %;Wert;
P;Positionsnummer;GTIN;Artikelnummer;Artikelbezeichnung;Menge;Verrechnungseinheit;Preis;Netto
Wert;Steuersatz;Lieferscheinnummer;Lieferdatum;
PR;Rabatt in %;Wert;
P;Positionsnummer;GTIN;Artikelnummer;Artikelbezeichnung;Menge;Verrechnungseinheit;Preis;Netto
Wert;Steuersatz;Lieferscheinnummer;Lieferdatum;
PR;Rabatt in %;Wert;
TAX;Basis Steuerwert;Steuerwert;Prozentsatz;
TAX;Basis Steuerwert;Steuerwert;Prozentsatz;
SUM;Brutto Wert;Netto Wert;Basis Steuerwert;Steuer;
```

Eine Rechnung aus den Testdaten (`INHOUSE\INVOIC_OUT.CSV`), eine Sammelrechnung über zwei Lieferscheine: Die Positionen 1 bis 3 kamen mit `LS-2026-500123`, die Positionen 4 und 5 mit `LS-2026-500125`; der Kopf führt deshalb keinen Lieferschein. Dazu 2 % Rabatt auf den Beleg, 5 % auf Position 3, die keinen eigenen Steuersatz führt und deshalb die 20 % aus dem Kopf bekommt, und Position 5 mit 10 %:

```
K;RE-2026-900001;PO-2026-100234;;06.10.2026;;4012345000009;ATU12345678;4012345111118;ATU12345678;Muster Handels
GmbH Lager Nord;Lagerstrasse 7;4020;Linz;AT;4012345999990;ATU87654321;LeNic Industriebedarf
GmbH;Industriestrasse 12;1100;Wien;AT;EUR;20;
KR;2;40.42;
P;1;4001234567890;10001;Schrauben M8x40 verzinkt;500;PCE;0.12;60.00;20;LS-2026-500123;14.10.2026;
P;2;4001234567891;10002;Kabel NYM-J 3x1,5 mm2;250;MTR;1.45;362.50;20;LS-2026-500123;14.10.2026;
P;3;4001234567892;10003;Wandfarbe weiss matt 10 Liter;24;PCE;39.90;909.72;;LS-2026-500123;14.10.2026;
PR;5;47.88;
P;4;4001234567893;10004;Industriereiniger Konzentrat;60;LTR;8.50;510.00;20;LS-2026-500125;16.10.2026;
P;5;4001234567894;10005;Handschuhe Nitril Gr. L (Karton a 100 Stk);12;CT;14.90;178.80;10;LS-2026-
500125;16.10.2026;
TAX;1805.38;361.08;20;
TAX;175.22;17.52;10;
SUM;2359.20;2021.02;1980.60;378.60;
```

Die Nachricht dazu:

```
UNH+1+INVOIC:D:01B:UN:EAN011'
BGM+380+RE-2026-900001+9'
DTM+137:20261102:102'
RFF+ON:PO-2026-100234'
DTM+171:20261006:102'
NAD+BY+4012345000009::9'
RFF+VA:ATU12345678'
NAD+DP+4012345111118::9++Muster Handels GmbH Lager Nord+Lagerstrasse 7+Linz++4020+AT'
RFF+VA:ATU12345678'
NAD+SU+4012345999990::9++LeNic Industriebedarf GmbH+Industriestrasse 12+Wien++1100+AT'
RFF+VA:ATU87654321'
CUX+2:EUR:4'
ALC+A+++1+DI'
PCD+1:2'
MOA+204:40.42'
LIN+1++4001234567890:SRV'
PIA+1+10001:SA::91'
IMD+F++:::Schrauben M8x40 verzinkt'
QTY+47:500:PCE'
MOA+203:60'
PRI+AAA:0.12'
RFF+DQ:LS-2026-500123'
DTM+171:20261014:102'
TAX+7+VAT+++:::20'
LIN+2++4001234567891:SRV'
PIA+1+10002:SA::91'
IMD+F++:::Kabel NYM-J 3x1,5 mm2'
QTY+47:250:MTR'
MOA+203:362.5'
PRI+AAA:1.45'
RFF+DQ:LS-2026-500123'
DTM+171:20261014:102'
TAX+7+VAT+++:::20'
LIN+3++4001234567892:SRV'
PIA+1+10003:SA::91'
IMD+F++:::Wandfarbe weiss matt 10 Liter'
QTY+47:24:PCE'
MOA+203:909.72'
PRI+AAA:39.9'
RFF+DQ:LS-2026-500123'
DTM+171:20261014:102'
TAX+7+VAT+++:::20'
ALC+A+++1+DI'
PCD+1:5'
MOA+204:47.88'
LIN+4++4001234567893:SRV'
PIA+1+10004:SA::91'
IMD+F++:::Industriereiniger Konzentrat'
QTY+47:60:LTR'
MOA+203:510'
PRI+AAA:8.5'
RFF+DQ:LS-2026-500125'
DTM+171:20261016:102'
TAX+7+VAT+++:::20'
LIN+5++4001234567894:SRV'
PIA+1+10005:SA::91'
IMD+F++:::Handschuhe Nitril Gr. L (Karton a 100 Stk)'
QTY+47:12:CT'
MOA+203:178.8'
PRI+AAA:14.9'
RFF+DQ:LS-2026-500125'
DTM+171:20261016:102'
TAX+7+VAT+++:::10'
UNS+S'
CNT+2:5'
MOA+77:2359.2'
MOA+79:2021.02'
MOA+124:378.6'
MOA+125:1980.6'
TAX+7+VAT+++:::20'
MOA+125:1805.38'
MOA+124:361.08'
TAX+7+VAT+++:::10'
```

```
MOA+125:175.22'
MOA+124:17.52'
UNT+76+1'
```

Zu beachten:

- **Die Beträge rechnet der Konverter nicht nach.** Positionsbeträge, Steuern und Summen kommen aus Ihrer Warenwirtschaft und gehen so hinaus, wie sie dort stehen.
- **Lieferschein je Position.** `Lieferscheinnummer` und `Lieferdatum` am Ende des Positionssatzes sind wahlweise. Eine Datei ohne diese Spalten – wie die zweite Rechnung der Testdaten – gilt weiter; eine Position mit leerer Lieferscheinnummer bekommt kein `RFF+DQ` und kein Datum. Ein `Lieferdatum` im Positionssatz ohne `Lieferscheinnummer` bleibt offen, denn sein `DTM+171` steht in der Gruppe des Lieferscheins.
- **Je Position ein Rabatt.** Folgen einer Position zwei Rabattsätze, gilt der letzte. Ebenso je Position ein Zuschlag.
- **Zuschläge** stehen wie Rabatte in der Nachricht, aber als `ALC+C` mit `PCD+2` (Prozent) und `MOA+23` (Betrag). Die Art des Zuschlags ist `ABK` (Verschiedenes); wissen Sie, was er ist, setzen Sie den genaueren Code ein – etwa `FC` für Fracht oder `PC` für Verpackung.
- **Eine Lieferbedingung** – Spalte `Lieferbedingung` oder `Incoterm` – geht in jeder Nachricht als Text in `TOD+6++:::frei Haus`. Verlangt Ihr Partner den Incoterm als Code (`TOD+6++EXW`), ändern Sie den Zielsatz `SEG-TOD` entsprechend.
- **Führt Ihre Datei kein Rechnungsdatum**, steht im `DTM+137` der Tag der Wandlung. Soll es das Datum aus der Warenwirtschaft sein, ergänzen Sie im Kopf eine Spalte „Rechnungsdatum“.

2.7 Den Entwurf von Hand weiterschreiben

Der Entwurf kommt **ungespeichert** in den Editor. Was danach zu tun bleibt:

1. **Die offenen Zeilen füllen.** Jede nicht erkannte Spalte steht als Kommentar da. Für die Gegenrichtung ist die Zuweisung eine Zeile: `MEIN-FELD = GET_VALUE (8)` . Für die eingehende Richtung hilft der Reiter **Beispieldatei**: Schreibmarke auf den Wert setzen, und der Editor schlägt das Muster vor, das ihn liest – etwa `RFF+YC1:VALUE` .
2. **Die Qualifier prüfen.** Der Entwurf nimmt die üblichen an: `220` für die Bestellung (`380` Rechnung, `351` Lieferavis, `231` Bestellbestätigung), `9` für das Original, `SRV` für die GTIN (in D.96A `EN`), `SA::91` für die Artikelnummer des Lieferanten, `IMD+F++:::` für die Artikelbezeichnung ohne Sprache, `DI` für einen Rabatt, `ABK` für einen Zuschlag. Ihr Partner kann andere verlangen; sein Nachrichtenhandbuch entscheidet. Fahren Sie im Editor mit der Maus über einen Qualifier, steht seine Bedeutung aus dem Verzeichnis da.
3. **Dem Konverter einen Namen geben und speichern.** Erst dann steht er in der Liste und lässt sich in einer Konfiguration auswählen.
4. **Einen Testlauf machen**, bevor eine Konfiguration darauf zeigt. Der Testlauf schreibt keine Datei und archiviert nichts.

Ein Wort zur Erwartung: Der Entwurf ist ein **Anfang, kein Ergebnis**. Er nimmt Ihnen das Gerüst ab – Segmentreihenfolge, Zähler, Freigabezeichen, Satzwechsel –, also genau die Arbeit, bei der man sich verzählt. Die letzten Prozent gehören Ihrem Partner und Ihrem Haus.

TEIL 3

Die Definitionssprache

Alle Sektionen, Befehle und Formate

Diese Beschreibung gibt wieder, was der Konverter **heute tatsächlich versteht**. Sie ist die Grundlage für den Editor und für jede Änderung an der Sprache: Was hier nicht steht, gibt es nicht. Wird die Sprache geändert, wird diese Datei im selben Zug fortgeschrieben.

3.1 Aufbau einer Definition

Eine Definition ist ein Text aus Zeilen. Maßgeblich ist das erste Zeichen einer Zeile, führende Leerzeichen zählen nicht:

Zeile beginnt mit	Bedeutung
`::` … `-CONF`	Beginn einer Sektion, zum Beispiel `::RUN-CONF`
`:`	Anweisung; der Rest der Zeile ohne den Doppelpunkt, beidseitig getrimmt
`*`	Kommentar, auch `*:` für eine auskommentierte Anweisung
alles andere	wird überlesen (Trennlinien, Überschriften, Leerzeilen)

Anweisungen, die vor der ersten Sektion stehen, gehören zu GLOBAL-CONF.

Sektionen

Sektion	Pflicht	Inhalt
`::GLOBAL-CONF`	ja	Einstellungen, Konstanten, Schablonen, Zeichensätze
`::TARGET-CONF`	ja	Aufbau des Ziels: Sätze und Felder
`::SOURCE-CONF`	ja	Aufbau der Quelle: Trennzeichen, Satzart
`::START-CONF`	nein	läuft einmal vor dem ersten Quellsatz
`::RUN-CONF`	ja	läuft einmal je Quellsatz
`::END-CONF`	nein	läuft einmal nach dem letzten Quellsatz
`::LAYOUT-CONF`	bei `FORMAT = LAYOUT`	die Ausgabevorlage als XML; ihre Zeilen gelten, wie sie dastehen – siehe 5
`::LINK-CONF`	bei `SQL-LINK`	die Verbindung zum Microsoft SQL Server (`SQL-CONNECTION` , `SQL-PASSWORD`) – siehe 4

Fehlt eine Pflichtsektion, meldet der Editor das als Befund.

Befunde

Beim Lesen werden **alle** Fehler gesammelt, nicht nur der erste. Jeder Befund nennt seine Zeile im Quelltext, zum Beispiel `Zeile 12: TARGET: Unbekannte Typangabe '35' …`. Ein Befund, der die ganze Definition betrifft, etwa eine fehlende Sektion, hat keine Zeile. Ein Text ohne schließendes Anführungszeichen, etwa `WARNING ("Achtung)`, ist in jeder Sektion ein Befund. Eine Definition mit Befunden wird nicht ausgeführt.

3.2 Werte

Schreibweise	Bedeutung	Beispiel
`"Text"`	Text; Leerzeichen innerhalb bleiben erhalten	`"NAD+BY+"`
`X"hh…"`	Zeichen über ihren Code, je zwei Hexziffern ein Zeichen	`X"0D0A"` (CR LF), `X"27"` (`'`)
`~~NAME`	Konstante aus `CONST` in GLOBAL-CONF	`~~INVOICE`
Ganzzahl	Zahl ohne Anführungszeichen	`380`, `0`
`EMPTY`	leerer Wert; siehe Zuweisungen und Bedingungen	

Vorbelegte Konstanten

Drei Konstanten stehen in keiner Definition. Ihr Wert kommt aus der Konfiguration, mit der der Konverter läuft:

Konstante	Wert	Beispiel
`~~PARTNER-GLN`	GLN des Partners der Konfiguration	`4012345000009`
`~~PARTNER-NAME`	Firmenname des Partners	`Meier KG`
`~~ABSENDER-GLN`	Absender-GLN, ein eigenes Feld der Konfiguration; darf leer bleiben, wenn die Definition sie nicht braucht	`4000001000005`

So muss keine GLN im UNB fest in der Definition stehen, und ein Konverter dient mehreren Partnern und Absendern. Die Konstanten gelten überall, wo eine Konstante stehen darf: in Zuweisungen, in `CONCAT`, in Bedingungen, als Wert einer Schablone in TARGET-CONF (`~~FILL-10 = ~~PARTNER-GLN`) und als Vorbelegung eines Feldes (`FIELD.NAD-ILN.N13 = ~~PARTNER-GLN`).

