

eInvoiceBridge.Cli – User Manual

Version	1.0.0.0
Date	October 2026
Author	D. Riekert

File links do not work in all browsers. In Chromium-based browsers (Edge, Brave, Chrome), they appear to work, probably depending on the security settings.

Table of Contents

1. Overview
2. Features and Limitations
3. Installation and Requirements
4. Getting Started
5. Overview
6. Calling the Application
7. Configuration File in Detail
8. Database Connection
9. Filters
10. Output Options and Their Dependencies
11. Fluid Language
12. Invoice Templates
13. PDF Creation
14. ZUGFeRD Creation and Embedding
15. XRechnung Validation
16. Localization / Culture Settings / Translations
17. GnuCash-Specific Settings
18. Address Export
19. Parallel Processing
20. Logging
21. Error Handling and Return Values
22. Complete Command-Line Reference
23. Common Problems (Troubleshooting)
24. Appendix: Available Template Placeholders

1. Overview

Please note the disclaimer. You can find it here: [disclaimer_en.PDF](#)

eInvoiceBridge.Cli is a command-line tool that currently reads invoice data from a **GnuCash** accounting database and can automatically generate the following output formats (other data sources are planned for the future):

- **Expanded invoice template** (e.g. HTML or LaTeX, with invoice data inserted)
- **PDF document** (generated from the expanded template)
- **XRechnung** (structured XML document according to the XRechnung standard)
- **ZUGFeRD PDF** (PDF with embedded XRechnung XML, hybrid format)
- **Validation report** (HTML/XML report on the conformity of the generated XRechnung)

The tool is designed for batch operation: any number of invoices can be filtered and processed in parallel in a single run.

Parallel processing does not appear to work reliably with HTML invoices. It is therefore disabled in the examples (batch files in the Examples folder) using the option (`--eInvoiceBridge.Cli:MaxDegreeOfParallelism=1`).

2. Features and Limitations

2.1 Features

The generated invoices offer the following options:

- **Multilingual support:** Currently, 23 European languages are supported using AI. Not all translations have been reviewed; however, they can easily be changed and extended.
- **Background images:** Different images can be used per customer, language, or invoice. This is used in many examples (TeX only).
- **Logos:** Different logos can be used per customer, language, or invoice. All examples use different logos; in HTML, only one fixed logo is currently provided.
- **Translation of invoice items:** See the Austria example: "Tomaten → Paradeiser".
- **Template customization:** For example, changing the font or other layout elements.
- **Multi-page invoices:** TeX only. The TeX table used supports automatic page breaks.
- **Additional references:** Digital invoices contain useful fields such as project number, project name, contract number, and purchase order number, which can be defined for an invoice (see examples).
- **Invoice output in HTML and TeX:** HTML invoices are suitable for sending by email, possibly with the XRechnung XML attached. PDF creation via TeX is simpler and also supports features such as automatic page breaks in the invoice-item table.

In GnuCash, an *Action* can be entered for each invoice item.

The value entered there can be used in different ways, including several options at the same time:

- **Unit** (`unit:=xxx`): The specified unit is looked up in a file. In digital invoices, the unit is entered in the corresponding BT field.
- **Text** (`text:=xxx`): The text appears as *Detail* on the PDF invoice. There is no equivalent in digital invoices.
- **Link** (`link:=xxx`): The description contains the specified text as a link. There is no equivalent in digital invoices.
- **Image reference** (`image:=xxx`): You can specify the name of an image file, which must be stored locally. The image appears behind the invoice item. There is no equivalent in digital invoices.

Several options can be used at the same time. However, note that `unit`, `text`, and `image` each add a column.

Since it is difficult to enter multiline definitions here, as is possible in invoice or customer notes, the sections can be separated by a delimiter (default: `;`).

For example: `unit:=m;text:=hello;link:=http://www.lvbprint.de;image:=myimage.png`

All options are used and explained in the examples.

2.2 Limitations

The following limitations are known:

- Only discounts (as an amount or percentage) on net amounts are supported, i.e. with the `<` sign.

Discounts with `'='` and `'>'` do not work, at least for digital invoices. They are rejected with an error message.

3. Installation and Requirements

Details on installation can be found in the [Installation Manual](#).

