

eInvoiceShuttle.Cli – Benutzerhandbuch

Version	1.0.1.0
Datum	Oktober 2026
Autor	D. Riekert

Dateilinks funktionieren nicht in allen Browsern. In Chromium-basierten Browsern (Edge, Brave und Chrome) scheinen sie abhängig von den Sicherheitseinstellungen zu funktionieren.

Inhaltsverzeichnis

1. Überblick
2. Erweiterungen und Einschränkungen
3. Installation und Voraussetzungen
4. Erste Schritte
5. Übersicht
6. Aufruf der Anwendung
7. Konfigurationsdatei im Detail
8. Datenbankverbindung
9. Filter
10. Ausgabeoptionen und deren Abhängigkeiten
11. Fluid-Sprache
12. Rechnungsvorlagen (Templates)
13. PDF-Erstellung
14. ZUGFeRD-Erstellung und Einbettung
15. XRechnung-Validierung
16. Lokalisierung / Kultureinstellungen / Übersetzungen
17. GnuCash-spezifische Einstellungen
18. Adresseexport
19. Parallelverarbeitung
20. Protokollierung (Logging)
21. Fehlerbehandlung und Rückgabewerte
22. Vollständige Kommandozeilen-Referenz
23. Häufige Probleme (Troubleshooting)
24. Anhang: Verfügbare Platzhalter in Vorlagen

1. Überblick

Bitte beachten Sie den Haftungsausschluss. Sie finden ihn hier: [disclaimer_DE.PDF](#)

eInvoiceShuttle.Cli ist ein Kommandozeilen-Tool, das aktuell Rechnungsdaten aus einer **GnuCash**-Buchhaltungsdatenbank liest und daraus automatisiert folgende Ausgabeformate erzeugen kann (in der Zukunft sind auch andere Datenquellen geplant):

- **Erweiterte Rechnungsvorlage** (z. B. HTML oder LaTeX, mit eingesetzten Rechnungsdaten)
- **PDF-Dokument** (aus der erweiterten Vorlage erzeugt)
- **XRechnung** (strukturiertes XML-Dokument nach dem XRechnung-Standard)
- **ZUGFeRD-PDF** (PDF mit eingebettetem XRechnung-XML, hybrides Format)
- **Validierungsbericht** (HTML/XML-Bericht über die Konformität der erzeugten XRechnung)

Das Tool ist für den Batch-Betrieb ausgelegt: Es können in einem Durchlauf beliebig viele Rechnungen gefiltert und parallel verarbeitet werden.

2. Erweiterungen und Einschränkungen

2.1 Erweiterungen

Die erzeugten Rechnungen bieten folgende Möglichkeiten:

- **Mehrsprachigkeit:** Aktuell werden 23 europäische Sprachen mithilfe von KI unterstützt. Nicht alle Übersetzungen wurden überprüft; sie lassen sich jedoch leicht ändern und erweitern.
- **Hintergrundbilder:** Es können unterschiedliche Bilder pro Kunde, Sprache oder Rechnung verwendet werden. Dies wird in vielen Beispielen genutzt (nur TeX).
- **Logos:** Es können unterschiedliche Logos pro Kunde, Sprache oder Rechnung verwendet werden. Alle Beispiele verwenden unterschiedliche Logos; in HTML ist derzeit nur ein festes Logo vorgesehen.
- **Übersetzung von Rechnungspositionen:** Siehe Beispiel Österreich: „Tomaten → Paradeiser“.
- **Templateanpassungen:** z. B. Anpassung der Schriftart oder anderer Layoutelemente.
- **Mehrseitige Rechnungen:** Nur TeX. Die verwendete TeX-Tabelle ermöglicht einen automatischen Umbruch.
- **Zusätzliche Referenzen:** In digitalen Rechnungen gibt es sinnvolle Felder wie Projekt-Nr., Projektname, Vertrags-Nr. und Bestell-Nr., die für eine Rechnung definiert werden können (siehe Beispiele).
- **Rechnungsausgabe in HTML und TeX:** HTML-Rechnungen eignen sich für das Versenden in E-Mails, ggf. zusätzlich mit der XRechnung-XML als Anhang. Die PDF-Erzeugung über TeX ist einfacher und bietet auch Möglichkeiten wie automatischen Umbruch in der Rechnungspositionstabelle.

In GnuCash kann für jede Rechnungsposition eine Aktion erfasst werden.

Der dort eingegebene Wert kann auf verschiedene Weise genutzt werden, sogar mehrere gleichzeitig.

- **Einheit** (`unit:=xxx`) Die eingegebene Einheit wird in einer Datei gesucht. In digitalen Rechnungen wird die Einheit im entsprechenden BT-Feld eingetragen.
- **Text** (`text:=xxx`) Der Text erscheint als *Detail* auf der PDF-Rechnung. Es gibt keine Entsprechung in digitalen Rechnungen.
- **Link** (`link:=xxx`) Die Beschreibung enthält den angegebenen Text als Link. Es gibt keine Entsprechung in digitalen Rechnungen.
- **Bild-Verweis** (`image:=xxx`) Es kann der Name eines Bildes angegeben werden, das lokal gespeichert sein muss. Es erscheint hinter der Rechnungsposition. Keine Entsprechung in digitalen Rechnungen.

Es können mehrere Möglichkeiten gleichzeitig verwendet werden. Sie müssen aber beachten, dass für 'unit', 'text' und 'image' jeweils eine Spalte hinzukommt.

Da sich hier mehrzeilige Definitionen nur schwer eingeben lassen – anders als bei Rechnungs- oder Kundenbemerkungen –, können die Abschnitte durch ein Trennzeichen (Standard: ;) getrennt werden.

Das sieht dann beispielsweise so aus: `unit:=m;text:=hallo;link:=http://www.lvbprint.de;image:=myimage.png`

Alle Möglichkeiten werden in den Beispielen verwendet und dort erläutert.

2.2 Einschränkungen

Folgende Einschränkungen sind bekannt:

- Es werden nur Rabatte (als Betrag oder prozentual) auf Netto-Beträge unterstützt, also mit dem Zeichen $\leq$.

Rabatte mit '=' und '>' funktionieren zumindest bei digitalen Rechnungen nicht. Sie werden mit einer Fehlermeldung abgelehnt.

3. Installation und Voraussetzungen

Ausführliche Informationen zur Installation finden Sie im [Installationshandbuch](#).

Kurz zusammengefasst:

- Entpacken Sie das eInvoiceShuttle-Archiv in einem beliebigen Verzeichnis.
- Installieren und aktualisieren Sie **MiKTeX**.
- Wechseln Sie ins Unterverzeichnis `Examples` und starten Sie
 - `RunTexExamples.cmd`: ZUGFeRD- und XRechnungsdateien + Validierung
 - `RunTexExamplesNoValidation.cmd`: ZUGFeRD- und XRechnungsdateien aber keine Validierung (kein Java erforderlich)
 - `RunTexExamplesInvoiceOnly.cmd`: Erzeugt nur die PDF-Rechnungen ohne eingebettete digitale Rechnung

- Bei einem Fehler installieren Sie gegebenenfalls die .NET-Runtime 8.0 oder aktualisieren MiKTeX.

Hier sollten die erzeugten Rechnungen liegen: <C:/ProgramData/eInvoiceShuttle.Invoices/tex/>

4. Erste Schritte

4.1 Verwenden einer eigenen GnuCash-Datei

Am besten speichern Sie Ihre GnuCash-Datei im SQLite-Format, falls sie nicht bereits in diesem Format vorliegt. Arbeiten Sie zunächst am besten mit einer Kopie.

Das GnuCash-**XML**-Format wird nicht unterstützt.

- Speichern Sie Ihre SQLite-Datei z. B. als `C:\MyGnuCash\MyGnuCashFile.gnucash`.
- Ändern Sie die [Datenbankverbindung](#) in der Datei `bin\cli\config\appsettings.json` im Bereich `Connection`.
Anstelle von
`"File": "examples/GnuCash/SKR03_Music.gnucash"`, tragen Sie dann
`"File": "c:/MyGnuCash/MyGnuCashFile.gnucash"`, ein.
Sie können auch den Backslash `\` verwenden, müssen ihn aber doppeln (Escape in JSON).
- Durch das Ändern der Verbindung in der Konfigurationsdatei bleiben die Verweise in den Batchdateien weiterhin gültig.

Jetzt sind Sie so weit, die Tests mit Ihrer Datenbank auszuprobieren. Starten Sie wieder eine der `RunTexExamples*` Batchdateien.