```
: RECORD.SEG-UNB
:    ~~FILL-10                    = "UNB+UNOC:3+"
:    ~~FILL-10                    = ~~ABSENDER-GLN
:    ~~FILL-10                    = ":14+"
:    ~~FILL-10                    = ~~PARTNER-GLN
:    ~~FILL-10                    = ":14"
```

- Ein `CONST` oder `TEMPLATE` mit einem dieser Namen ist ein Befund.
- Im **Testlauf** des Editors werden Partner und Absender-GLN neben der Eingabedatei angegeben. Fehlt der Wert, bricht eine Definition, die die Konstante braucht, mit einer Meldung ab. Das gilt auch im Betrieb bei leerer Absender-GLN (Rückgabewert 7). Der Name wird nie als Text ausgegeben.
- Ein `CONST`, dessen Wert eine vorbelegte Konstante ist (`CONST ~~EMPFAENGER = ~~PARTNER-GLN`), ist ebenfalls ein Befund, weil der Wert erst mit dem Lauf kommt. Die Konstante wird direkt verwendet.

Ein Zeichencode aus `X"nn"` steht für genau diesen Bytewert. Die **Ausgabe** wird Byte für Byte in Latin-1 geschrieben, sodass ein `X"84"` als Byte `0x84` in der Datei ankommt.

Die **Eingabedatei** darf auch in UTF-8 kommen (Festlegung des Auftraggebers vom 18.09.2026). Welcher Zeichensatz gilt, steht in der Konfiguration; die Vorgabe „Automatisch" erkennt ihn am Inhalt: Eine Byte-Reihenfolge-Marke entscheidet, sonst gilt UTF-8, wenn sich die ganze Datei als UTF-8 lesen lässt und mindestens ein Zeichen über 127 vorkommt, andernfalls Latin-1.

Für die Definition ändert das nichts: Ein „ä" wird in beiden Fällen zu demselben Zeichen mit der Nummer `E4`, und `CHAR-LIST` trifft es wie zuvor. Nur ein Zeichen, das es in Latin-1 überhaupt nicht gibt – etwa das Eurozeichen aus einer UTF-8-Quelle –, lässt sich nicht in die Ausgabe schreiben; das ist ein Wandlungsfehler (Rückgabewert 7) und kein stillschweigendes Fragezeichen. Wo solche Zeichen vorkommen, setzt man sie mit `CHAR-LIST` um.

3.3 GLOBAL-CONF

Eintrag	Bedeutung
`TARGET-DECIMAL-MARK = "."`	Dezimalzeichen der Ausgabe für die Formate `Z` und `-`
`DECIMAL-MARK = ","`	Dezimalzeichen für Quelle und Ausgabe, wo `SOURCE-DECIMAL-MARK` bzw. `TARGET-DECIMAL-MARK` fehlt
`SOURCE-DECIMAL-MARK = …`	Dezimalzeichen der Quelle; bei EDIFACT mit UNA gilt das Zeichen aus UNA
`CONST ~~NAME = Wert`	Konstante, zum Beispiel `CONST ~~INVOICE = 380`
`TEMPLATE ~~NAME FIELD.…`	Felddefinitionsschablone, siehe TARGET-CONF
`CHAR-LIST.NAME = …`	Zeichenliste für `TRANSLATE` und `REMOVE`, siehe unten
`VALIDATE = "OFF" / "ON" / "STRICT"`	Prüfung der EDIFACT-Seite gegen das Verzeichnis des Konverters, siehe unten. Ohne Angabe `OFF`

Ohne `TARGET-DECIMAL-MARK` und `DECIMAL-MARK` gilt der Punkt. **Jeder andere Eintrag ist ein Befund** – ein Tippfehler in GLOBAL-CONF bleibt nicht still.

Zeichenlisten

```
CHAR-LIST.INHOUSE-CHARS     = X"E4", X"C4", X"F6", "'", ":", "+", "?"
CHAR-LIST.EANCOM-CHARS      = X"84", X"8E", X"94", "?'", "?:", "?+", "??"
CHAR-LIST.NON-NUMERIC-CHARS = X"00" - X"2F", X"3A" - X"FF"
CHAR-LIST.KLEIN             = "a" - "z", "ä", "ü"
```

Eine Zeichenliste ist eine geordnete, durch Kommas getrennte Folge von Einträgen:

- `X"nn"` oder `"Text"`: ein Eintrag, der auch mehrere Zeichen tragen darf (`"?'"`)
- `X"nn" - X"nn"` oder `"a" - "z"`: ein Bereich von Zeichen

Ein Komma innerhalb von Anführungszeichen trennt nicht.

Prüfung gegen das EDIFACT-Verzeichnis

`VALIDATE` schaltet die Prüfung der EDIFACT-Seite gegen das UN/EDIFACT- Verzeichnis ein, das am Konverter gewählt ist – Segmentaufbau, Formate und Codelisten der Nachrichten, den Umschlag und die Zähler.

Wert	Bedeutung
`OFF` (Vorgabe)	Nicht prüfen. Ohne Angabe bleibt alles wie zuvor
`ON`	Prüfen und **nur melden**: Jeder Befund wird zur Warnung, die Ausgabe entsteht trotzdem (Rückgabewert 11)
`STRICT`	Prüfen und **auf Fehler bestehen**: Ein Befund vom Rang „Fehler" bricht ab, es wird nichts geschrieben (Rückgabewert 10). Hinweise und Warnungen werden wie bei `ON` gemeldet

Geprüft wird die Seite, die EDIFACT ist – bei **Outbound** die erzeugte Ausgabe, bevor sie geschrieben wird, bei **Inbound** die gelesene Eingabe. Bei Outbound verhindert ein Befund also noch eine fehlerhafte Datei beim Partner.

Geprüft wird:

- **jede Nachricht, von ihrem `UNH` bis zum `UNT`**, gegen den Aufbau ihrer Nachrichtenart im Verzeichnis;
- **der Umschlag** – `UNB` und `UNZ`, bei Gruppen `UNG` und `UNE` – gegen das Verzeichnis der Dienstsegmente;
- **die Zähler:** Der `UNT` muss die Segmente seiner Nachricht nennen, `UNH` und `UNT` einschließlich, und auf die Referenz seines `UNH` verweisen; der `UNZ` die Zahl der Nachrichten (bzw. Gruppen) und die Referenz aus dem `UNB`. Ein falscher Zähler lässt den Partner die Datei abweisen und gilt bei `STRICT` als Fehler.

Codes einer fremden Liste werden nicht geprüft: Nennt das Element eine andere herausgebende Stelle als UN/ECE, etwa EAN in `IMD+F+ANM::9`, steht der Code nicht im UN/EDIFACT-Verzeichnis; Format und Aufbau werden weiterhin geprüft.

Welches Verzeichnis gilt, steht am Konverter: fest gewählt oder aus dem UNH der Nachricht. Liegt die genannte Ausgabe nicht bei, gilt ersatzweise D.01B – wie im Editor. Lässt sich die EDIFACT-Seite gar nicht lesen, ist auch das ein Befund und wird nicht verschwiegen.

Die Befunde stehen im Log des Laufes und in der Meldung, an derselben Stelle wie `WARNING` und `ERROR` aus der Definition.

3.4 SOURCE-CONF

Eintrag	Bedeutung
`RECORD-DELIMITER = X"0D0A"`	Satztrenner. Bei `X"0D0A"` werden auch Dateien gelesen, die nur mit LF enden
`FIELD-DELIMITER = ";"`	Feldtrenner; `GET_VALUE (n)` zählt danach
`GROUP-DELIMITER = "+"`	Gruppentrenner; nur bei `FORMAT = EDIFACT`, wenn er von der Norm abweicht
`NO-RECORD-ID`	Die Quelle hat kein eigenes Satzartkennzeichen
`FIXED-LENGTH`	**Feste Satzlänge**: Die Quelle hat keine Feldtrenner, jedes Feld steht an einer festen Stelle, siehe unten
`FORMAT = CSV` / `EDIFACT` / `JSON` / `XML`	Aufbau der Quelldatei; ohne Angabe `CSV`, also eine Datei mit Trennzeichen oder fester Satzlänge. `EDIFACT` siehe unten
`RECORD-PATH = /bestellung.positionen`	Bei JSON und XML der Weg zu den Quellsätzen, siehe unten
`TEMP-TABLE.Name`	Eine **temporäre Tabelle**, gefüllt mit `INSERT` oder `SQL-LINK`, gelesen mit `WHILE NEXT`; siehe unten
`SKIP-RECORDS = 1`	Übergeht so viele Sätze am Anfang der Datei. Gedacht für eine **Überschriftenzeile**, die die Spalten nennt und kein Datensatz ist; ohne Angabe wird nichts übergangen
`RECORD.Name`	Name des Quellsatzes
`FIELD.Name.Format`	Das **erste** Feld nimmt das erste Feld jedes Quellsatzes auf und dient zur Prüfung der Satzart, etwa `IF SOURCE-DATA = "NAD"`; bei EDIFACT ist das die Segmentkennung (`IF S-SEG = "LIN"`)
weitere `FIELD`-Zeilen	Felder der Quelle an festen Stellen; werden gelesen, der Zugriff geht aber über `GET_VALUE` oder `VALUE`

Ein leerer Satz am Dateiende, der durch einen abschließenden Satztrenner entsteht, wird nicht verarbeitet.

Feste Satzlänge als Quelle

Viele Warenwirtschaftssysteme liefern eine Datei ohne Trennzeichen, in der jedes Feld an einer festen Stelle steht. `FIXED-LENGTH` in SOURCE-CONF liest eine solche Datei:

```
::SOURCE-CONF
: RECORD-DELIMITER = X"0D0A"
: FIXED-LENGTH
: RECORD.INHOUSE
:    FIELD.SATZART.X1                Stelle  1- 1
:    FIELD.KUNDEN-NR.N6              Stelle  2- 7
:    FIELD.NAME.X20                  Stelle  8-27
:    FIELD.DATUM.N8                  Stelle 28-35
```

- Die **Stellen ergeben sich aus den Größen** der vorangehenden Felder; sie werden nicht eigens angegeben.
- **Gelesen wird über den Feldnamen**: `IF SATZART = "K"`, `NAD-KDNR = KUNDEN-NR`. Alle Felder aus SOURCE-CONF sind belegt, nicht nur das erste. `GET_VALUE (n)` liefert daneben weiterhin das n-te Feld.
- **Abschließende Leerzeichen** gehören zur Auffüllung und fallen weg; führende bleiben stehen, denn dort stehen bei Zahlen die führenden Nullen.
- Ein Satz, der **vorzeitig endet**, ist kein Fehler: Die übrigen Felder sind leer – wie ein fehlendes Feld in einer CSV-Quelle.

- `FIXED-LENGTH` zusammen mit `FIELD-DELIMITER` oder `FORMAT = EDIFACT` ist ein Befund; jedes Feld braucht eine Größe.

Die **andere Richtung** braucht keinen Schalter: Ein Zielsatz in TARGET-CONF ohne Trennzeichen zwischen den Feldern ist bereits ein Satz fester Länge, weil jedes Feld ohne `.TRIM` auf seine Größe gebracht wird.

JSON und XML als Quelle

`FORMAT = JSON` bzw. `XML` liest eine Datei, die ein Baum ist statt einer Folge von Zeilen. Werte werden dort über **Wege** angesprochen.