In brief:

- Extract the `elInvoiceBridge` archive to any directory.
- Install and update **MiKTeX**.
- Go to the `Examples` subdirectory and run:
 - `RunTexExamples.cmd`: ZUGFeRD and XRechnung files + validation
 - `RunTexExamplesNoValidation.cmd`: ZUGFeRD and XRechnung files without validation (no Java required)
 - `RunTexExamplesInvoiceOnly.cmd`: Generates only the PDF invoices without an embedded digital invoice
- If an error occurs, install .NET Runtime 8.0 if necessary or update MiKTeX.

The generated invoices should be located here: [C:/ProgramData/elInvoiceBridge.Invoices/tex/](#)

4. Getting Started

4.1 Using Your Own GnuCash File

It is best to save your GnuCash file in SQLite format if it is not already in this format. Initially, it is best to work with a copy.

GnuCash **XML** format is not supported.

- Save your SQLite file, for example, as `C:\MyGnuCash\MyGnuCashFile.gnucash`.
- Change the [database connection](#) in `bin\cli\config\appsettings.json` in the Connection section.
Instead of
`"File": "examples/GnuCash/SKR03_Music.gnucash",`
enter
`"File": "c:/MyGnuCash/MyGnuCashFile.gnucash",`
You can also use a backslash `\`, but it must be doubled (JSON escape).
- By changing the connection in the configuration file, the references in the batch files remain valid.

You can now try the tests with your database. Run one of the `RunTexExamples*` batch files again.

Log output can be found here:

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

If everything works correctly, you will find the invoices here:

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

4.2 Checking Address Data

Especially when creating digital invoices, the address data must be parsed correctly so that it can be assigned to the corresponding BT fields.

The `Examples` directory contains the batch file `RunAddressCheck.cmd`. It creates

`C:\ProgramData\eInvoiceBridge.Invoices\addresses.csv` containing all addresses found.

Open the file afterwards and check whether everything was recognized correctly.

In particular, the fields with capital letters in the heading should be correct.

Something is not correct – what now?

- If no destination country is specified in the customer address or the country cannot be recognized, the entry `"DefaultCountry": "Germany"`, from `appsettings.json` is used. Adjust the entry to your home country as appropriate.
- The `config` directory contains the file `countries.json`. This serves as the basis. You can add aliases if necessary, as is done for Germany or Austria.
- When searching the file, the fields `name`, `short`, `alpha2`, `alpha3` and `alias` are checked. `alias` is a comma-separated list of names.
- Addresses from United Kingdom, Ireland, Canada, USA, Japan should be recognized. Many addresses have the same structure as German addresses and should therefore work as well. If something is missing and you use an ISO address format, please report it.
- If necessary, adjust the addresses in GnuCash.