Logausgaben finden Sie hier:

[C:/ProgramData/eInvoiceShuttle.Cli/eInvoiceShuttle.Cli.log](#)

Wenn alles gut geht, finden Sie die Rechnungen dann in:

[C:/ProgramData/eInvoiceShuttle.Invoices/tex/](#)

4.2 Überprüfen der Adressdaten

Insbesondere bei der Erstellung digitaler Rechnungen müssen die Adressdaten korrekt geparkt werden können, damit diese den entsprechenden BT-Feldern zugewiesen werden können.

Im `Examples`-Verzeichnis finden Sie die Batchdatei `RunAddressCheck.cmd`. Die Batchdatei erzeugt die Datei `C:\ProgramData\eInvoiceShuttle.Invoices\addresses.csv` mit allen gefundenen Adressen.

Öffnen Sie anschließend die Datei und überprüfen Sie, ob alles richtig erkannt wurde.

Insbesondere die Felder mit Großbuchstaben im Titel sollten korrekt sein.

Etwas ist nicht korrekt – was nun?

- Wenn kein Zielland bei der Kundenadresse angegeben oder dieses nicht erkannt wird, wird der Eintrag `"DefaultCountry": "Germany"`, aus der `appsettings.json` verwendet. Passen Sie den Eintrag entsprechend Ihrem Heimatland an.
- Im `config`-Verzeichnis finden Sie die Datei `countries.json`. Diese dient als Grundlage. Sie können ggf. Aliase hinzufügen wie bei Deutschland oder Österreich.
- Bei der Suche in der Datei werden die Felder `name`, `short`, `alpha2`, `alpha3` und `alias` überprüft, wobei `alias` eine durch Komma getrennte Liste von Namen ist.
- Adressen aus den Ländern `Großbritannien`, `Irland`, `Kanada`, `USA`, `Japan` sollten erkannt werden. Viele Adressen haben den gleichen Aufbau wie deutsche Adressen. Diese sollten dann auch funktionieren. Wenn Sie etwas vermissen und ein ISO-Adressformat benutzen, dann melden Sie sich.
- Notfalls passen Sie die Adressen in GnuCash an.

4.3 Geschäftsdaten eintragen

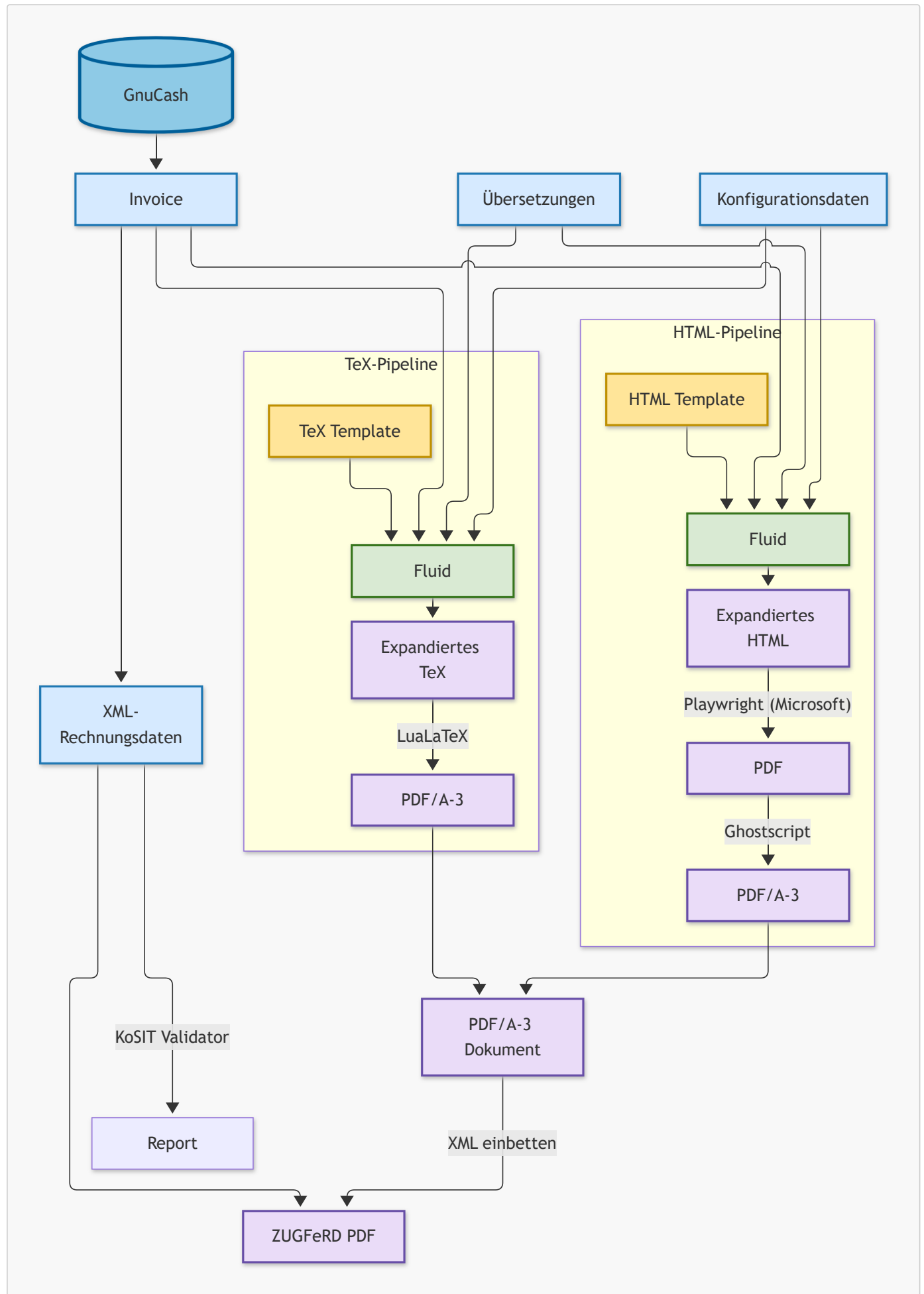
Die Rechnungen enthalten jetzt vermutlich teilweise Ihre Geschäftsdaten aus GnuCash und teilweise die Bankdaten, die in GnuCash nicht erfasst werden können.

Vergleichen Sie ggf. mit der Beispiel-GnuCash-Datei, welche Daten notwendig sind.

Bankdaten werden in der `appsettings.json` konfiguriert. Diese ist der Dreh- und Angelpunkt aller Konfigurationsdaten.

Sie können ggf. auch eine spezifische Settingsdatei erzeugen und diese mit der Option `-CF <Datei>` verwenden.

5. Übersicht



Die Abbildung zeigt den internen Ablauf und dient dem besseren Verständnis.

- Die Rechnungsdaten kommen aus einer Datenquelle. Zunächst wird nur `GnuCash` unterstützt.
- Diese werden in ein internes Format `invoice` überführt, das sehr *GnuCash*-lastig ist.
- Ein textbasiertes Template – aktuell `tex` und `html` (andere wie `rtf` wären denkbar) – wird mit Hilfe der sehr leistungsfähigen `Fluid`-Bibliothek expandiert. *Fluid* ist eine Implementierung der *Liquid*-Sprache, die ursprünglich von *Shopify* entwickelt wurde. Siehe [Fluid-Sprache](#)
- Die expandierten Templates werden jetzt auf unterschiedliche Weise nach PDF/A-3 konvertiert.
 - Bei *TeX* funktioniert das mit Hilfe von `LuaLaTeX`, das in den gängigen *TeX*-Installationen enthalten ist.
 - Bei *HTML* wird `Playwright` mit der Chromium-Engine verwendet, um ein PDF zu erzeugen. Dieses hat leider noch nicht das benötigte Format. Dazu wird zusätzlich noch `Ghostscript` benötigt. Also alles etwas umständlicher ...
- Für digitale Rechnungen wird eine entsprechende *XML*-Datei erzeugt. Unterstützt werden nur das `CII`-Format und die Profile `Comfort` und `XRechnung` (Da keine digitalen Rechnungen eingelesen werden, können die modernsten Einstellungen verwendet werden. Das reduziert den Testaufwand).
- Falls eine ZUGFeRD-Rechnung (in PDF/A-3 eingebettetes XML) benötigt wird, dann wird diese noch in das erzeugte *PDF/A-3* eingebettet. Behörden benötigen meist die *XML*-Datei und andere Empfänger die lesbarere *PDF*. Es können beide Formate erzeugt werden.
- Zu guter Letzt kann das Ergebnis noch mithilfe des mitgelieferten `KoSIT-Validators` überprüft werden. Das ist auf jeden Fall am Anfang sinnvoll. Für den Aufruf wird Java benötigt (siehe Installationshandbuch; Java Runtime Environment (JRE 11 oder höher)). Der Aufruf dauert relativ lange. Als Ergebnis wird ein *HTML*- und *XML*-Report pro überprüfter Rechnung erstellt. Es gibt auch genügend Online-Tools für die Überprüfung und ggf. Visualisierung der Rechnung. [PortInvoice](#) überprüft ebenfalls die PDF auf Gültigkeit und ermöglicht eine Visualisierung der Rechnung.
Falls Änderungen am XML notwendig sind, würde ich zu [PDF24](#) greifen. Dort können Sie die XML-Datei einladen, ggf. Änderungen vornehmen und wieder eine PDF/XML erstellen. Selbstverständlich dürfen Sie dann die vorher erzeugte **PDF-Rechnung nicht verwenden**.