```
::SOURCE-CONF
: FORMAT = JSON
: RECORD-PATH = /bestellung.positionen

::RUN-CONF
: BESTELLNR   = /bestellung.nummer
: GLN-BY      = /bestellung.partner[typ = "BY"].gln
: POSNR       = pos
: EAN         = artikel.ean
: MENGE       = menge
: TEXT        = texte[sprache = "DE"].text
: WRITE ZEILE
```

Ein Weg besteht aus Namen, durch Punkte getrennt:

Schreibweise	Bedeutung
`/bestellung.nummer`	von der **Wurzel** des Dokuments – so werden Kopfdaten aus jedem Satz heraus erreicht
`artikel.ean`	im **laufenden Quellsatz**
`partner[typ = "BY"].gln`	**Auswahl über ein Nachbarfeld**: der Eintrag, dessen `typ` „BY“ ist
`partner[@typ = "BY"].gln`	dasselbe in XML, wo `typ` ein **Attribut** ist
`texte[1].text`	Auswahl über die **Stelle**, gezählt ab 1 wie überall im Konverter

Die **Auswahl über ein Nachbarfeld ist der Hauptweg**, nicht die Stelle. Sie ist der direkte Verwandte der EDIFACT-Muster, die der Konverter schon kennt:

EDIFACT	JSON / XML
`NAD+BY+VALUE`	`partner[typ = "BY"].gln`
`IMD+*+*+:::VALUE::DE`	`texte[sprache = "DE"].text`
`DTM+137:VALUE`	`termine[art = "137"].datum`

Beide sagen dasselbe: *nimm den Eintrag, der so gekennzeichnet ist, und lies daraus dieses Feld.* Eine Stellenangabe bleibt möglich, ist aber selten das Gemeinte – niemand sieht der Definition an, was an Stelle 1 steht, und ein weiterer Eintrag in der Datei ließe sie still das Falsche lesen.

Was ein Quellsatz ist, legt `RECORD-PATH` fest: Jeder Treffer dieses Weges ist ein Satz, für den RUN-CONF einmal läuft. Ohne Angabe ist das ganze Dokument ein einziger Satz – richtig für eine Datei, die nur einen Vorgang enthält. Eine Satzart wie bei CSV gibt es nicht; die Struktur der Datei sagt bereits, was ein Satz ist.

Weiteres:

- **Listen sind keine eigene Ebene.** Aus `"positionen": [ {…}, {…} ]` werden zwei Einträge namens `positionen`; in XML liegen zwei `<position>` nebeneinander. Deshalb bedeutet derselbe Weg in beiden Formaten dasselbe.
- **Zahlen kommen an, wie sie in der Datei stehen** – aus `12.50` wird nicht `12.5`. Ein `null` wird zum leeren Wert.
- **Ein XML-Element ohne Kindelemente ist ein Wert**, auch wenn es Attribute trägt: `text[@sprache = "DE"]` liefert gleich den Text. Ausdrücklich geht das mit `#text` als letztem Schritt: `betrag.#text`.
- **Ein Namensraum-Präfix wird beim Lesen übergangen**: `rsm:ExchangedDocument` und `ExchangedDocument` treffen dasselbe Element.

- **Eine DTD wird abgelehnt** (`<!DOCTYPE …>`): Die Datei kommt von außen, und Entitäten darin könnten sie beim Lesen vervielfachen. Der Lauf bricht mit der Meldung ab, dass sich die XML-Datei nicht lesen lässt.
- **Trifft ein Weg nicht, bleibt das Zielfeld unverändert** – dieselbe Festlegung wie bei einem Muster mit `VALUE` .
- **Findet `RECORD-PATH` nichts**, bricht der Lauf mit einer Meldung ab und nennt, was ganz außen in der Datei steht. Ein stilles „null Sätze" wäre kaum zu bemerken.
- **Ein einzelner Name ist zuerst ein Feld:** `EAN = MENGE` übernimmt das Feld `MENGE` . Nur wenn der Name das Zielfeld selbst wäre – `MENGE = menge` –, ist der Weg in die Datei gemeint. Ein Weg mit Punkt oder Auswahl (`artikel.ean`) ist immer ein Weg.
- `FIELD-DELIMITER` , `GROUP-DELIMITER` und `FIXED-LENGTH` sind neben `FORMAT = JSON` oder `XML` ein Befund.

EDIFACT als Quelle

`FORMAT = EDIFACT` liest die Quelle als EDIFACT:

```
::SOURCE-CONF
: FORMAT = EDIFACT
: NO-RECORD-ID
: RECORD.S-SATZ
:    FIELD.S-SEG.X3
```

- **UNA** am Dateianfang legt die Trennzeichen fest und ist selbst kein Quellsatz. Ohne UNA gelten die der Norm – Segmentende `'` , Gruppentrenner `+` , Feldtrenner `:` –, oder was `RECORD-` , `GROUP-` und `FIELD-DELIMITER` abweichend angeben.
- **Freigabezeichen** ist `?` bzw. das aus UNA: `?+` ist ein Plus und trennt nicht, `?'` beendet kein Segment, `??` ist ein Fragezeichen. In den Werten, die `GET_VALUE` und `VALUE` liefern, ist die Freigabe aufgehoben. Eine Quelle ohne Gruppentrenner, etwa CSV, kennt kein Freigabezeichen.
- **Zeilenumbrüche** (CR, LF) sind nur Formatierung und werden überlesen – hinter einem Segment, vor dem Segmentende und auch mitten im Segment, wenn die Datei nach 80 Zeichen umbrochen ist. Nur wenn UNA oder `RECORD-DELIMITER` selbst CR oder LF zum Segmentende macht, trennen sie.
- **Jedes Segment braucht eine Kennung** aus Großbuchstaben und Ziffern, etwa `UNH` oder `BGM` . Beginnt ein Segment mit etwas anderem – meist, weil die Datei anders getrennt ist, als die Definition annimmt –, bricht die Wandlung mit „Segment n ist kein EDIFACT-Segment" ab, statt still nichts zu liefern.
- **Werte werden mit Mustern gelesen**, die aussehen wie die Nachricht selbst, siehe VALUE: `NAD+BY+VALUE` , `DTM+137:VALUE` , `QTY+21:VALUE` . Ein Muster prüft sein Segment und seinen Qualifier selbst. Eine Verzweigung `IF S-SEG = "DTM"` ist dafür nicht nötig.
- **`GET_VALUE` ist in einer EDIFACT-Quelle ein Befund.** Es zählt Stellen, und ohne EDIFACT-Kenntnis ist nicht zu erkennen, was `GET_VALUE (3, 1)` liest. Der Befund nennt das gleichwertige Muster. Steht die Zuweisung in einer Verzweigung auf ein Segment, trägt er einen Vorschlag, den der Editor per Doppelklick einsetzt. Jede übersprungene Stelle wird dabei zu einem `*` , und das `*` sollte man durch den Qualifier ersetzen.

geschrieben	im Befund vorgeschlagen	besser mit Qualifier
`GET_VALUE (3, 1)` in NAD	`NAD+*+VALUE`	`NAD+BY+VALUE`
`GET_VALUE (2, 2)` in DTM	`DTM+*:VALUE`	`DTM+137:VALUE`
`GET_VALUE (4, 1)` in LIN	`LIN+++++VALUE`	`LIN++++VALUE`
`GET_VALUE (4, 4)` mit Sprachprüfung in IMD	`IMD+++++:*:*:VALUE`	`IMD+++++:::VALUE::DE`

- **Erklärung im Editor:** Zeigt man mit der Maus auf ein Muster oder auf eine Segmentkennung wie `"NAD"` , erklärt der Editor sie aus dem UN/EDIFACT-Verzeichnis. Er nennt das Segment, die gelesene Stelle und die Bedeutung der Qualifier. Ist eine Testdatei gewählt, zeigt er auch, was das Muster darin liest. Das Verzeichnis (D.01B oder D.02B) richtet sich nach dem UNH der Testdatei. Die Bezeichnungen sind die des amtlichen Verzeichnisses und damit englisch.
- **Muster aus der Beispieldatei:** Der Reiter „Beispieldatei" zeigt die Testdatei, ein Segment je Zeile. Steht die Schreibmarke auf einem Wert, zeigt der Editor das Muster, das ihn liest, und was es in der ganzen Datei trifft. „Einfügen" oder ein Doppelklick setzt es in die Definition ein. Stellen davor bleiben als fester Wert stehen, wenn sie Qualifier sind,

wenn also das Verzeichnis für ihr Datenelement eine Codeliste führt. So entsteht `NAD+BY+VALUE`, bei der Positionsnummer im LIN aber `LIN+*++VALUE`. Mit „auch prüfen, was dahinter steht" kommen die kurzen Codes hinter dem Wert dazu, etwa `IMD+F+ANM:*+:::VALUE::DE`. In einer CSV-Quelle entsteht `GET_VALUE (n)`.

Zahlen aus der Quelle

Das Dezimalzeichen der Quelle kommt aus UNA, sonst aus `SOURCE-DECIMAL-MARK`, sonst aus `DECIMAL-MARK`, sonst ist es der Punkt.

Umgedeutet wird nur, wo eine Zahl gebraucht wird: in den Zahlenformaten `N`, `F`, `S`, `Z` und `-`, in `CALCULATE` und bei Vergleichen. Ein Textfeld behält `12,5`, wie es kam; so bleibt eine Referenz oder ein Artikeltext mit Komma unberührt.

UNA	Quellwert	Feld `Z15(3).TRIM`	Feld `X10.TRIM`
`UNA:+,? '`	`12,5`	`12.5`	`12,5`
`UNA:+,? '`	`12.5`	`12.5`	`12.5`
`UNA:+.? '`	`12,5`	`12,5` (keine Zahl)	`12,5`

Ausgegeben wird immer mit dem Dezimalzeichen der Ausgabe (`TARGET-DECIMAL-MARK`).

Temporäre Tabellen

Eine temporäre Tabelle sammelt Werte während des Laufs, um sie später – meist in END-CONF – noch einmal zu verarbeiten, etwa die Positionen mehrerer Bestellungen für eine eigene Datei. Sie lebt nur während eines Laufs und wird nie selbst geschrieben.