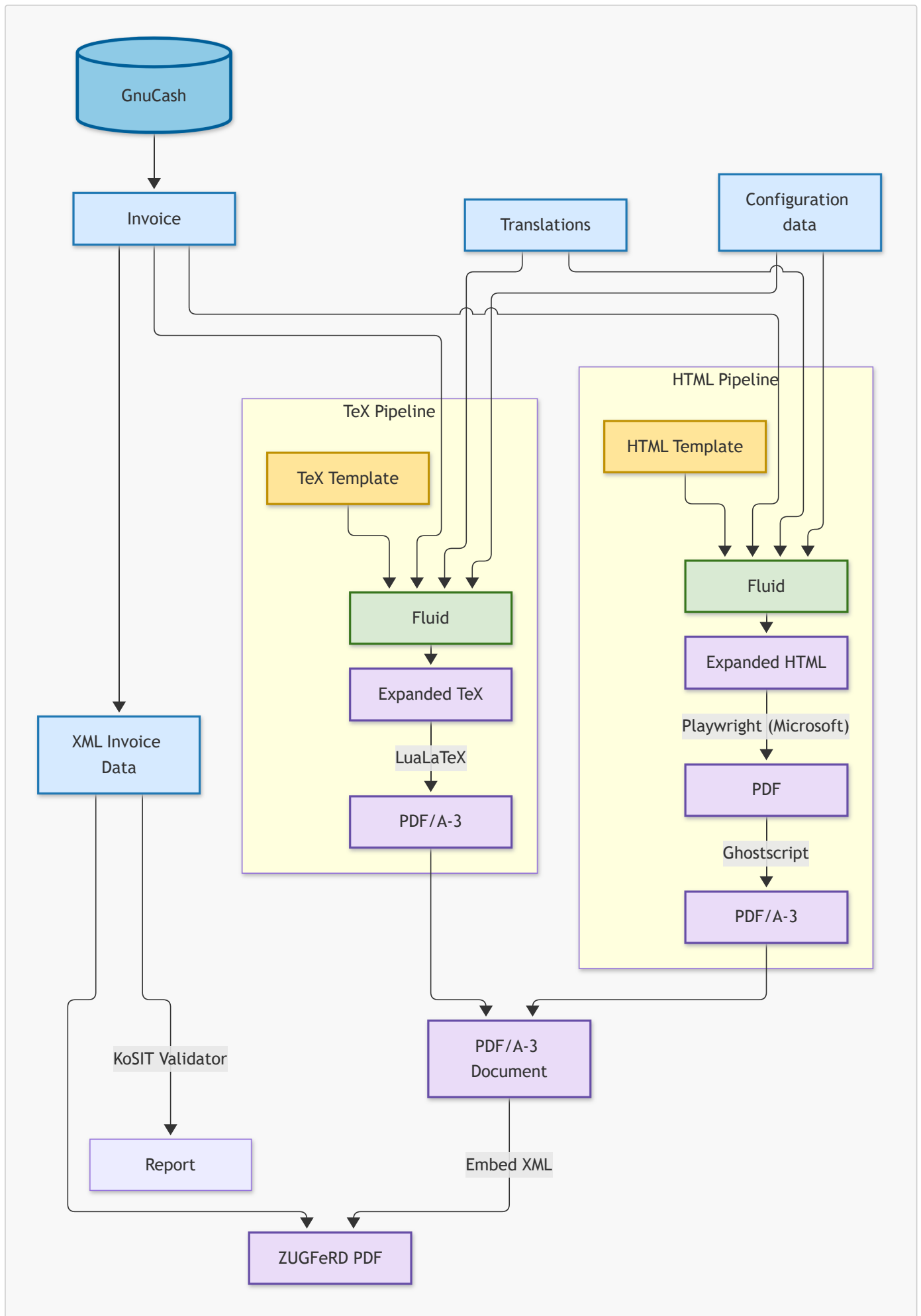
4.3 Entering Company Data

The invoices now probably contain some of your company data from GnuCash and some bank data that cannot be stored in GnuCash. Compare them with the example GnuCash file if necessary to determine which data is required.

Bank details are configured in `appsettings.json`. This is the central configuration file for all configuration data.

You can also create a specific settings file and use it with the `-CF <file>` option.

5. Overview



The diagram shows the internal workflow and is intended to improve understanding.

- Invoice data comes from a data source. Currently, only GnuCash is supported.
 - The data is converted into an internal `invoice` format that is heavily oriented towards *GnuCash*.
 - A text-based template – currently `tex` and `html` (others such as `rtf` would be conceivable) – is expanded using the powerful `Fluid` library. *Fluid* is an implementation of the *Liquid* language originally developed by *Shopify*. See [Fluid Language](#).
 - The expanded templates are then converted to PDF/A-3 in different ways:
 - For *TeX*, this is done using `LuaLaTeX`, which is included in common `TeX` installations.
 - For *HTML*, `Playwright` with the Chromium engine is used to create a PDF. Unfortunately, this does not yet have the required format, so `Ghostscript` is additionally required. This makes the process somewhat more complicated ...
 - For digital invoices, an appropriate *XML* file is generated. Only the `CII` format and the `Comfort` and `XRechnung` profiles are supported. Since digital invoices are not read/imported, the most modern settings can be used. This reduces the testing effort.
 - If a ZUGFeRD invoice (XML embedded in PDF/A-3) is required, the XML is embedded in the generated *PDF/A-3*. Authorities generally require the *XML* file, while other recipients prefer the more readable *PDF*. Both formats can be generated.
 - Finally, the result can be checked using the included `KoSIT Validator`. This is definitely useful at the beginning. Java is required to run it (see the installation manual; Java Runtime Environment (JRE 11 or newer)). The process takes relatively long. An *HTML* and *XML* report is generated for each invoice checked. There are also sufficient online tools for checking and, where applicable, visualizing the invoice. [PortInvoice](#) also checks the PDF for validity and provides invoice visualization.
- If changes to the XML are necessary, I would recommend [PDF24](#). You can load the XML file there, make any necessary changes, and create a new PDF/XML. Naturally, you must then **not use the PDF invoice generated previously**.