6. Aufruf der Anwendung

- Ohne Argumente oder mit `-h`, `--help`, `-?` wird die integrierte Kurzhilfe angezeigt und das Programm beendet sich sofort mit Exit-Code 0.
- Alle Optionen können sowohl mit `-` als auch mit `/` als Präfix angegeben werden (z. B. `-DBType` oder `/DBType`).
- Boolesche Schalter werden mit `+` (aktivieren) oder `-` (deaktivieren) angegeben, z. B. `-OP+`.
- Mit `-CF <pfad>` kann eine alternative Konfigurationsdatei anstelle von `config/appsettings.json` verwendet werden.

7. Konfigurationsdatei im Detail

Hier eine tabellarische Übersicht über die wichtigsten Einstellungen aus der `config\appsettings.json`. Kommandozeilenparameter überschreiben Werte aus der Datei. Die wichtigsten Einstellungen folgen in den nächsten Kapiteln. Die meisten Einstellungen sind in der Datei dokumentiert.

Parameter können auch ohne Kürzel übergeben werden. Der Befehl muss mit `--` beginnen und ein Wert wird mit `=` anstelle eines Leerzeichens übergeben, also z. B.:

Folgende Einstellungen sind z.B. kompatibel:

- `--eInvoice.Connector.GnuCash:Culture=BG`
- `-CC BG`
- `/CC BG`

Einstellung	CLI	Beschreibung
eInvoice.Connector.GnuCash		
eInvoice.Connector.GnuCash:PaymentTerms		Wird als Standardwert für Zahlungsbedingungen verwendet. In der Regel werden die Zahlungsbedingungen bei der Rechnung ausgewählt (Tipp : In den Beispielen wird oft <i>Common</i> verwendet für einen übersetzten Standardtext)
eInvoice.Connector.GnuCash:CultureMode	-CM	Modus für die Auswahl der Sprache. Siehe Kapitel Lokalisierung

Einstellung	CLI	Beschreibung
eInvoice.Connector.GnuCash:Culture	-CC	Sprache bei CultureMode <i>Force</i> oder Fallback. Siehe Kapitel Lokalisierung
eInvoice.Connector.GnuCash:DefaultCountry		Wird verwendet, falls bei der Adresse kein Land angegeben wird
eInvoice.Connector.GnuCash:NotesDelimiter		Verwendeter Operator innerhalb von GnuCash für Einstellungen bei Bemerkungen; Escape durch Doppeln; z. B. :=:=
eInvoice.Connector.GnuCash:TranslateNotes		Erlaubt Fluid Syntax für Übersetzungen in Käufer- und Rechnungsbemerkungen
eInvoice.Connector.GnuCash:TranslateItems		Erlaubt Fluid Syntax für die Bemerkung von Rechnungspositionen
eInvoice.Connector.GnuCash:ActionInterpretation		Falls nicht überschrieben, wird diese Einstellung verwendet; in den Beispielen werden alle Möglichkeiten gezeigt
Referenzen		Über die 5 Einstellungen wie BuyerRefNotes können Namen für die Zuweisung von Referenzen festgelegt werden
Bankdaten		Kontodaten können nicht in GnuCash konfiguriert werden; dies geschieht hier
eInvoice		
eInvoice:ZUGFeRDVersion		Es wird nur die neueste Einstellung <i>Version23</i> unterstützt
eInvoice:ZUGFeRDFormat		Es wird nur <i>CII</i> unterstützt
eInvoice:ZUGFeRDProfile		<i>Comfort</i> / <i>XRechnung</i> , Standard <i>XRechnung</i>
eInvoice:BusinessProcess		Kann ggf. angepasst werden, wenn nötig
eInvoice:Fluid:Assignments:_extension	-Ext	Hier kann zwischen TeX und HTML Templates umgeschaltet werden; Das passiert z. B. in den Batchdateien
eInvoice:Fluid:Assignments:*		Alle anderen werden im TeX Layout verwendet und können überschrieben werden durch Übersetzungen, Käuferbemerkung, Rechnungsbemerkung; Siehe Beispiele
eInvoice:Fluid:curly_bracket_open_replacement		Definition eines besonderen Zeichens, das in Templates verwendet werden kann und später zu { umgewandelt wird
eInvoice:Fluid:curly_bracket_close_replacement		Das gleiche für die geschlossene Klammer; wird beides weiter unten bei Fluid-Sprache erklärt
eInvoiceShuttle.Core		
eInvoiceShuttle.Core:Connection:*		Wird im Kapitel Datenbankverbindung detailliert erläutert

Einstellung	CLI	Beschreibung
eInvoiceShuttle.Core:Validation:*		Einstellungen für den Aufruf des KoSIT Validators; Hier sollte keine Änderung notwendig werden.
eInvoiceShuttle.Core:Adressen:*		Gültige Adressen sind eine Grundlage für digitale Rechnungen. Es gibt eine Beispiel-Batchdatei, mit der alle Adressen in eine CSV-Datei exportiert und überprüft werden können
eInvoiceShuttle.Core:Filter:*		Einstellungen zum Filtern von Rechnungen; siehe Kapitel Filter
eInvoiceShuttle.Core:TemplateFiles	<i>nicht möglich</i>	Auswahl des zu verwendenden Templates; Hierzu gibt es ein eigenes Kapitel; nicht per Kommandozeile überschreibbar
eInvoiceShuttle.Core:PDFCreationCommands:*		Hier können gegebenenfalls andere Tools konfiguriert werden, um aus einem Template ein PDF/A-3 zu erzeugen. Sollte die Anwendung nicht im Pfad vorhanden sein, muss der Pfad hier eingetragen werden.
eInvoiceShuttle.Core:Ghostscript:*		Einstellungen zum Aufruf von Ghostscript
eInvoiceShuttle.Cli		
eInvoiceShuttle.Cli:InvoiceOutputFile		Name der Ausgabedatei für die Rechnung (<i>PDF</i>)
eInvoiceShuttle.Cli:XRechnungOutputFile		Name der Ausgabedatei für die XRechnungsdatei (<i>XML</i>)
eInvoiceShuttle.Cli:ValidationOutputPath		Ausgabepfad für die Validierungsberichte; der Name wird von den KoSIT-Tools automatisch generiert
eInvoiceShuttle.Cli:MaxDegreeOfParallelism		Maximale Anzahl gleichzeitiger Rechnungserstellungen, falls mehrere Rechnungen verarbeitet werden. Siehe: Parallelverarbeitung
eInvoiceShuttle.Cli:Create*		Siehe Kapitel Ausgabeoptionen

8. Datenbankverbindung

Die Verbindung zur GnuCash-Datenbank kann auf zwei Arten konfiguriert werden.

8.1 Direkte Angaben

Unterstützte Datenbanktypen: SQLite, MySQL, PostgreSQL.

Für MySQL/PostgreSQL werden zusätzlich benötigt: Host, Port, DatabaseName, User, Password. Fehlende Werte werden mit sinnvollen Standardwerten aufgefüllt:

Einstellung	Standardwert
Host	localhost
Port (MySQL)	3306
Port (PostgreSQL)	5432
DatabaseName	interner Standardwert (gncash)

CLI-Parameter	Konfigurationspfad	Beschreibung
-DBType	eInvoiceShuttle.Core:Connection:Type	Datenbanktyp
-DBFile	eInvoiceShuttle.Core:Connection:File	Dateipfad (nur SQLite)
-DBHost	eInvoiceShuttle.Core:Connection:Host	Host
-DBPort	eInvoiceShuttle.Core:Connection:Port	Port
-DBName	eInvoiceShuttle.Core:Connection:DatabaseName	Datenbankname
-DBUser	eInvoiceShuttle.Core:Connection:User	Benutzername
-DBPassword	eInvoiceShuttle.Core:Connection:Password	Passwort

8.2 Verschlüsselter Verbindungsschlüssel

Für den produktiven Einsatz empfiehlt es sich, die Zugangsdaten nicht im Klartext zu speichern. Beim Start mit vollständigen Verbindungsdaten und zusätzlichem Parameter `-DBCreateKey` gibt die Anwendung einen verschlüsselten Schlüssel auf der Konsole aus.