```
::SOURCE-CONF
: RECORD.QUELLE
:    FIELD.SATZART.X1
: TEMP-TABLE.POSITIONEN
:    FIELD.T-NR.X10
:    FIELD.T-EAN.X13

::RUN-CONF
: IF SATZART = "P"
:    T-NR  = TMP-NR
:    T-EAN = GET_VALUE (2)
:    INSERT POSITIONEN
: END-IF

::END-CONF
: CHANGE TARGET-FILE = "BEST_{ZAEHLER}_{DATUM}_{ZEIT}.csv"
: WHILE NEXT POSITIONEN
:    Z-NR  = T-NR
:    Z-EAN = T-EAN
:    WRITE ZEILE
: END-WHILE
```

Teil	Bedeutung
`TEMP-TABLE.Name`	legt in SOURCE-CONF eine Tabelle an. Die folgenden `FIELD`-Zeilen sind die Felder einer Zeile, bis zum nächsten `TEMP-TABLE` oder `RECORD`. Sie steht deshalb am besten hinter den Feldern der Quelle
`INSERT Name`	legt die aktuellen Werte der Tabellenfelder als neue Zeile ab und setzt die Felder auf ihre Vorbelegung zurück, wie `WRITE` es mit einem Zielsatz tut
`INSERT Name NO-INIT`	legt ab, ohne zurückzusetzen
`WHILE NEXT Name` … `END-WHILE`	läuft einmal je Zeile, in der Reihenfolge des Einfügens. Vor jedem Durchlauf stehen die Werte der Zeile in den Tabellenfeldern. Eine leere Tabelle lässt die Schleife aus

Die Tabellenfelder werden wie alle Felder über ihren Namen angesprochen. Ihr Name darf deshalb nicht zugleich ein Feld aus TARGET-CONF, aus SOURCE-CONF oder aus einer anderen Tabelle sein. Befunde sind außerdem eine Tabelle ohne Felder, ein `INSERT` oder `WHILE NEXT` zu einer Tabelle, die es nicht gibt, ein `INSERT` in die Tabelle, über die das

umgebende `WHILE` gerade läuft, und zwei verschachtelte `WHILE` über dieselbe Tabelle. Die Zahl der Zeilen wird zu Beginn der Schleife festgehalten.

Daten aus dem SQL Server (SQL-LINK)

`SQL-LINK` fragt einen **Microsoft SQL Server** ab und stellt das Ergebnis als temporäre Tabelle bereit – etwa um zu einer Kundennummer aus der Eingabe die Artikel oder die Lieferanschrift aus der Warenwirtschaft zu holen. Die Verbindung steht in einer eigenen Sektion `::LINK-CONF`, die Tabelle in SOURCE-CONF:

```
::LINK-CONF
: SQL-CONNECTION = "Server=sql01;Database=WaWi;User Id=edi_leser;TrustServerCertificate=True"
: SQL-PASSWORD   = "dpapi:AQAAANCMnd8BFdERjHoAwE…"

::SOURCE-CONF
: RECORD.QUELLE
:    FIELD.KDNR.X10
: TEMP-TABLE.ARTIKEL
:    FIELD.A-EAN.X13
:    FIELD.A-TEXT.X35

::RUN-CONF
: SQL-LINK ARTIKEL = "SELECT EAN, Bezeichnung FROM dbo.Artikel WHERE KundenNr = {KDNR}"
: WHILE NEXT ARTIKEL
:    Z-EAN  = A-EAN
:    Z-TEXT = A-TEXT
:    WRITE ZEILE
: END-WHILE
```

Teil	Bedeutung
`SQL-CONNECTION = "…"`	in LINK-CONF: die Verbindung des Konverters, wie sie der SQL Server versteht (`Server`, `Database`, `User Id`, `Integrated Security`, `Encrypt`, `TrustServerCertificate` …), **ohne Kennwort**. Je Konverter genau eine
`SQL-PASSWORD = "…"`	in LINK-CONF: das Kennwort zur Verbindung, **verschlüsselt**. Fehlt es, meldet sich der Konverter mit dem Windows-Konto an, unter dem er läuft (`Integrated Security=True`)
`SQL-LINK Tabelle = "SELECT …"`	in START-, RUN- oder END-CONF: führt die Abfrage aus und **ersetzt** die Zeilen der Tabelle durch das Ergebnis. Danach liest `WHILE NEXT` sie wie jede temporäre Tabelle

Spalten und Felder: Die erste Spalte des Ergebnisses füllt das erste Feld der Tabelle, die zweite das zweite und so fort. Die Namen der Spalten zählen nicht; ihre **Zahl muss gleich** sein, sonst bricht der Lauf mit einer Meldung ab. `NULL` wird zum leeren Wert, Zahlen kommen als Zahl, `bit` als `1` oder `0`. Ein `date` kommt als CCYYMMDD, ein `time` als HHMMSS, die übrigen Zeitstempel als CCYYMMDDHHMMSS – je nach Typ der Spalte, nicht nach dem Wert. Eine andere Form liefert die Abfrage selbst, etwa mit `FORMAT(Datum, 'dd.MM.yyyy')`.

Werte aus der Definition stehen als Platzhalter in geschweiften Klammern: `{FELD}` für ein Feld oder eine Variable, `{~~KONSTANTE}` für eine Konstante, auch `{~~PARTNER-GLN}`. Ein Platzhalter geht als **Parameter** an den Server und wird nie Teil des Abfragetextes; ein Hochkomma oder Semikolon in den Daten kann die Abfrage deshalb nicht verändern. Aus demselben Grund steht ein Platzhalter nicht in Hochkommas: `LIKE '%' + {SUCHE} + '%'`, nicht `LIKE '%{SUCHE}%'`.

Nur lesen. `SQL-LINK` führt ausschließlich **ein einzelnes** `SELECT` aus. Beim Einlesen ist ein Befund:

- eine Abfrage, die nicht mit `SELECT` beginnt – auch `WITH` und `EXEC`;
- ein Semikolon, ein Kommentar (`--`, `/* */`) und eine Variable mit `@`;
- jedes Wort, das schreibt, Rechte ändert, Code ausführt oder auf andere Server oder Dateien greift: `INSERT`, `UPDATE`, `DELETE`, `MERGE`, `TRUNCATE`, `INTO`, `DROP`, `ALTER`, `CREATE`, `EXEC`, `EXECUTE`, `DECLARE`, `SET`, `BEGIN`, `COMMIT`, `ROLLBACK`, `SAVE`, `WAITFOR`, `GO`, `USE`, `GRANT`, `REVOKE`, `DENY`, `SHUTDOWN`, `KILL`, `DBCC`, `BACKUP`, `RESTORE`, `RECONFIGURE`, `CHECKPOINT`, `ENABLE`, `DISABLE`, `BULK`, `WRITETEXT`, `UPDATETEXT`, `OPENROWSET`, `OPENDATASOURCE`, `OPENQUERY`, `OPENXML`.

Texte in Hochkommas und Namen in eckigen Klammern sind davon ausgenommen. Heißt eine Spalte wie eines dieser Wörter, steht sie in eckigen Klammern: `SELECT [Update] FROM …`. Im Lauf wird dieselbe Prüfung vor jeder Abfrage

wiederholt; die Abfrage läuft zudem in einer Transaktion, die **immer zurückgerollt** wird, und meldet der Server geänderte Zeilen, bricht der Lauf ab. Am sichersten ist dennoch ein Datenbankbenutzer, der nur lesen darf (Rolle `db_datareader`).

Das Kennwort schreibt man im Editor zunächst lesbar hin – `SQL-PASSWORD = "geheim"`. Beim **Speichern** und vor dem **Testlauf** ersetzt der Editor es durch `"dpapi:…"`: verschlüsselt mit Windows (DPAPI) und an den Rechner gebunden, wie das Kennwort des Mailversands. Editor und Konverter lesen es unter jedem Benutzerkonto dieses Rechners. Eine Definition, die auf einen anderen Rechner kopiert wird, trägt das Kennwort nicht mit; dort wird es neu eingegeben und gespeichert. Ein Kennwort im Klartext ist bis dahin ein Befund, ebenso ein `Password=` oder `Pwd=` in `SQL-CONNECTION`.

Weitere Befunde und Abbrüche:

Fall	Folge
`SQL-LINK` ohne `SQL-CONNECTION`, zu einer Tabelle, die es nicht gibt, in `WHILE NEXT` über dieselbe Tabelle, mit einem unbekannten Platzhalter	Befund
`SQL-CONNECTION` oder `SQL-PASSWORD` doppelt, leer oder ohne Anführungszeichen, in TARGET-CONF statt LINK-CONF; `SQL-PASSWORD` ohne `SQL-CONNECTION`; ein anderer Eintrag in LINK-CONF	Befund
Server nicht erreichbar, Anmeldung oder Abfrage scheitert, Spaltenzahl stimmt nicht, Kennwort lässt sich auf diesem Rechner nicht entschlüsseln	Wandlungsfehler (Rückgabewert 7); die Meldung nennt die Tabelle und den Text des Servers
mehr als 100 000 Zeilen oder länger als 60 Sekunden	Wandlungsfehler; eine Bedingung in `WHERE` grenzt die Abfrage ein

Jeder Aufruf von `SQL-LINK` fragt den Server erneut; in RUN-CONF also einmal je Quellsatz. Ein Wert, der für alle Sätze gleich ist, wird besser einmal in START-CONF geholt.

3.5 TARGET-CONF

```
: RECORD.SEG-NAD
:    ~~FILL-10                          = "NAD+BY+"
:    FIELD.NAD-ILN.N13
:    ~~FILL-10                          = "+"
:    FIELD.NAD-NAME.X35.TRIM
:    ~~SEG-END
```

Eintrag	Bedeutung
`RECORD.Name`	Beginn eines Zielsatzes (Segments)
`FIELD.Name.FormatGröße[(Nachkommastellen)][.Attribut][ = Vorbelegung]`	ein Feld des Satzes
`~~SCHABLONE [= Wert]`	eine Konstante aus einer Schablone (`TEMPLATE`); ohne Wert gilt deren Vorbelegung. Sie geht **unverändert** in die Ausgabe
`FORMAT = CSV` / `JSON` / `XML`	Aufbau der Zieldatei; ohne Angabe `CSV`, also Sätze hintereinander
`FORMAT = LAYOUT`	XML nach der Ausgabevorlage in LAYOUT-CONF, siehe unten
`NAMESPACE Präfix = "URI"`	Bei `FORMAT = XML`: vereinbart ein Präfix für die Wege; ohne Präfix den Standard-Namensraum, siehe unten
`… -> Weg`	Bei einem Baumziel: wohin der Satz bzw. das Feld gehört, siehe unten

Ein Satz wird in der Reihenfolge seiner Bestandteile zusammengesetzt. Felder erscheinen in der Form, die ihr Format und Attribut verlangen; Konstanten unverändert.

JSON und XML als Ziel

Ein Baum hat keine Reihenfolge, aus der sich ergäbe, wohin ein Feld gehört. Deshalb sagt es bei `FORMAT = JSON` oder `XML` jeder Satz und jedes Feld mit einem **Pfeil**:

```
::TARGET-CONF
: FORMAT = JSON
: RECORD.KOPF     -> /bestellung
:    FIELD.K-NUMMER.X35       -> nummer
:    FIELD.K-DATUM.N8         -> datum
: RECORD.POSITION -> /bestellung.positionen[]
:    FIELD.P-POSNR.Z6         -> pos
:    FIELD.P-EAN.N13          -> artikel.ean
:    FIELD.P-ARTNR.X35        -> artikel.nr
:    FIELD.P-MENGE.Z15(3)     -> menge
:    FIELD.P-BEMERKUNG.X70.IF-CHANGED  -> bemerkung
```

ergibt

```
{ "bestellung": {
    "nummer": "780307082681", "datum": "20260804",
    "positionen": [
      { "pos": "1", "artikel": { "ean": "5996227007944", "nr": "50091040010" },
        "menge": "24", "bemerkung": "eilig" } ] } }
```

Die Wege sind dieselben wie auf der Quellseite:

- **Am Satz** steht, wohin jedes `WRITE` einen Eintrag hängt. Der letzte Schritt ist der Name des Eintrags, davor steht, wo er hingehört; fehlende Zwischenstufen wie `artikel` entstehen von selbst.
- **`[]` am Ende** sagt, dass dort eine **Liste** steht. Dann entsteht auch bei einer einzigen Position eine Liste – sonst bekäme das lesende System bei einer Bestellung mit einer Position etwas anderes als bei einer mit zweien. Ohne `[]` entsteht ein einzelner Eintrag, wie beim Kopfsatz.
- **Am Feld** steht der Weg im Eintrag. Mit Schrägstrich gilt er von der Wurzel – dort landen Kopfdaten, die es nur einmal gibt. Wird derselbe Weg mehrfach beschrieben, gilt der letzte Wert.
- **In XML wird ein Weg mit `@` ein Attribut**: `-> @nr` schreibt `<position nr="1">`. Hat der Baum ganz außen mehrere Einträge, kommt ein Element `daten` darum, damit die Datei gültiges XML ist.
- **`#text` ist der Text eines Elements.** Enden ein Weg mit `#text` und einer mit `@` beim selben Element, trägt es beides: `-> ram:TaxTotalAmount.#text` und `-> ram:TaxTotalAmount.@currencyID` ergeben `<ram:TaxTotalAmount currencyID="EUR">19.00</ram:TaxTotalAmount>`. `#text` steht nur als letzter Schritt eines Feldweges. Es darf auch fehlen: `-> ram:TaxTotalAmount` und `-> ram:TaxTotalAmount.@currencyID` ergeben dasselbe, in welcher Reihenfolge auch immer.

Namensräume

Viele XML-Formate – etwa ZUGFeRD/XRechnung (CII) – verlangen, dass jedes Element zu einem Namensraum gehört. `NAMESPACE` vereinbart dafür ein Präfix, das dann in den Wegen steht:

```
::TARGET-CONF
: FORMAT = XML
: NAMESPACE rsm = "urn:un:unece:uncefact:data:standard:CrossIndustryInvoice:100"
: NAMESPACE ram = "urn:un:unece:uncefact:data:standard:ReusableAggregateBusinessInformationEntity:100"
: NAMESPACE udt = "urn:un:unece:uncefact:data:standard:UnqualifiedDataType:100"
: RECORD.KOPF -> /rsm:CrossIndustryInvoice.rsm:ExchangedDocument
:    FIELD.K-NUMMER.X35       -> ram:ID
:    FIELD.K-DATUM.X8         -> ram:IssueDateTime.udt:DateTimeString.#text
:    FIELD.K-DATUMFORMAT.X3   -> ram:IssueDateTime.udt:DateTimeString.@format
```

ergibt

```
<rsm:CrossIndustryInvoice xmlns:rsm="urn:…:CrossIndustryInvoice:100" xmlns:ram="urn:…" xmlns:udt="urn:…">
  <rsm:ExchangedDocument>
    <ram:ID>RG106</ram:ID>
    <ram:IssueDateTime>
      <udt:DateTimeString format="102">20261026</udt:DateTimeString>
    </ram:IssueDateTime>
  </rsm:ExchangedDocument>
</rsm:CrossIndustryInvoice>
```

- Die Vereinbarungen stehen am äußersten Element, mit den Präfixen aus der Definition.
- `NAMESPACE = "URI"` **ohne Präfix** vereinbart den Standard-Namensraum. Er gilt für jedes Element ohne Präfix; ein Attribut ohne Präfix gehört wie in XML üblich zu keinem Namensraum.
- Das Präfix `xml` (etwa `@xml:lang`) ist von XML selbst vergeben und braucht keine Vereinbarung.
- Die Reihenfolge der Elemente folgt der Reihenfolge, in der sie geschrieben werden. Ein Schema, das sie vorschreibt – wie bei CII –, bestimmt also, in welcher Reihenfolge die Sätze geschrieben werden müssen.

Kurznamen, Blöcke und feste Werte

Bei tief geschachtelten Formaten wie CII werden die Wege lang und wiederholen sich. Vier Schreibweisen halten TARGET-CONF lesbar; sie ändern nichts an der erzeugten Datei:

Eintrag	Bedeutung
`PATH NAME = /weg`	ein Kurzname für einen Weg von der Wurzel. Vorne in jedem Weg – auch in einem weiteren `PATH` – steht dann `$NAME` dafür
`NODE weg` … `END-NODE`	ein Block in einem Satz: Jeder Weg darin ohne Schrägstrich hängt am Weg des Blocks. Blöcke dürfen ineinander stehen
`FIX weg = "Wert"`	ein fester Wert ohne eigenen Feldnamen, etwa `FIX ram:TypeCode = "VAT"`
`weg[@attribut="Wert"]`	am Ende eines Feldweges: ein festes Attribut am Element des Feldes. Es erscheint nur, wenn das Feld erscheint

```
: PATH RECHNUNG    = /rsm:CrossIndustryInvoice
: PATH TRANSAKTION = $RECHNUNG.rsm:SupplyChainTradeTransaction

: RECORD.POSITION -> $TRANSAKTION.ram:IncludedSupplyChainTradeLineItem
:    FIELD.PO-NR.X6                    -> ram:AssociatedDocumentLineDocument.ram:LineID
:    NODE ram:SpecifiedTradeProduct
:       FIELD.PO-GTIN.X14.IF-CHANGED   -> ram:GlobalID[@schemeID="0160"]
:       FIELD.PO-NAME.X100             -> ram:Name
:    END-NODE
:    NODE ram:SpecifiedLineTradeSettlement.ram:ApplicableTradeTax
:       FIX ram:TypeCode               = "VAT"
:       FIELD.PO-SATZ.Z6(2)            -> ram:RateApplicablePercent
:    END-NODE
```

- Ein Weg mit Schrägstrich beginnt auch im Block an der Wurzel; ein Block mit Schrägstrich (oder `$NAME`) ebenso. So schreibt ein Satz Kopfdaten an anderer Stelle mit.
- Ohne GTIN entfällt `ram:GlobalID` samt `schemeID` – ein Hilfsfeld für das Attribut und ein `IF … NOT EMPTY` in RUN-CONF braucht es nicht.
- Ein Satz endet mit dem nächsten `RECORD`; ein Block muss bis dahin mit `END-NODE` geschlossen sein.
- Für eine ZUGFeRD-Rechnung ist die **Ausgabevorlage** (unten) der einfachere Weg: Die Reihenfolge des Schemas steht dann in der Vorlage, nicht in RUN-CONF. So arbeiten die Muster `Samples\ZUGFERD-CSV.DEF` und `Samples\ZUGFERD-INVOIC.DEF`.

Aufgefüllt wird in einem Baumziel nicht. In einer JSON- oder XML-Datei liest das Zielsystem Werte, keine Spalten; eine Kundennummer als `"0000000000024"` oder ein Name mit angehängten Leerzeichen wäre dort falsch. Die Größe gilt weiterhin als **Höchstmaß** – zu lang bleibt ein Fehler –, und die Nachkommastellen werden wie sonst gerundet. Ein Feld mit `.IF-CHANGED` behält seine Bedeutung: Steht es noch im Initialisierungszustand, erscheint es gar nicht. Aufgefüllt wird es

aber genauso wenig wie jedes andere – ein Datum in einem `N8.IF-CHANGED` steht als `20260804` in der Datei, nicht als `0020260804`.

Befunde: Ein Satz oder Feld ohne Pfeil, ein Pfeil ohne `FORMAT = JSON` / `XML`, etwas hinter dem Pfeil, das kein Weg ist, und eine Konstante (`~~SCHABLONE`) im Baumziel – Trennzeichen und Satzenden gibt es dort nicht. Ebenso ein Präfix ohne `NAMESPACE`, ein Präfix, das zweimal vereinbart wird, `NAMESPACE` ohne `FORMAT = XML` und `#text`, das nicht der letzte Schritt eines Feldweges ist. Bei den Kurzschreibweisen: ein unbekannter oder doppelter `PATH`, ein `PATH`, der nicht an der Wurzel beginnt, `END-NODE` ohne `NODE`, ein offener `NODE` am Satzende, `NODE` oder `FIX` vor dem ersten `RECORD` und eine Auswahl in eckigen Klammern, die kein festes Attribut am Ende eines Feldweges ist.

Ausgabevorlage (FORMAT = LAYOUT)

Bei einem XML-Ziel mit Pfeilen entsteht die Datei in der Reihenfolge der `WRITE`. Schreibt ein Schema die Reihenfolge der Elemente vor – wie CII für ZUGFeRD und XRechnung –, muss RUN-CONF dann dem Schema folgen statt der Eingabe. Eine **Ausgabevorlage** dreht das um: Sie ist die Zieldatei selbst, mit Platzhaltern. RUN-CONF sammelt nur die Werte; gefüllt wird die Vorlage **einmal am Ende** des Laufs. In welcher Folge die Sätze in der Eingabe kommen, spielt keine Rolle.

```
::TARGET-CONF
: FORMAT = LAYOUT
: RECORD.RECHNUNG
:    FIELD.KO-NUMMER.X35
: RECORD.POSITION
:    FIELD.PO-NR.X6
:    FIELD.PO-NAME.X100

::RUN-CONF
: IF SATZART = "K"
:    KO-NUMMER = GET_VALUE (2)
: END-IF
: IF SATZART = "P"
:    PO-NR     = GET_VALUE (2)
:    PO-NAME   = GET_VALUE (3)
:    WRITE POSITION
: END-IF