Dieser Wert kann anschließend dauerhaft übernommen werden.

Der entsprechende CLI-Parameter zur direkten Übergabe ist `-DBKey <schlüssel>`. Ist ein `EncryptedKey` gesetzt, hat dieser Vorrang vor allen anderen Verbindungseinstellungen.

Es wird nur eine einfache, computerbezogene Verschlüsselung verwendet.

9. Filter

Ohne mindestens einen gesetzten Filter werden keine Rechnungen verarbeitet – dies ist eine bewusste Sicherheitsmaßnahme, um versehentliche Massenverarbeitung zu vermeiden.

CLI-Parameter	Konfigurationspfad	Beschreibung
-FI	eInvoiceShuttle.Core:Filter:InvoiceId	Filter nach Rechnungsnummer
-FC	eInvoiceShuttle.Core:Filter:Customer	Filter nach Kundenname
-FD	eInvoiceShuttle.Core:Filter:Date	Filter nach Datum

Alle Filter unterstützen zwei Modi:

- **Exakte/Wildcard-Suche** (Standard): z. B. `-FD 202609*` (alle Rechnungen aus September 2026). Wie bei der Dateisuche werden `?` und `*` unterstützt.
- **Regulärer Ausdruck** mit Präfix `r:`: z. B. `-FI r:0*[23]$` (Rechnungsnummern, die auf 2 oder 3 enden).

Um wirklich alle Rechnungen zu verarbeiten, kann folgender Ausdruck verwendet werden: `-FI *`.

10. Ausgabeoptionen und deren Abhängigkeiten

CLI-Parameter	Bedeutung
-OT+ / -OT-	Expandierte Vorlage erstellen und behalten / wieder löschen
-OX+ / -OX-	XRechnung-Dokument erstellen / nicht erstellen
-OZ+ / -OZ-	ZUGFeRD (in PDF eingebettet) erstellen / nicht erstellen
-OP+ / -OP-	Original-PDF erstellen und behalten / wieder löschen
-OV+ / -OV-	Validierungsbericht erstellen / nicht erstellen

10.1 Implizite Abhängigkeiten

Da manche Ausgaben auf anderen aufbauen, werden fehlende Vorstufen automatisch aktiviert – jedoch nur temporär (die Zwischendatei wird nach Gebrauch wieder gelöscht, sofern sie nicht explizit mit `+` angefordert wurde):

- `-OZ+` aktiviert implizit `-OP` (falls nicht gesetzt), das erzeugte PDF wird nach der Einbettung gelöscht, außer `-OP+` wurde explizit angegeben.
- `-OP+` aktiviert implizit `-OT` (falls nicht gesetzt), die Vorlagen-Datei wird danach gelöscht, außer `-OT+` wurde explizit angegeben.
- `-OV+` aktiviert implizit `-OX` (falls nicht gesetzt), die XRechnung-Datei wird danach gelöscht, außer `-OX+` wurde explizit angegeben.

10.2 Ausgabepfade

Die Pfade für die einzelnen Ausgaben werden über Fluid/Liquid-Platzhalter dynamisch pro Rechnung expandiert:

Hinweis: `InvoiceOutputFile` benötigt keine Dateiendung – diese wird automatisch anhand der gefundenen Vorlage angehängt (z. B. `.html` oder `.tex`).

11. Fluid-Sprache

Fluid ist eine Implementierung der Liquid-Sprache, die ursprünglich von Shopify entwickelt wurde. Es ist eine leistungsfähige Methode, um Textdateien zu expandieren.

Die entsprechenden Stellen sind entweder durch `{% ... %}` für kompliziertere Sprachkonstrukte wie Schleifen, Bedingungen usw. eingeschlossen oder durch `{{ ... }}` für Variablen.

Einen guten Überblick über die Möglichkeiten bietet diese Seite: [Fluid-Sprache](#).

Hier sind zwei Beispiele aus dem TeX-Template, um einige Punkte zu erläutern:

```
\newcommand{\InvoiceNotes}{\span style="color:blue">{% for entry in Notes.List %}{{entry|escape_ext}}
</span>\\<span style="color:blue">{% endfor %}</span>}
\textbf{\span style="color:blue">{{_account_holder}}:</span>} & \SellerOwner & \textbf{\span
style="color:blue">{{_account_bank_name}}:</span>} & \SellerBank \\
```

TeX-Syntax ist schwarz

Fluid-Syntax ist blau

Hier die Besonderheiten:

- Zunächst fallen bei TeX die seltsamen Zeichen `{}` und `}` auf:
Da sowohl TeX als auch Liquid die geschweiften Klammern in besonderer Weise benutzen, müssen diese unterschieden werden. Die TeX-Klammern werden daher durch zwei Zeichen aus dem Unicode-Zeichensatz ersetzt. Die Definition können Sie gegebenenfalls in der `appsettings.json` durch andere Zeichen ersetzen. Nach dem Fluid-Aufruf werden alle Zeichen im Text ersetzt.
Eine falsche Verwendung kann sowohl bei Fluid als auch bei TeX zu Compilerfehlern führen, insbesondere bei der Kombination mit `%`, da auch das `%`-Zeichen eine besondere Bedeutung hat (Kommentare in TeX).
- `{% for ... %}` ist wie `{% if ... %}` und viele andere eine Syntax in Fluid für Sprachkonstrukte wie Bedingungen und Schleifen.
- `Notes.List` ist eine Variable, die aus der Rechnung kommt. Da `List` eine Aufzählung ist, kann man darüber iterieren. Eine Liste der Variablen finden Sie weiter hinten. Namen für Variablen sind hier großgeschrieben.
- `_account_bank_name` ist eine Variable, die aus der `appsettings.json` kommt. Es gibt noch weitere Möglichkeiten, Variablen einzuschleusen. Beispiele hierfür sind `_logo`, `_background` usw. Dazu später mehr.
- `escape_ext` ist ein Filter, der über das Programm definiert wurde. Es gibt viele in der Sprache enthaltene Filter ([Fluid-Sprache](#)) und einige durch das Programm definierte. Siehe Tabelle weiter unten.

Hier noch ein extremes Beispiel aus der Konfiguration für die Ausgabedatei:

Dies ist kein TeX, sondern reines Fluid.

In der aktuellen Konfigurationsdatei ist das Datum entfernt. Hier ist es als Beispiel noch enthalten.

```
"{{CommonApplicationData}}/eInvoiceShuttle.Invoices/{{_extension}}/{{Buyer.Company.Name | remove_blanks |
replace_invalid_filename_chars:\"_\"}}/{{InvoiceDate | format_date:\"yyyyMMdd\"}}_{{InvoiceId}}_{{CI}}"
```

- `CommonApplicationData` ist eine von einigen vordefinierten Variablen. Unter Windows wird diese zu `C:\ProgramData` expandiert.
- `_extension` ist eine Variable, die in der `appsettings.json` im Abschnitt `Fluid:Assignments` definiert wurde. Dadurch muss nur an einer Stelle die Templateendung definiert werden.

- `Buyer.Company.Name` ist wieder eine Variable aus der Rechnung.
- `remove_blanks`, `replace_invalid_filename_chars` sind vom `elInvoice`-Toolkit definierte Filter. Wie die Namen bereits sagen, entfernen sie Leerzeichen und ungültige Zeichen aus dem Firmennamen, damit dieser überhaupt als Dateiname verwendet werden kann.
- `format_date` ist ein von Fluid bereitgestellter Filter.
- `InvoiceId` ist die Rechnungsnummer aus der Rechnung.
- `CI` ist die verwendete Sprache, die ermittelt wurde. Dafür gibt es verschiedene Möglichkeiten.

12. Rechnungsvorlagen (Templates)

Die Rechnungsdaten werden mittels der **Fluid**-Template-Engine (Liquid-Syntax) in eine Vorlage eingesetzt. Die Suche nach der passenden Vorlage erfolgt über eine konfigurierbare Liste von Dateipfaden.