::LAYOUT-CONF
<rechnung xmlns:lenic="urn:lenic:edikonverter:layout">
  <nummer>{KO-NUMMER}</nummer>
  <position lenic:for-each="POSITION" nr="{PO-NR}">
    <name>{PO-NAME}</name>
  </position>
</rechnung>
```

Vorlage und Definition

Die Vorlage gehört zur Definition: TARGET-CONF mit den Sätzen und Feldern, LAYOUT-CONF mit der XML und dazu SOURCE-CONF und RUN-CONF für die Quelle – alles in einem Konverter. Was er ausgibt, steht also in ihm selbst; einen anderen Konverter nennt er nicht. Ein Name hinter `FORMAT = LAYOUT` ist ein Befund.

Bekommt eine zweite Quelle dieselbe Rechnung, übernimmt ihre Definition TARGET-CONF und LAYOUT-CONF. Die Felder sind die Schnittstelle: Sind sie gleich, lässt sich auch die LAYOUT-CONF austauschen, etwa für ein anderes Profil.

Sätze

TARGET-CONF beschreibt Sätze und Felder mit ihrem Format – ohne Pfeile, `PATH`, `NODE`, `FIX`, `NAMESPACE` und Konstanten. Ein Satz ist eins von beidem:

Satz	erkannt daran	`WRITE`	in der Vorlage
Einzelsatz	kein `lenic:for-each` nennt ihn	nicht erlaubt (Befund)	seine Felder mit dem Wert, den sie am Ende des Laufs haben
Wiederholsatz	ein Element trägt `lenic:for-each="SATZ"`	je `WRITE` ein Eintrag	das Element einmal je Eintrag, mit dessen Werten

Ein `WRITE` merkt sich die Werte des Satzes und setzt ihn wie gewohnt zurück. **Ein Feld, dem nichts zugewiesen wurde, ist leer – auch ein Zahlenfeld.** Die Nullen für leere Zahlenfelder gelten in einer Vorlage nicht, sonst entfiele nie ein Element; wer eine 0 will, weist sie zu oder belegt das Feld vor.

LAYOUT-CONF

- Die Vorlage beginnt mit der ersten Zeile, die mit `<` anfängt; davor sind Trennlinien und Kommentare erlaubt. Ab da gilt jede Zeile so, wie sie dasteht, ohne `:` am Anfang, bis zur nächsten Sektion. Kommentare sind XML-Kommentare.
- Sie ist **wohlgeformtes XML** – sonst ein Befund mit Zeile.
- Die Steuerattribute gehören zum Namensraum `urn:lenic:edikonverter:layout`, üblich mit dem Präfix `lenic`. Vereinbarung und Attribute erscheinen nicht in der Ausgabe; die übrigen Namensräume bleiben, wie sie in der Vorlage stehen.

Schreibweise	wird zu
`{FELD}`, `{SATZ.FELD}`	der Wert des Feldes – wie im Baumziel: Größe als Höchstmaß, gerundet, nicht aufgefüllt; `&`, `<`, `"` werden für XML umgesetzt
`{~~KONSTANTE}`	der Wert einer Konstanten, etwa `{~~PARTNER-GLN}`
`{{`, `}}`	eine geschweifte Klammer

Platzhalter stehen im Text eines Elements und in Attributwerten, auch mehrere und zusammen mit festem Text.

Steuerattribute

Attribut	Wirkung
`lenic:for-each="SATZ"`	das Element einmal je geschriebenem Eintrag, in der Folge der `WRITE`; ohne Eintrag gar nicht. Darin stehen die Felder dieses Satzes, der äußeren Schleifen und aller Einzelsätze. Ein Satz darf an mehreren Stellen wiederholt werden
`lenic:where="Bedingung"`	in einer Schleife in einer Schleife: zu welchem äußeren Eintrag ein innerer gehört, siehe unten
`lenic:distinct="FELD"`	mit `for-each`: je verschiedenem Wert des Feldes nur der erste Eintrag; leere Werte zählen nicht – etwa für eine Liste der Lieferscheine aller Positionen
`lenic:if="Bedingung"`	das Element nur, wenn die Bedingung wahr ist; mit `for-each` je Eintrag
`lenic:keep-empty="true"`	das Element bleibt auch leer stehen

Eine Bedingung ist aufgebaut wie bei `IF`; Text steht in einfachen Anführungszeichen (`lenic:if="KO-ART = '384'"`), und statt `<` schreibt man im Attribut `<` oder dreht den Vergleich um.

Zu welchem äußeren Eintrag ein innerer gehört: Ohne `where` zu dem, der **zuletzt davor** geschrieben wurde – ein Nachlass gehört zur Position davor. Wurde vorher keiner geschrieben, bricht der Lauf ab (Rückgabewert 7). Mit `where` entscheidet die Bedingung, gleich wann beide geschrieben wurden: `lenic:where="PR-POSITION = PO-NR"`. Sie vergleicht Felder des inneren Eintrags mit denen der **nächsten** äußeren Schleife und den Einzelsätzen. Passt ein Eintrag zu keinem äußeren oder zu mehreren, bricht der Lauf mit einer Meldung ab, die ihn nennt – ein Nachlass verschwindet nicht still.

Weglassen

Ein Element, das nach dem Einsetzen leer ist, entfällt – kein Text außer Leerraum, kein Kindelement. Attribute zählen nicht: Ohne GTIN entfällt `<ram:GlobalID schemeID="0160">{PO-GTIN}</ram:GlobalID>` ganz. Das setzt sich nach außen fort; ein Element, dessen Kinder alle entfallen sind, entfällt ebenfalls. Ein Attribut, das nur aus leeren Platzhaltern bestand, entfällt auch. `lenic:if` braucht es deshalb nur, wo fester Inhalt ein Element nie leer werden ließe – etwa `<udt:Indicator>false</udt:Indicator>`.

Ausgabe

Der XML-Kopf nennt den Zeichensatz der Ausgabedatei, wie beim XML-Ziel; ein Kopf in der Vorlage wird ersetzt. Eingerückt wird neu. Ein Lauf ergibt ein Dokument; END-CONF sieht schon alle Werte und ist der Ort für Prüfungen über den ganzen Beleg. Ausgabeart ZUGFeRD-PDF wirkt wie bei jedem XML-Ziel, und das Profil in den PDF-Metadaten kommt aus der XML.

Befunde

Beim Lesen einer Vorlage: kein wohlgeformtes XML; ein Platzhalter, der kein Feld aus TARGET-CONF und keine vereinbarte Konstante nennt; eine offene `{` ; `for-each` mit einem unbekannten Satz oder am äußersten Element; ein Feld eines Wiederholsatzes außerhalb seiner Schleife; `where` ohne äußere Schleife oder mit einem Feld einer Schleife weiter außen; `where` oder `distinct` ohne `for-each` ; `distinct` mit dem Feld eines anderen Satzes; eine Bedingung, die keine ist; ein unbekanntes Steuerattribut; Pfeile, `PATH` , `NODE` , `FIX` , `NAMESPACE` oder Konstanten in TARGET-CONF.

Außerdem: ein Name hinter `FORMAT = LAYOUT` ; LAYOUT-CONF ohne `FORMAT = LAYOUT` ; `WRITE` eines Einzelsatzes; `CHANGE TARGET-FILE` .

Mitgeliefert werden zwei Definitionen mit eigener Vorlage: `ZUGFERD-CSV.DEF` aus einem Inhouse-CSV (Profil EN 16931) und `ZUGFERD-INVOIC.DEF` aus einer EANCOM-INVOIC (Profil EXTENDED). Dazu liegen TARGET-CONF und LAYOUT-CONF beider Profile als **Kopiervorlagen** bei, `Samples\ZUGFERD-VORLAGE-EN16931.LAYOUT` und `Samples\ZUGFERD-VORLAGE-EXTENDED.LAYOUT` – zum Übernehmen in eine Definition, nicht als eigener Konverter.

Feldnamen

Ein Feld wird über seinen Namen angesprochen. Ein vorangestellter Satzname ist erlaubt, wird aber nicht ausgewertet: `SEG-NAD.NAD-ILN` und `NAD-ILN` meinen dasselbe Feld. **Feldnamen müssen deshalb über alle Sätze eindeutig sein.**

Feldformate

Die Größe ist eine Bytegröße. Wo Komma und Vorzeichen dargestellt werden, zählen sie mit.

Format	Inhalt	Beispiel	Ergebnis
`X`	Text	`X10` bei `Meier KG`	`Meier KG␣␣`
`U`	Text, in Großbuchstaben gewandelt	`U10.TRIM` bei `Meier KG`	`MEIER KG`
`N`	nur Ziffern, führende Nullen, kein Vorzeichen	`N6` bei `1`	`000001`
`F`	wie `N` , mit virtuellem Komma: Nachkommastellen ohne Trennzeichen	`F6(2)` bei `1.1`	`000110`
`S`	wie `F` , Vorzeichen auf der ersten Stelle	`S14(4)` bei `145.75`	`+0000001457500`
`Z`	Zahl mit dargestelltem Komma, Minus bei negativen Werten, mit Leerzeichen rechtsbündig aufgefüllt	`Z8(3)` bei `-123.456`	`-123.456`
`-`	wie `Z` , das Vorzeichen steht immer	`-8(3)` bei `123.456`	`+123.456`

Weitere Regeln:

- **Überlänge ist ein Fehler.** Abgeschnitten wird nicht; die Wandlung bricht mit einer Meldung ab, die Satz, Feld, Wert und erlaubte Größe nennt.
- **Zu viele Nachkommastellen werden kaufmännisch gerundet**, von der Null weg.
- Ein negativer Wert in `N` oder `F` ist ein Fehler.
- Was sich nicht als Zahl lesen lässt, geht in Zahlenformaten unverändert hinaus. Führende und abschließende Leerzeichen werden dabei geduldet. Das Dezimalzeichen der Quelle wird verstanden, siehe „Zahlen aus der Quelle“.

Feldattribute

Attribut	Bedeutung
ohne (oder `.F`)	feste Länge – **die Vorgabe**; das Feld hat immer seine Größe
`.TRIM`	variabel: führende und abschließende Leerzeichen entfallen; bei `Z` und `-` auch nachlaufende Nullen und ein Nachkommaanteil, der null ist. Die Größe ist nur das Maximum. Entspricht EANCOM `an..35`
`.TRIM-END`	abschließende Leerzeichen entfallen
`.LEFT`	linksbündig, auch bei Zahlen
`.IF-CHANGED`	das Feld erscheint nur, wenn sein Wert vom Initialisierungszustand abweicht

Bei `N`, `F` und `S` sind die führenden Nullen Teil des Wertes; `.TRIM` findet dort nichts zum Entfernen. Für EDIFACT-Segmente sind deshalb `X….TRIM` und `Z….TRIM` die üblichen Formen, etwa `FIELD.NAD-NAME.X35.TRIM`.

Vorbelegung und Initialisierungszustand

Jedes Zielfeld hat einen Initialisierungszustand: die Vorbelegung hinter dem Gleichheitszeichen, sonst typenabhängig die Zahl `0` für `N`, `F`, `S`, `Z` und `-` und leerer Text für `X` und `U`. In der Ausgabe wird daraus etwa `00000` für `N5` oder fünf Leerzeichen für `X5`.

- Zu Beginn stehen alle Zielfelder in ihrem Initialisierungszustand.
- **Nach jedem `WRITE` gilt er für die Felder des geschriebenen Satzes wieder**, außer bei `WRITE … NO-INIT`.
- **Ein leeres Zahlenfeld wird mit Nullen geschrieben.** In den Formaten `N`, `F`, `S`, `Z` und `-` zählt ein leerer Wert als `0` – gleich, ob das Feld nie belegt wurde oder ihm ausdrücklich ein leerer Wert zugewiesen wurde, etwa mit `FELD = EMPTY` oder durch ein `GET_VALUE` auf ein leeres Quellfeld. So steht in einem Zahlenfeld immer eine lesbare Zahl, und ein Feld mit fester Länge behält seine Breite.
- Ein leeres **Textfeld** (`X`, `U`) geht dagegen leer hinaus und wird höchstens auf seine feste Länge mit Leerzeichen gebracht.

3.6 START-CONF, RUN-CONF, END-CONF

START-CONF läuft einmal vor dem ersten Quellsatz, RUN-CONF einmal je Quellsatz, END-CONF einmal nach dem letzten. Alle drei arbeiten auf **denselben Feldern**: Ein Zähler aus START-CONF lebt bis END-CONF weiter.

Die Ausgabe wird ausschließlich über `WRITE` gesteuert: Ein Zielsatz entsteht genau dann, wenn die Definition ihn schreibt.

Zuweisungen

Form: `Feld = Ausdruck`. Der Ausdruck ist genau eine der folgenden Formen:

Ausdruck	Ergebnis
`"Text"`, `X"hh"`, `~~KONSTANTE`, Ganzzahl	der Wert
`EMPTY`	leerer Wert
`anderes-Feld`	der Wert des anderen Feldes
`GET_VALUE (n)`	das n-te Feld des aktuellen Quellsatzes einer CSV-Quelle; gezählt ab 1, fehlt es, leer. **In einer EDIFACT-Quelle ein Befund**, dort gilt ein Muster mit `VALUE`, siehe 4
`GET_VALUE (g, n)`	das n-te Feld der g-ten Gruppe; eine CSV-Quelle hat genau eine Gruppe. In einer EDIFACT-Quelle ein Befund
`CONCAT Teil Teil …`	die Teile ohne Trenner aneinandergehängt; ein Teil ist `"Text"` oder ein Feld. **Die Teile werden durch Leerzeichen getrennt**, nicht durch Kommas. Wie bei `WARNING` dürfen sie in Klammern stehen: `CONCAT (A " " B)` ist dasselbe wie `CONCAT A " " B`
`CALCULATE Formel`, `CALCULATE (Formel) …`	Rechnung mit `+ - * /`, Klammern, Zahlen und Feldern; Punkt vor Strich. **Operatoren stehen frei, mit Leerzeichen ringsum**, denn Feldnamen enthalten selbst Bindestriche. Gerundet wird erst bei der Ausgabe, auf die Stellen des Zielfeldes. Teilung durch null ist ein Fehler
`DATE ()`, `DATE (Format)`	Tagesdatum; ohne Format `CCYYMMDD`. Formate: `CCYYMMDD`, `YYYYMMDD`, `YYMMDD`, `DDMMYY`, `HHMM`, `HHMMSS` – keine freien Formattexte
`TIME ()`, `TIME (Format)`	Uhrzeit; ohne Format `HHMMSS`, sonst werden `HH`, `MM`, `SS` im Formattext ersetzt: `TIME (Es ist HH:MM)`
`CHECK_DIGIT (Feld)`	GS1-Prüfziffer (Modulo 10) über die Ziffern des Feldes
`SUBSTRING (Start, Länge, Wert)`	Ausschnitt ab Stelle `Start` (gezählt ab 1), höchstens `Länge` Zeichen. Der Wert ist ein Feld, ein `"Text"`, eine `~~KONSTANTE` oder `GET_VALUE (…)`: `SUBSTRING (5, 2, GET_VALUE (3))`. Reicht der Wert nicht bis zum Ende des Ausschnitts, kommt, was da ist; beginnt er dahinter, ist er leer. Eine andere Form oder Start 0 ist ein Befund
`FORMAT-DATE (Wert, "Ausgabe")`, `FORMAT-DATE (Wert, "Eingabe", "Ausgabe")`	Stellt ein Datum um. Der Wert ist ein Feld, ein `"Text"`, eine `~~KONSTANTE` oder `GET_VALUE (…)`. Platzhalter, gleich ob groß oder klein: `CCYY` / `YYYY` / `JJJJ` Jahr, `YY` / `JJ` Jahr zweistellig, `MM` Monat, `DD` / `TT` Tag, `DDD` / `TTT` Tag im Jahr (001–366), `WW` Kalenderwoche nach ISO 8601, `HH` Stunde, `MI` Minute, `SS` Sekunde; alles andere steht so in der Ausgabe. Statt eines Formats geht auch ein EDIFACT-Formatcode: `101`, `102`, `203`, `204`, `616`. Fehlt die Eingabe, wird sie erkannt: 6, 8, 12 oder 14 Ziffern als `YYMMDD`, `CCYYMMDD`, `CCYYMMDDHHMI`, `CCYYMMDDHHMISS`, dazu `DD.MM.CCYY` und `CCYY-MM-DD`. Beispiel: `FORMAT-DATE (K-DATUM, "DD.MM.CCYY")`, `FORMAT-DATE (GET_VALUE (3), "DD.MM.CCYY", "102")`. Passt der Wert nicht genau zum Eingabeformat, zählen nur die Ziffern: `06.10.2026` wird auch mit `DDMMCCYY` gelesen, `06102026` auch mit `DD.MM.CCYY`. Ein leerer Wert bleibt leer; ein Wert, der auch so nicht passt oder kein gültiges Datum ist, bricht die Wandlung ab. Ein Format ohne Platzhalter ist ein Befund
`LEN (Wert)`	Länge des Wertes in Zeichen, als Zahl – etwa für `IF` oder `CALCULATE`. Der Wert ist ein Feld, ein `"Text"`, eine `~~KONSTANTE` oder `GET_VALUE (…)`: `LEN (GET_VALUE (3))`. Gezählt wird der Inhalt, wie er gerade im Feld steht, Leerzeichen eingeschlossen – nicht die Feldgröße aus TARGET-CONF. Ein leeres Feld hat die Länge 0. Eine andere Form ist ein Befund
`TRANSLATE Liste1 TO Liste2`	wandelt den **eigenen** Wert des Zielfeldes: der n-te Eintrag von Liste1 wird zum n-ten von Liste2; trägt Liste2 nur einen Eintrag, gilt er für alle. Der längere Eintrag geht vor; ein eingesetzter Wert wird nicht erneut gewandelt
`REMOVE Liste`	die Zeichen der Liste entfallen aus dem eigenen Wert des Zielfeldes
`Muster VALUE`	siehe unten

Was keiner dieser Formen entspricht, ist schon beim Einlesen ein **Befund**, nicht erst ein Abbruch mitten in der Wandlung. Der Befund nennt, was fehlt:

Zeile	Befund
`EAN = MNEGE`	ein Feld oder eine Variable dieses Namens gibt es nicht – ein Text gehört in Anführungszeichen
`ABSENDER = ~~ABSENDR`	eine Konstante dieses Namens gibt es nicht
`SUMME = A + B`	gerechnet wird mit `CALCULATE`: `CALCULATE A + B`
`PREIS = 3.5`	eine Zahl mit Nachkommastellen steht in Anführungszeichen oder wird mit `CALCULATE` gerechnet
`WERT = UPPER (A)`	die Funktion `UPPER` gibt es nicht

Bekannt ist ein Name, wenn er in TARGET-CONF oder SOURCE-CONF steht oder irgendwo zugewiesen, mit `INC` gezählt oder als Zählvariable von `FOR` benutzt wird. Bei einer JSON- oder XML-Quelle ist ein Name, der kein Feld ist, ein Weg in der Datei.

Eine Variable ist leer, bis sie zum ersten Mal belegt wird. So lässt sich etwas genau einmal tun, etwa beim ersten Satz einer Art:

```
: IF TMP-ERSTER = EMPTY
:    WRITE PARTNER
:    TMP-ERSTER                          = "J"
: END-IF
```

VALUE

`Feld = Muster VALUE` sucht das Muster **am Anfang des aktuellen Quellsatzes** und liefert, was ab der Stelle von `VALUE` bis zum nächsten Feld-, Gruppen- oder Satztrennzeichen steht. Ein `*` im Muster steht für beliebig viele Zeichen, und zwar für so wenige wie möglich.

Das Muster endet auf ein Trennzeichen, unmittelbar vor `VALUE`: `NAD+BY+VALUE`, `QTY+21:VALUE`. Ein Name wie `TMP-RETURN-VALUE` ist deshalb nie ein Muster, sondern immer ein Feld.

Hinter `VALUE` darf das Muster weitergehen. Dann muss auch dieser Teil passen, sonst trifft das Muster nicht. So wird ein Wert nur gelesen, wenn dahinter das Richtige steht, etwa der Artikeltext nur in deutscher Sprache:

Quellsatz	Anweisung	Ergebnis
`IMD+F+ANM::9+:::Wiesbauer Grillmix::DE`	`TEXT = IMD+***+:::VALUE::DE`	`Wiesbauer Grillmix`
`IMD+F+ANM::9+:::Grillmix (FGY) 360g::HU`	dito	trifft nicht

Quellsatz	Anweisung	Ergebnis
`UNB+UNOC:3+4012345000009:14+4000000000003:14+…`	`SENDER = UNB+*+VALUE`	`4012345000009`
dito	`EMPFAENGER = UNB+*+*+VALUE`	`4000000000003`
`DTM+137:20260911:102`	`DATUM = DTM+*:VALUE`	`20260911`
`LIN+1++4012345000009:EN`	`EAN = LIN+2+VALUE`	trifft nicht

Trifft das Muster nicht, bleibt das Feld unverändert. So lassen sich mehrere Muster nacheinander auf dasselbe Feld setzen, ohne dass ein späterer Fehlschlag einen früheren Treffer löscht:

```
: ADRESSE = NAD+CN+VALUE
: ADRESSE = NAD+DP+VALUE
```

Ein freigegebener Trenner (`?+`) beendet den Wert nicht, und im Wert ist die Freigabe aufgehoben.

Anweisungen

Anweisung	Bedeutung
`WRITE Satz`, `WRITE (Satz)`	schreibt den Zielsatz und setzt seine Felder in den Initialisierungszustand zurück
`WRITE Satz NO-INIT`, `WRITE (Satz) NO-INIT`	schreibt, ohne zurückzusetzen
`INC (Feld)`, `INC (Feld, n)`	erhöht ein Feld um 1 bzw. um n
`IF … [ELSE …] END-IF`	Verzweigung, beliebig verschachtelt
`WARNING (Text …)`	Warnung ins Protokoll und ins Log; der Text ist aufgebaut wie bei `CONCAT`, Platzhalter wie `{FELD}` setzen Werte ein, siehe unten. Die Wandlung läuft weiter, die Ausgabe wird geschrieben
`ERROR (Text …)`	Fehler ins Protokoll und ins Log, Text wie bei `WARNING`. Die Wandlung läuft weiter, damit alle Fehler gemeldet werden, aber **es wird keine Ausgabedatei geschrieben**
`INSERT Tabelle [NO-INIT]`	legt die Werte der Tabellenfelder als Zeile in einer temporären Tabelle ab, siehe 4
`WHILE NEXT Tabelle … END-WHILE`	Schleife über die Zeilen einer temporären Tabelle, siehe 4
`SQL-LINK Tabelle = "SELECT …"`	füllt eine temporäre Tabelle mit dem Ergebnis einer Abfrage an den SQL Server, nur lesend, siehe 4
`FOR (I = Start; Bedingung; Schritt) … END-FOR`	Zählschleife, siehe unten
`CHANGE TARGET-FILE = "Name"`	nur in END-CONF und höchstens einmal, nicht in einer Schleife (`WHILE` oder `FOR`): Was danach geschrieben wird, geht in eine **weitere Zieldatei**, siehe unten

Zählschleife