Ablauf der Vorlagensuche:

1. Die Liste wird der Reihe nach durchsucht.
2. Enthält ein Eintrag Platzhalter (`{}`), wird er zunächst mit den Rechnungsdaten expandiert (z. B. um eine kulturspezifische Datei wie `invoice_de-AT.html` zu finden).
3. Die erste existierende Datei wird verwendet.
4. Wird keine Datei gefunden, schlägt die Verarbeitung mit einer Fehlermeldung fehl, die alle durchsuchten Dateinamen auflistet.

Die gefundene Vorlagendatei wird vollständig mit den Rechnungsdaten expandiert (siehe Anhang für verfügbare Platzhalter) und anschließend als Datei gespeichert.

Die Dateiergung der Ausgabedatei richtet sich nach der Endung der Vorlagendatei (z. B. `.html`, `.tex`).

13. PDF-Erstellung

Die PDF-Erstellung erfolgt in zwei möglichen Varianten, abhängig von der Dateiergung der erweiterten Vorlage:

13.1 Über ein externes Shell-Kommando (empfohlen für LaTeX o. Ä.)

Für jede Dateiergung kann ein eigenes Kommando konfiguriert werden.

Eigenschaften des Shell-Kommandos:

Eigenschaft	Beschreibung
<code>ExePath</code>	Verzeichnis der ausführbaren Datei (optional)
<code>Cmd</code>	Name/Pfad der ausführbaren Datei
<code>Args</code>	Argumente; sie unterstützen Fluid-Platzhalter inkl. <code>{{ srcFile }}</code>
<code>ExitCodeOnSuccess</code>	Erwarteter Exit-Code bei Erfolg
<code>RunTwice</code>	Kommando zweimal ausführen (z. B. für LaTeX-Referenzen notwendig)
<code>RemoveLogFilesOnSuccess</code>	Temporäre Log-Dateien nach Erfolg löschen
<code>TempfileExtension</code>	Leerzeichen-getrennte Liste von Dateiergungen, die nach Erfolg gelöscht werden

Standardausgabe und Fehlerausgabe des Prozesses werden automatisch in `<RechnungsNr>.log` bzw. `<RechnungsNr>_err.log` im Arbeitsverzeichnis der Vorlage protokolliert.

13.2 Intern über Playwright (für HTML-Vorlagen)

Ist für die Dateiergung `.html` **kein** externes Kommando konfiguriert, wird intern **Playwright/Chromium** genutzt, um das HTML-Dokument direkt in ein PDF (Format A4, mit Rändern von 20 mm oben/unten und 10 mm links/rechts) zu rendern. Anschließend wird das PDF mittels **Ghostscript** in ein PDF/A-3-konformes Dokument konvertiert (notwendig für ZUGFeRD).

Hinweis: Für andere Dateiergungen als `.html` **muss** ein externes Kommando konfiguriert werden, da keine interne Unterstützung existiert.

14. ZUGFeRD-Erstellung und Einbettung

Die ZUGFeRD-Konfiguration bestimmt Version, Profil und Format des erzeugten XRechnung-XML, das anschließend in das PDF eingebettet wird:

CLI-Parameter	Beschreibung
-ZV	ZUGFeRD-Version (derzeit unterstützt: Version23)
-ZP	ZUGFeRD-Profil (Comfort, XRechnung)
-ZF	ZUGFeRD-Format (CII)

Ablauf bei aktiviertem -OZ+:

1. PDF wird erzeugt (falls nicht bereits vorhanden).
2. XRechnung-XML wird generiert.
3. Ist die Generierung erfolgreich, wird das XML in das PDF eingebettet (resultierendes hybrides ZUGFeRD-Dokument).
4. Schlägt die XRechnung-Erstellung fehl, wird die Einbettung übersprungen; ein zuvor nur für ZUGFeRD erzeugtes PDF wird gelöscht, sofern -OP+ nicht explizit gesetzt war.

15. XRechnung-Validierung

Die Validierung erfolgt über einen externen, Java-basierten Validator (z. B. den offiziellen KoSIT-Validator). Voraussetzungen:

- Eine funktionierende **Java-Laufzeitumgebung**. Der Pfad wird ermittelt über:
 1. Konfigurationswert `eInvoiceShuttle.Core.Validation:JavaHome`
 2. Systemumgebungsvariable `JAVA_HOME`
- Ein **Validator-Ordner** mit dem Validierungs-JAR. Der Pfad wird ermittelt über:
 1. Konfigurationswert `eInvoiceShuttle.Core.Validation:ValidatorPath`
 2. Automatische Suche nach einem Ordner namens `Validator` in den übergeordneten Verzeichnissen des Programmverzeichnisses

15.1 Konfiguration

Einstellung	Beschreibung
JavaHome	Pfad zur Java-Installation (überschreibt <code>JAVA_HOME</code>)
JavaOptions	Zusätzliche JVM-Optionen
ValidatorPath	Verzeichnis mit dem Validator-JAR
ValidatorJarMask	Suchmuster für die JAR-Datei (bei mehreren Treffern wird der alphabetisch letzte Treffer verwendet)
ValidatorExtOptions	Zusätzliche Kommandozeilenoptionen für den Validator
ValidatorResultsPath	Ausgabepfad für Zwischenergebnisse (Standard: gleich wie Berichtspfad)
ValidatorScenarioFile	Szenariendatei, die die anzuwendenden Prüfregeln definiert

15.2 Ablauf und Ergebnis

Der Validator wird als externer Java-Prozess gestartet. Die Standardausgabe wird nach dem Muster `Acceptable: <n> Rejected: <n>` durchsucht:

- **Rejected > 0** oder ein **Exit-Code > 0** führen zu einem Validierungsfehler.
- Bei Erfolg wird optional (mit `-OV+`) ein **HTML-Bericht** erzeugt (`<dateiname>-report.html`), zusätzlich zum XML-Bericht.

16. Lokalisierung / Kultureinstellungen / Übersetzungen

Für die Sprache der Rechnung gibt es in der `appsettings.json`-Datei zwei wichtige Einstellungen.

- Im `CultureMode Force` wird der in `Culture` konfigurierte Wert stets verwendet (Standard in der Konfigurationsdatei ist keine Sprache). In diesem Beispiel würden also alle Rechnungen in Französisch erstellt.
- Im `CultureMode Auto` gibt es eine bestimmte Reihenfolge der Suche. Zunächst wird die Kundenadresse verwendet und anhand des Landes die meistgenutzte Sprache verwendet. Das funktioniert nicht immer in mehrsprachigen Ländern wie der Schweiz oder

Belgien. Dafür können Sie bei den Bemerkungen zum Kunden eine andere Sprache wählen, und selbst diese kann pro Rechnung überschrieben werden.

Hier ein Beispiel aus der Rechnung 000003. Bei den Kundenbemerkungen gilt Entsprechendes.

```
Der Posten wird storniert / The invoice item will be canceled<br>
culture:=en<br>
action:=text<br>
_logo:=drunken_notes_small.png
```

Zu beachten sind hier die Zeichen `:=`. Diese haben eine besondere Bedeutung und können in der Konfigurationsdatei festgelegt werden. Sie werden aus den Bemerkungen entfernt. Somit bleibt nur noch die erste Zeile auf der Rechnung übrig.

- `culture` und `action` sind vordefiniert und erlauben das Überschreiben der Sprache im Auto-Modus und die Interpretation der Spalte Aktion. Bei den Bezeichnungen auf der linken Seite wird die Groß- und Kleinschreibung nicht beachtet.
- `_logo` überschreibt die in der `appsettings.json` definierte Variable. Sie können sich hier weitere überlegen, wenn Sie möchten. In den Beispielen werden diese für Layouteinstellungen verwendet (z. B. auch `_tex_font` usw.).

Die Werte können auch über die Kommandozeile überschrieben werden.

CLI-Parameter	Beschreibung
<code>-CC</code>	Kultur (Culture) im „Force“-Modus, z. B. <code>de-AT</code>
<code>-CM</code>	Kultur-Modus: <code>Auto</code> (aus Rechnungsdaten/Notizen abgeleitet) oder <code>Force</code> (erzwungen über <code>-CC</code>)
<code>-Ext</code>	Dateiendung für generierte Vorlagen, z. B. <code>tex</code> oder <code>html</code> (Konfigurationspfad: <code>eInvoice:Fluid:Assignments:_extension</code>)

Die tatsächlich verwendete Kultur (CI) einer Rechnung wird entweder aus den GnuCash-Notizen/Adressdaten abgeleitet (Modus `Auto`) oder fest über `-CC` vorgegeben (Modus `Force`). Sie beeinflusst sowohl die Auswahl der Rechnungsvorlage (siehe Kapitel 12) als auch die Formatierung von Zahlen und Datumswerten im generierten Dokument.

Nachdem nun geklärt ist, wie die Sprache festgelegt wird, folgt eine Übersicht darüber, an welchen Stellen Übersetzungen angewendet werden.

• Übersetzungstabellen

Alle Dateien mit dem Muster `*Translations.json` im Verzeichnis `config` werden als Übersetzungstabellen interpretiert. Folgende Dateien sind standardmäßig enthalten (alle Keys fangen mit einem Unterstrich und einem Kleinbuchstaben an, um die Texte von anderen z. B. Rechnungsinhalten im Template unterscheiden zu können):

- `InvoiceTranslations.json` für die Übersetzung der Überschriften im Template
- `OtherTranslations.json` Beispiele für die Übersetzung von Rechnungspositionen und Bemerkungen

• Rechnungspositionen

Sie können theoretisch die Beschreibungen der Rechnungspositionen übersetzen. Beispiele dafür gibt es in der deutschen und der österreichischen Rechnung, bei denen `Tomaten` unterschiedlich übersetzt werden. Falls es hierbei Probleme geben sollte, können Sie die Option in der `appsettings.json` deaktivieren (`TranslateItems`).

• Rechnungsbemerkungen

Bemerkungen auf Rechnungen wollen Sie womöglich auch mehrsprachig haben.

Beispiel: Example from invoice notes: `'{{_invoice_notes_example | escape_ext}}'`.

Auch diese Option kann ggf. deaktiviert werden (`TranslateNotes`).

• Zahlungsbedingungen

Die Beispielrechnungen enthalten eine Übersetzung bei den Zahlungsbedingungen mit Namen `Common` mit dem

Inhalt: `{{_pay_until_duedate | escape_ext}}`.

Den Text und die Übersetzungen finden Sie in der `InvoiceTranslations.json` Datei.

Bei allen Übersetzungen gilt, dass diese in `{{. }}` eingeschlossen sein müssen und falls Sie besondere Zeichen enthalten, dann sollte auch der Filter `escape_ext` Anwendung finden, der Sonderzeichen für TeX und HTML durch die entsprechenden Escape-Zeichen ersetzt.

17. GnuCash-spezifische Einstellungen

CLI-Parameter	Konfigurationspfad	Beschreibung
-Action	eInvoice.Connector.GnuCash:ActionInterpretation	Interpretation der GnuCash-„Action“-Spalte pro Rechnungsposition

Mögliche Werte für `ActionInterpretation`:

Wert	Bedeutung
None	Keine besondere Interpretation (Standard)
Unit	Der Inhalt der Action-Spalte wird als Maßeinheit interpretiert (z. B. „Stk“, „h“)
Image	Der Inhalt wird als Bildreferenz interpretiert (z. B. Produktbild in der Vorlage)
Text	Der Inhalt wird als zusätzlicher Freitext zur Position interpretiert
Link	Die Beschreibung erhält den Inhalt als Link

Die daraus resultierenden Eigenschaften `ActionIsUnit`, `ActionIsImage`, `ActionIsText`, `ActionIsLink` einer Rechnung stehen in der Vorlage zur Verfügung (siehe Anhang).

18. Adressexport

Zusätzlich zur Rechnungserstellung kann eine Liste aller im System bekannten Adressen exportiert werden (z. B. für den Abgleich mit anderen Systemen):

Einstellung	Beschreibung
Dump	Adressexport aktivieren
OutputFile	Zieldatei (unterstützt Fluid-Platzhalter und Pfadangaben mit <code>/</code>)
Delimiter	CSV-Trennzeichen (Standard: <code>;</code>)
Encoding.Name	Zeichenkodierung: <code>utf-8</code> , <code>utf-16</code> , <code>utf-32</code> oder ein beliebiger von .NET unterstützter Codepage-Name (z. B. <code>windows-1252</code>)
Encoding.WithBom	Byte-Order-Mark einfügen (nur für UTF-Kodierungen relevant)
Encoding.BigEndian	Big-Endian-Bytereihenfolge (nur für UTF-16/32 relevant)

Der Adressexport erfolgt unabhängig von den Filtereinstellungen und wird vor der eigentlichen Rechnungsverarbeitung durchgeführt.

19. Parallelverarbeitung

Die Verarbeitung mehrerer Rechnungen erfolgt parallel, um große Stückzahlen effizient zu verarbeiten. Der Grad der Parallelität wird über den Konfigurationswert `eInvoiceShuttle.Cli:MaxDegreeOfParallelism` gesteuert (Standard: `1`):

- Der Wert wird automatisch auf mindestens `1` und höchstens die Anzahl der verfügbaren Prozessorkerne begrenzt.
- Die Rechnungen werden vorab vollständig aus der Datenbank geladen (um parallele Datenbankzugriffe zu vermeiden), bevor die parallele Verarbeitung beginnt.
- Tritt bei einer einzelnen Rechnung ein Fehler auf, wird dieser protokolliert; die Verarbeitung der übrigen Rechnungen wird fortgesetzt.

20. Protokollierung (Logging)

Die Anwendung nutzt **log4net**, konfiguriert über `config/log4net.config`. Protokolldateien werden standardmäßig unter folgendem Pfad abgelegt:

```
%ALLUSERSPROFILE%\<Anwendungsname>\<Anwendungsname>.log
```

Zusätzlich existieren dedizierte Logger für bestimmte Programmteile:

Logger-Name	Zweck
CreatePDF	Protokolliert die PDF-Erstellung
Validation	Protokolliert den Validierungsvorgang

Die Feineinstellung der Log-Level (Debug, Info, Warn, Error) erfolgt vollständig über `log4net.config` und kann unabhängig je Logger konfiguriert werden.

21. Fehlerbehandlung und Rückgabewerte

- Rechnungen mit Validierungsfehlern (`invoice.Valid == false`) werden übersprungen; die XRechnung-Erstellung findet für diese Rechnung nicht statt.
- Schlägt die XRechnung-Erstellung fehl, wird eine begonnene ZUGFeRD-Einbettung abgebrochen.
- Ein PDF, das nur als Zwischenschritt für ZUGFeRD erzeugt wurde, wird bei Fehlschlag der XRechnung-Erstellung automatisch gelöscht (sofern `-OP+` nicht explizit gesetzt wurde).
- Temporäre Zwischendateien (erweiterte Vorlage, XRechnung-XML) werden nach Gebrauch automatisch entfernt, sofern sie nicht explizit über `-OT+` bzw. `-OX+` dauerhaft angefordert wurden.
- Exit-Code:**
 - `0` – vollständiger Erfolg
 - `1` – mindestens ein Fehler ist während der Verarbeitung aufgetreten (z. B. fehlgeschlagene Validierung, Dateizugriffsfehler, nicht gefundene Vorlage)

22. Vollständige Kommandozeilen-Referenz

Parameter	Beschreibung
<code>-CF <pfad></code>	Konfigurationsdatei (JSON)
<code>-DBKey <schlüssel></code>	Verschlüsselten Verbindungsschlüssel verwenden (überschreibt andere DB-Einstellungen)
<code>-DBType <typ></code>	Datenbanktyp: <code>SQLite</code> , <code>MySQL</code> , <code>PostgreSQL</code>
<code>-DBFile <pfad></code>	Datenbank-Dateipfad (nur SQLite)
<code>-DBHost <host></code>	Datenbank-Host (Standard: <code>localhost</code>)
<code>-DBPort <port></code>	Datenbank-Port (Standard: <code>3306/5432</code>)
<code>-DBName <name></code>	Datenbankname
<code>-DBUser <user></code>	Datenbank-Benutzer
<code>-DBPassword <pwd></code>	Datenbank-Passwort
<code>-DBCreateKey</code>	Verschlüsselten Verbindungsschlüssel generieren (für <code>appsettings.json</code>)
<code>-FI <id></code>	Filter nach Rechnungsnummer (<code>r:</code> für reguläre Ausdrücke)
<code>-FC <kunde></code>	Filter nach Kundenname (<code>r:</code> für reguläre Ausdrücke)
<code>-FD <datum></code>	Filter nach Datum, z. B. <code>2024-01-01</code> (<code>r:</code> für reguläre Ausdrücke)
<code>-OT+ / -OT-</code>	Expandierte Vorlage erstellen und behalten / wieder löschen
<code>-OX+ / -OX-</code>	XRechnung-Dokument erstellen / nicht erstellen
<code>-OZ+ / -OZ-</code>	ZUGFeRD (eingebettet in PDF) erstellen / nicht erstellen
<code>-OP+ / -OP-</code>	Original-PDF erstellen und behalten / wieder löschen
<code>-OV+ / -OV-</code>	Validierungsbericht erstellen / nicht erstellen
<code>-CC <kultur></code>	Kultur im „Force“-Modus
<code>-CM <modus></code>	Kultur-Modus: <code>Auto</code> oder <code>Force</code>
<code>-Ext <endung></code>	Dateiendung für generierte Vorlagen, z. B. <code>tex</code> oder <code>html</code>
<code>-ZV <version></code>	ZUGFeRD-Version (<code>Version23</code>)