```
: FOR (I = 1; I <= TMP-ANZAHL; I++)
:    FTX-NR = I
:    WRITE SEG-FTX
: END-FOR
```

In den Klammern stehen, durch Semikolon getrennt:

Teil	Bedeutung
`I = Start`	Die Zählvariable bekommt zu Beginn den Startwert: eine Zahl, ein Feld oder eine `~~KONSTANTE`
Bedingung	Ein Vergleich wie bei `IF`, auch mit `AND` und `OR`. Er wird **vor jedem Durchlauf** geprüft; ist er nicht wahr, geht es hinter `END-FOR` weiter. Ist er schon zu Beginn nicht wahr, läuft der Rumpf nie
Schritt	Nach jedem Durchlauf: `I++` zählt um 1 hinauf, `I--` um 1 hinunter, `I += n` und `I -= n` um n. n ist eine Zahl, ein Feld oder eine Konstante

`FOR (I = 0; I < TMP-ANZAHL; I++)` läuft also mit 0, 1, … bis eins unter TMP-ANZAHL. Die Zählvariable ist ein gewöhnliches Feld: Der Rumpf kann sie lesen, in `WARNING` als `{I}` nennen und auch ändern – setzt er sie ans Ende, ist die Schleife nach diesem Durchlauf vorbei. Schleifen lassen sich verschachteln, jede mit ihrer eigenen Zählvariablen.

Kommt eine Schleife nach 100.000 Durchläufen nicht zu Ende, bricht die Wandlung mit einer Meldung ab, statt endlos zu laufen – meist zählt der Schritt dann in die falsche Richtung. Ein Schritt, der eine andere Variable ändert als die gezählte, ist schon beim Einlesen ein **Befund**.

Weitere Zieldatei

`CHANGE TARGET-FILE = "Name"` in END-CONF legt eine zweite Datei an. Alles, was vorher geschrieben wurde, steht in der Ausgabedatei der Konfiguration, alles danach in der weiteren Datei. Sie hat dasselbe Format (`FORMAT` in TARGET-CONF) und denselben Zeichensatz wie die Ausgabedatei.

- Der Name trägt dieselben Platzhalter wie die Ausgabedatei, siehe 7: `{ZAEHLER}`, `{ZAEHLER:6}`, `{DATUM}`, `{ZEIT}`. Ein unbekannter Platzhalter ist ein Befund. Statt des Textes darf ein Feld oder eine Konstante stehen. Ihr Wert kann aus der Eingabe stammen und ergibt deshalb **nur einen Dateinamen**: `\`, `/`, `:` und andere Zeichen, die Windows im Namen nicht erlaubt, werden zu `_`, ein Wert nur aus Punkten ebenso. Einen Ordner nennt nur ein Text in Anführungszeichen.
- Ein Name ohne Ordner, oder mit einem relativen Ordner, gilt ab dem Ordner der Ausgabedatei.
- `{ZAEHLER}` trägt dieselbe Nummer wie die Ausgabedatei, damit beide zusammengehören: `AUS_17.csv` und `BEST_17.csv`. Hat die Ausgabedatei keine Nummer, zieht die weitere Datei eine eigene.
- Eine vorhandene Datei wird nicht überschrieben. Dann wird **keine** der beiden Dateien geschrieben, und der Lauf endet mit Rückgabewert 6. Ebenso entsteht bei einem `ERROR` keine von beiden.
- Die Prüfung gegen das EDIFACT-Verzeichnis (`VALIDATE`) gilt der Ausgabedatei.
- Im Testlauf des Editors steht die weitere Datei unter der Ausgabe, mit ihrem Namen.

Platzhalter in WARNING und ERROR

Im Text von `WARNING` und `ERROR` setzt `{NAME}` den aktuellen Wert eines Feldes oder einer Variablen ein, auch mit dem Satz davor wie `{SEG-BGM.BGM-NR}`. `{~~NAME}` setzt den Wert einer Konstanten ein. Es dürfen beliebig viele sein:

```
: WARNING ("Kunde {TMP-KD-NR} ohne Namen, Auftrag {SEG-BGM.BGM-NR}")
: ERROR ("Adresse {KD-NR} fehlt für {~~PARTNER-NAME}")
```

Ein Name, den es in der Definition weder als Feld noch als belegte Variable oder Konstante gibt, ist ein Befund mit Zeile. Ein Feld, das im Lauf noch nicht belegt wurde, ergibt leer. Ersetzt wird nur im Text der Definition, geschweifte Klammern im Wert eines Feldes bleiben stehen. Die Schreibweise wie bei `CONCAT`, `WARNING ("Kunde ohne Namen: " TMP-KD-NR)`, gilt weiter.

Bedingungen

```
: IF SOURCE-DATA = "NAD"
: OR SOURCE-DATA = "NAD2"
:    …
: END-IF
```

Vergleich	Bedeutung
`A = B`	gleich, siehe unten
`A <> B`, `A != B`, `A NOT = B`	ungleich, siehe unten
`A < B`, `A > B`, `A <= B`, `A >= B`	Vergleich als Zahl; was keine Zahl ist, gilt als 0
`A = EMPTY`	A enthält nichts oder nur Leerzeichen
`A NOT EMPTY`, `A NOT = EMPTY`	Gegenteil davon

Rechts vom Vergleich steht ein Feld, ein Text in Anführungszeichen, eine `~~KONSTANTE` oder eine Zahl.

Was verglichen wird, muss ein Name oder ein Wert sein. Steht auf einer Seite etwas anderes – ein verschriebenes `NOR` statt `NOT`, ein Kommentar hinter der Bedingung –, ist das ein **Befund**. Sonst gälte der Rest der Zeile als eigener Text, die Bedingung wäre stillschweigend immer falsch, und der Satz dahinter entstünde nie. Ein Kommentar gehört deshalb in eine eigene Zeile: Ein `*` mitten in der Zeile ist keiner.

Den Namen muss es auch geben. Ein Name ist ein Feld aus TARGET-CONF oder SOURCE-CONF, eine Variable, die irgendwo zugewiesen oder mit `INC` gezählt wird, oder eine Konstante `~~NAME`. Ein verschriebener Name wie `P-TXT` statt `P-TEXT` ist ein **Befund**; ein Text ohne Anführungszeichen ebenso.

Gleich und ungleich hängen davon ab, was rechts steht:

- **Text in Anführungszeichen:** Es wird Zeichen für Zeichen verglichen. `"+1"` ist nicht `"1"`, und `"NAD"` ist immer der Text, auch wenn es ein Feld dieses Namens gibt.

- **Zahl, Feld oder Konstante:** Sind beide Seiten Zahlen, zählt der Wert. `0.00 = 0` und `+1 = 1` sind wahr. Sonst wird Zeichen für Zeichen verglichen. Ein leerer Wert ist keine Zahl, also ist leer nicht gleich `0`.

Ein Text ist nur gleich, wenn er ganz gleich ist; Texte ungleicher Länge sind nie gleich. Mehrere Vergleiche werden mit `AND` und `OR` verbunden, auch über Fortsetzungszeilen, die mit `AND` bzw. `OR` beginnen. Sie werden **von links nach rechts ohne Vorrang** ausgewertet.

3.7 Aufruf und Rückgabewerte

Der Konverter wird nur mit der ID einer gespeicherten Konfiguration aufgerufen: `LeNic.EdiKonverter.App.exe <ID>`. Richtung, Partner, Eingabe- und Ausgabedatei und der Konverter stehen in der Konfiguration. Die Rückgabewerte 0 bis 14 sind in `PROJEKTSTAND.md` aufgeführt. Jeder Lauf wird in der Datenbank protokolliert, mit Rückgabewert, Dateien, Zahl der Quell- und Zielsätze und der vollständigen Meldung samt `WARNING` und `ERROR`. Der Editor zeigt das Protokoll im Reiter „Verlauf".

Der Name der Ausgabedatei ist ein Muster mit Platzhaltern:

Platzhalter	Ergebnis
`{ZAEHLER}`	laufende Nummer der Konfiguration: `1`, `2`, …
`{ZAEHLER:6}`	dito, mit Nullen auf 6 Stellen: `000001`
`{DATUM}`	Tagesdatum `20260911`
`{ZEIT}`	Uhrzeit `100507`
`{RECHNUNGSNUMMER}`, `{RG-NO}`	die Nummer der Rechnung – nur bei EDIFACT INVOIC (`BGM`) oder ZUGFeRD/CII-XML (`ExchangedDocument/ID`)

Beispiel: `C:\EDI\aus\INVOIC_{DATUM}_{ZAEHLER:6}.edi`. Mit `{ZAEHLER}` wird die Ausgabe immer erzeugt – die Nummer wird erst nach gelungener Wandlung vergeben, eine belegte Nummer wird übersprungen. Ohne `{ZAEHLER}` wird eine vorhandene Datei nicht überschrieben.

Die Rechnungsnummer wird aus der fertigen Ausgabe gelesen, nicht aus einem Feld der Definition – gleich, wie die Definition ihre Felder nennt. Ist die Ausgabe keine Rechnung, trägt sie keine Nummer oder stehen mehrere Rechnungen mit verschiedenen Nummern darin, endet der Lauf mit Rückgabewert 7, und es wird nichts geschrieben. Zeichen, die in einem Dateinamen nicht erlaubt sind, werden zu `_`.

Log eines Laufs

Jeder Lauf zu einer bekannten Konfiguration schreibt eine Logdatei: die Angaben des Laufs mit Rückgabewert und Meldung, die Texte aus `WARNING` und `ERROR` und die **IF-Statistik**. Sie nennt für jede `IF`-Anweisung Sektion und Zeile und zählt, wie oft die Bedingung wahr und wie oft sie falsch war; nie erreichte stehen mit 0 darin. Bricht die Wandlung ab, steht darin, was bis dahin gezählt wurde.

Einstellung der Konfiguration	Bedeutung
Logordner	leer: `temp` im Basisverzeichnis; ein relativer Pfad gilt ab dem Basisverzeichnis
Logs aufbewahren (Tage)	ältere `.LOG`-Dateien im Logordner werden beim Lauf gelöscht; 0 heißt unbegrenzt

Der Name folgt der Ausgabedatei. Mit `{ZAEHLER}` heißt das Log wie die geschriebene Ausgabe, etwa `INVOIC_000123.LOG`, und ein vorhandenes wird überschrieben. Sonst trägt es einen Zeitstempel, `INVOIC_20260913_101500.LOG`; wurde mit `{ZAEHLER}` nichts geschrieben, nach der Eingabedatei.

Löschen nur im Basisverzeichnis

Der Konverter löscht Dateien nur im Basisverzeichnis, dem gemeinsamen Installationsordner. Alte Logs außerhalb bleiben liegen, und der Lauf meldet das. Die einzige Ausnahme ist das Archiv, das meist neben der Eingabedatei liegt. Dort wird nur gelöscht, wenn in der Konfiguration „Archivdateien auch außerhalb des Basisverzeichnisses löschen" angehakt und bestätigt

ist. Das Erteilen und Entziehen der Freigabe und jede Datei, die außerhalb gelöscht wird, stehen mit Zeit und Windows-Benutzer im Löschprotokoll der Datenbank (Tabelle `Loeschung`). Nur die halbfertige `.tmp`-Datei eines gescheiterten Schreibens entfernt der Lauf überall, denn er hat sie selbst angelegt.

Gleichzeitige Läufe

Jeder Aufruf ist ein eigener Prozess, verschiedene Konfigurationen laufen nebeneinander. Wird eine Konfiguration gestartet, die gerade läuft, oder eine andere mit derselben Eingabedatei, wartet der zweite Lauf in der Warteschlange, bis der erste fertig ist, höchstens 10 Minuten. Danach wird er mit Rückgabewert 14 abgewiesen, ohne etwas zu wandeln. Das gilt auch zwischen Aufgabenplanung und Editor.

3.8 Vollständiges Beispiel

Diese Definition legt der Editor bei einem neuen Konverter an.

```
::GLOBAL-CONF
:  TARGET-DECIMAL-MARK = "."
:  TEMPLATE ~~SEG-END FIELD.SEG-ENDE.X1.TRIM = X"27"
:  TEMPLATE ~~FILL-10 FIELD.FILL.X10.TRIM

::TARGET-CONF
: RECORD.SEG-BGM
:    ~~FILL-10                          = "BGM+380+"
:    FIELD.BGM-NR.X35.TRIM
:    ~~SEG-END

::SOURCE-CONF
: RECORD-DELIMITER = X"0D0A"
: FIELD-DELIMITER  = ";"
: NO-RECORD-ID
: RECORD.SOURCE-RECORD
:    FIELD.SOURCE-DATA.X50

::RUN-CONF
: IF SOURCE-DATA = "BGM"
:    BGM-NR                             = GET_VALUE (2)
:    WRITE SEG-BGM
: END-IF
```

Die Eingabezeile `BGM;RG2026-12345` ergibt `BGM+380+RG2026-12345'`.