Parameter	Beschreibung
-ZP <profil>	ZUGFeRD-Profil (Comfort, XRechnung)
-ZF <format>	ZUGFeRD-Format (CII)
-Action <aktion>	GnuCash-Aktionsinterpretation (None, Unit, Image, Text, Link)
-h, --help, -?	Hilfe anzeigen

23. Häufige Probleme (Troubleshooting)

Problem	Mögliche Ursache	Lösung
„No invoices found matching the specified filter criteria“	Kein passender Filter oder falsche Filterwerte	Filterwerte prüfen, ggf. mit <code>-FI *</code> testen
„No default database connection configured“	Weder <code>EncryptedKey</code> noch <code>Type</code> in der Konfiguration gesetzt	Verbindungsdaten in <code>appsettings.json</code> oder per CLI ergänzen
„JAVA_HOME not configured. Validation cannot continue“	Java nicht installiert oder Umgebungsvariable fehlt	Java installieren und <code>JAVA_HOME</code> setzen, oder <code>-OV-</code> verwenden
„Validator folder not found“	Kein Ordner <code>Validator</code> im Programmverzeichnis oder übergeordneten Verzeichnissen	<code>ValidatorPath</code> explizit konfigurieren
„No template file found for invoice ... with Culture ...“	Keine passende Vorlage für die ermittelte Kultur vorhanden	Vorlage für die entsprechende Kultur anlegen oder <code>TemplateFiles</code> um einen Fallback-Eintrag ohne Kulturplatzhalter ergänzen
„Failed to get pdf creation command. No command defined and no internal support for extension“	Vorlage hat eine Endung ungleich <code>.html</code> , aber kein <code>PDFCreationCommands</code> -Eintrag konfiguriert	Passendes Shell-Kommando in <code>PDFCreationCommands</code> hinterlegen
Validierung schlägt trotz augenscheinlich korrekter XRechnung fehl	Falsches oder veraltetes <code>ValidatorScenarioFile</code>	Aktuelle Szenariendatei des Validators verwenden
Rechnung wird trotz Filter nicht verarbeitet	<code>invoice.Valid == false</code> (Datenfehler in GnuCash)	Fehlermeldung im Log prüfen (<code>invoice.ErrorMessage</code>) und Buchungsdaten korrigieren

24. Anhang: Verfügbare Platzhalter in Vorlagen

Die folgenden Eigenschaften aus einer Rechnung stehen in Fluid/Liquid-Vorlagen zur Verfügung. Zugriff erfolgt in Liquid-Syntax, z. B. `{{ InvoiceId }}`. Die meisten davon werden in den Templates verwendet.

24.1 Allgemeine Rechnungsdaten

Eigenschaft	Typ	Beschreibung
Gültigkeit		
Valid	bool	Gibt an, ob die Rechnung gültig und vollständig verarbeitet werden kann.
ErrorMsg	string	Fehlermeldung, wenn die Rechnung ungültig ist.
Bestimmte Daten vorhanden		
HasReference	bool	Gibt an, ob mindestens eine Rechnungsreferenz vorhanden ist.
HasVat	bool	Gibt an, ob Mehrwertsteuerdaten vorhanden sind.
HasMultipleVat	bool	Gibt an, ob mehrere unterschiedliche Mehrwertsteuersätze verwendet werden.
HasDeliveryAddress	bool	Gibt an, ob eine Lieferadresse vorhanden ist.
HasItemsDiscount	bool	Gibt an, ob mindestens eine Rechnungsposition einen Rabatt enthält.
HasTaxCodes	bool	Gibt an, ob Steuerkategorien oder Steuerkennzeichen vorhanden sind.
NeedsTaxCodesColumn	bool	Gibt an, ob die Ausgabe eine zusätzliche Spalte für Steuerkennzeichen benötigt.
Basisdaten		
InvoiceId	string	Eindeutige Rechnungsnummer oder Rechnungskennung.
InvoiceType	string	Rechnungsart als übersetzte Zeichenkette.
CI	CultureInfo	Kulturinformationen für Sprache, Datums-, Zahlen- und Währungsformatierung.
Symbol	string	Währungssymbol, beispielsweise €.
Currency	string	Währungscode, beispielsweise EUR.
Notes	NotesParser	Geparste Notizen der Rechnung. Enthält Freitext und Schlüssel-Wert-Angaben.
DueDate	DateTime?	Zahlungsziel beziehungsweise Fälligkeitsdatum.
InvoiceDate	DateTime?	Rechnungsdatum.
DeliveryDate	DateTime?	Liefer- oder Leistungsdatum.
TermsDescription	string	Beschreibung der Zahlungsbedingungen.
Payed	decimal?	Bereits bezahlter Betrag.
Actions	Dictionary<string, bool>	Aktionstypen mit Daten in der Rechnung.
Referenzen		
ProjectId	string	Projektnummer oder Projektkennung.
ProjectName	string	Name des Projekts.
Contract	string	Vertragsreferenz.
PurchaseOrder	string	Bestellnummer oder Bestellreferenz.

Eigenschaft	Typ	Beschreibung
SalesOrder	string	Verkaufsauftragsnummer.
Unterobjekte		
Seller	SellerData	Daten des Verkäufers beziehungsweise Rechnungsausstellers.
Buyer	CustomerData	Daten des Käufers beziehungsweise Rechnungsempfängers.
Items	List<IInvoiceEntry>	Liste der Rechnungspositionen.
Summen und weitere Daten		
IsCreditNote	bool	Gibt an, ob die Rechnung eine Gutschrift beziehungsweise negative Rechnung ist.
Terms	BillTerms	Zahlungsbedingungen der Rechnung. Die auf der Rechnung definierten Zahlungsbedingungen haben Vorrang vor den Kundenzahlungsbedingungen.
AllPaid	bool	Gibt an, ob der zu zahlende Betrag vollständig ausgeglichen ist.
ItemsCount	int	Anzahl der Rechnungspositionen.
TermsDueDays	int	Anzahl der Tage bis zur Fälligkeit. Wird Terms nicht angegeben, wird 0 zurückgegeben.
SumItems	List<InvoiceEntry>	Liste der kumulierten Rechnungspositionen. Jede Position enthält die bis zu dieser Position aufgelaufenen Summen.
TaxEntries	List<InvoiceTaxEntry>	Liste der nach Steuersatz zusammengefassten Steuerdaten.
TaxCodes	List<TaxData>	Liste aller in den Rechnungspositionen verwendeten Steuerdaten. Für steuerfreie oder steuerfreie Positionen kann ein eigener Eintrag enthalten sein.
NetAmountSum	decimal	Summe aller Nettobeträge der Rechnung vor Steuern.
AmountToPay	decimal?	Betrag, der nach Abzug bereits geleisteter Zahlungen noch zu bezahlen ist.
TaxSum	decimal	Gesamtsumme aller Steuerbeträge der Rechnung.
RoundingDiff	decimal	Rundungsdifferenz zwischen Einzelpositionssummen und der Gesamtsumme.
GrossAmountSum	decimal	Bruttosumme der Rechnung einschließlich Steuern.

24.2 SellerData / CustomerData / Adressen

Member	Datentyp	Beschreibung
SellerData		
Company	CompanyData	Unternehmensdaten des Verkäufers.
Address	AddressData	Adresse des Verkäufers.
Account	AccountData	Bank- und Zahlungsdaten des Verkäufers.
CustomerData		
BuyerReference	string	Käuferreferenz gemäß BT-10, beispielsweise eine Leitweg-ID.
Terms	BillTerms	Zahlungsbedingungen des Kunden.
Company	CompanyData	Unternehmensdaten des Kunden.
Address	AddressData	Rechnungsadresse des Kunden.
ShipAddress	AddressData	Lieferadresse des Kunden.

Member	Datentyp	Beschreibung
Notes	NotesParser	Geparste Kundennotizen mit zusätzlichen Konfigurationswerten.
CompanyData		
Name	string	Rechtlicher oder allgemeiner Name des Unternehmens.
TradingName	string	Handelsname beziehungsweise abweichender Firmenname.
Vat	string	Umsatzsteuer-Identifikationsnummer.
Tin	string	Steueridentifikationsnummer, insbesondere für B2C-Rechnungen.
Url	string	Internetadresse des Unternehmens.
TaxRegistration	string	Weitere Steuerregistrierungsinformationen.
LegalInformation	string	Rechtliche Unternehmensinformationen, beispielsweise Handelsregisterangaben.
ElectronicAddress	string	Elektronische Adresse des Unternehmens, beispielsweise eine E-Rechnungsadresse.
AddressData		
OrgAddrLines	List<string>	Ursprüngliche, nicht weiter verarbeitete Adresszeilen.
AddrLines	List<string>	Bereinigte Adresszeilen ohne nachfolgende Leerzeilen.
HasAddress	bool	Gibt an, ob verwertbare Adress- oder Kontaktdaten vorhanden sind (insbesondere bei Lieferadresse sinnvoll).
Contact	string	Ansprechpartner.
Phone	string	Telefonnummer.
Fax	string	Faxnummer.
Mail	string	E-Mail-Adresse.
PostCode	string	Postleitzahl.
City	string	Ort oder Stadt.
Street	string	Straße und Hausnummer.
Country	string	Ländername.
SubCountry	string	Bundesland, Region oder Teilstaat.
CountryCode	string	Länderkennzeichen, beispielsweise DE.
Culture	string	Zugeordneter Kulturcode, beispielsweise de-DE.
AccountData		
Type	string	Zahlungsart beziehungsweise Payment Means.
Owner	string	Kontoinhaber.
BIC	string	BIC beziehungsweise SWIFT-Code der Bank.
IBAN	string	Internationale Bankkontonummer.
Bank	string	Name der Bank.
BillTerms		
Name	string	Name der Zahlungsbedingung.
Description	string	Beschreibung der Zahlungsbedingung.
DueDays	int	Anzahl der Tage bis zur Fälligkeit.

24.3 Steuerdaten

Member	Datentyp	Beschreibung
TaxData		
TaxCategory	string	Steuerkategorie, beispielsweise S, K, G oder AE.
VatexCode	string	VAT-EX-Code für steuerbefreite oder nicht steuerbare Umsätze.
TaxReason	string	Übersetzte Begründung für die Steuerbehandlung.
TaxTableEntry		
Amount	decimal	Steuerbetrag oder Steuerfaktor. Bei prozentualen Steuern entspricht der Wert dem Steuersatz.
TaxType	TaxTypes	Art der Steuerberechnung, beispielsweise Prozent oder fester Betrag.
InvoiceTaxEntry		
TaxId	string	Steuerkennzeichen beziehungsweise Steuerkategorie, beispielsweise S, K, G oder AE.
TaxRate	decimal	Angewandeter Steuersatz in Prozent. Bei steuerfreien oder nicht steuerbaren Umsätzen ist der Wert normalerweise 0.
TaxSum	decimal	Gesamter Steuerbetrag für das Steuerkennzeichen und den Steuersatz.
Netto	decimal	Summe der Nettobeträge aller Rechnungspositionen mit diesem Steuerkennzeichen und Steuersatz.

24.4 Positionen und Summen

Eigenschaft	Typ	Beschreibung
Items	Liste	Rechnungspositionen
ItemsCount	int	Anzahl der Positionen
SumItems	Liste	Zwischensummen-Positionen
TaxEntries	Liste	Steuerbeträge je Steuersatz
TaxCodes	Liste	Verwendete Steuercodes
NetAmountSum	decimal	Nettosumme
TaxSum	decimal	Summe der Steuerbeträge
RoundingDiff	decimal	Rundungsdifferenz
GrossAmountSum	decimal	Bruttosumme
HasVat	bool	Mindestens ein Steuersatz ungleich 0 %
HasMultipleVat	bool	Mehrere unterschiedliche Steuersätze vorhanden
HasTaxCodes	bool	Mindestens ein Steuercode vorhanden
NeedsTaxCodesColumn	bool	Mehr als ein Steuercode in den Positionen (Anzeige einer Steuercode-Spalte sinnvoll)
HasItemsDiscount	bool	Mindestens eine Position mit Rabatt
HasDeliveryAddress	bool	Lieferadresse vorhanden

24.5 Benutzerdefinierte Filter

Die folgenden benutzerdefinierten Filter stehen in Fluid-Vorlagen zur Verfügung. Sie werden mit dem Pipe-Zeichen (|) auf einen Wert angewendet, z. B. {{ Dateiname | to_slash }}.

Filter	Beschreibung	Beispiel / Hinweise
replace_invalid_filename_chars	Ersetzt Zeichen, die im Dateinamen ungültig sind.	Optional kann als erstes Argument die Ersatzzeichenfolge angegeben werden, z. B. `{{ Invoiceld

Filter	Beschreibung	Beispiel / Hinweise
<code>escape_ext</code>	Passt einen Wert durch Escaping passend zur Dateierdung der Vorlage.	<code>.tex</code> : TeX-Escaping; <code>.ps</code> : PostScript-Escaping; <code>.html</code> : HTML-Encoding. Bei anderen Endungen bleibt der Wert unverändert.
<code>to_slash</code>	Ersetzt Backslashes (\\) durch Schrägstriche (/).	Für einheitliche Pfadangaben.
<code>remove_blanks</code>	Entfernt sämtliche Whitespace-Zeichen aus einem Wert.	Entfernt unter anderem Leerzeichen, Tabs und Zeilenumbrüche.
<code>find_file</code>	Sucht eine Datei über die Konfigurationsdatei-Suche und gibt den gefundenen Pfad zurück.	Wenn keine Datei gefunden wird, wird <code>File <Name> not found</code> zurückgegeben.
<code>to_backslash</code>	Ersetzt Schrägstriche (/) durch Backslashes (\\).	Für Windows-Pfade.
<code>to_pathchar</code>	Ersetzt beide Schrägstrichvarianten durch das plattformabhängige Pfadtrennzeichen.	Unter Windows <code>\\</code> , unter Unix-artigen Systemen <code>/</code> .

24.6 Vordefinierte Variablen

Zusätzlich zu den Eigenschaften des `Invoice`-Objekts stellt `AddInternalConstants` die folgenden Variablen bereit. Sie können wie andere Fluid-Variablen verwendet werden, z. B. `{{ BasePath }}`.

Variable	Beschreibung
<code>CommonApplicationData</code>	Pfad zum gemeinsamen Anwendungsdatenverzeichnis des Systems.
<code>ApplicationData</code>	Pfad zum Anwendungsdatenverzeichnis des aktuellen Benutzers.
<code>LocalApplicationData</code>	Pfad zum lokalen Anwendungsdatenverzeichnis des aktuellen Benutzers.
<code>Personal</code>	Pfad zum persönlichen Dokumentenverzeichnis des aktuellen Benutzers.
<code>CommonDesktopDirectory</code>	Pfad zum gemeinsamen Desktopverzeichnis.
<code>CommonDocuments</code>	Pfad zum gemeinsamen Dokumentenverzeichnis.
<code>BasePath</code>	Basisverzeichnis der Anwendung.
<code>ImagesPath</code>	Pfad zum Unterverzeichnis <code>Images</code> im Anwendungsverzeichnis.
<code>ConfigPath</code>	Pfad zum Unterverzeichnis <code>config</code> im Anwendungsverzeichnis.
<code>TemplatesPath</code>	Pfad zum Unterverzeichnis <code>templates</code> im Anwendungsverzeichnis.
<code>NOW</code>	Aktuelles Datum und aktuelle Uhrzeit (<code>DateTime.Now</code>).
<code>CS</code>	Währungssymbol der Rechnung; leer, wenn keine Rechnung vorliegt.
<code>CI</code>	Name der verwendeten Kultur (<code>CultureInfo.Name</code>).
<code>DirSep</code>	Plattformabhängiges Pfadtrennzeichen.
<code>CL</code>	Zweistelliger ISO-Sprachcode der Kultur, z. B. <code>de</code> .
<code>InvoiceDateCI</code>	Rechnungsdatum, formatiert gemäß der verwendeten Kultur. Wenn kein Rechnungsdatum vorhanden ist, wird das aktuelle Datum verwendet.