6. Calling the Application

- Without arguments, or with `-h`, `--help`, `-?`, the integrated short help is displayed and the program terminates immediately with exit code 0.
- All options can be specified with either `-` or `/` as the prefix (e.g. `-DBType` or `/DBType`).
- Boolean switches are specified with `+` (enable) or `-` (disable), e.g. `-OP+`.
- `-CF <path>` can be used to specify an alternative configuration file instead of `config/appsettings.json`.

7. Configuration File in Detail

Here is a tabular overview of the most important settings from `config\appsettings.json`. Command-line parameters override values from the file. The most important settings are explained in the following chapters. Most settings are documented in the file itself.

Parameters can also be passed without abbreviations. The command must begin with `--`, and a value is specified with `=` instead of a space, for example:

The following settings are compatible, for example:

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

Setting	CLI	Description
eInvoice.Connector.GnuCash		
eInvoice.Connector.GnuCash:PaymentTerms		Used as the default value for payment terms. Normally, payment terms are selected for the invoice (Tip: The examples often use <i>Common</i> for a translated standard text.)
eInvoice.Connector.GnuCash:CultureMode	-CM	Mode for selecting the language. See Localization .
eInvoice.Connector.GnuCash:Culture	-CC	Language when CultureMode is <i>Force</i> or as fallback. See Localization .
eInvoice.Connector.GnuCash:DefaultCountry		Used if no country is specified in the address.
eInvoice.Connector.GnuCash:NotesDelimiter		Operator used within GnuCash for settings in notes; escaped by doubling it, e.g. <code>:=:=</code> .

Setting	CLI	Description
eInvoice.Connector.GnuCash:TranslateNotes		Allows Fluid syntax for translations in buyer and invoice notes.
eInvoice.Connector.GnuCash:TranslateItems		Allows Fluid syntax for invoice-item descriptions.
eInvoice.Connector.GnuCash:ActionInterpretation		Used if not overridden; all options are shown in the examples.
References		The five settings such as BuyerRefNotes can be used to define names for assigning references.
Bank details		Account details cannot be configured in GnuCash; this is done here.
eInvoice		
eInvoice:ZUGFeRDVersion		Only the latest setting Version23 is supported.
eInvoice:ZUGFeRDFormat		Only CII is supported.
eInvoice:ZUGFeRDProfile		Comfort / XRechnung, default <i>XRechnung</i> .
eInvoice:BusinessProcess		Can be adjusted if necessary.
eInvoice:Fluid:Assignments:_extension	-Ext	Switch between TeX and HTML templates; this is done, for example, in the batch files.
eInvoice:Fluid:Assignments:*		All others are used in the TeX layout and can be overridden by translations, buyer notes, or invoice notes; see examples.
eInvoice:Fluid:curly_bracket_open_replacement		Definition of a special character that can be used in templates and is later converted to {.
eInvoice:Fluid:curly_bracket_close_replacement		Same for the closing brace; both are explained below under Fluid Language .
eInvoiceBridge.Core		
eInvoiceBridge.Core:Connection:*		Explained in detail in Database Connection .
eInvoiceBridge.Core:Validation:*		Settings for calling the KoSIT Validator; normally no changes should be necessary.
eInvoiceBridge.Core:Adressen:*		Valid addresses are a prerequisite for digital invoices. An example batch file extracts all addresses to a CSV file for checking.
eInvoiceBridge.Core:Filter:*		Settings for filtering invoices; see Filters .
eInvoiceBridge.Core:TemplateFiles	<i>not possible</i>	Selection of the template to use; there is a separate chapter for this and it cannot be overridden via the command line.
eInvoiceBridge.Core:PDFCreationCommands:*		Other tools can be configured here to create PDF/A-3 from a template. If the application is not in the PATH, its path must be

Setting	CLI	Description
		entered here.
eInvoiceBridge.Core:Ghostscript:*		Settings for calling Ghostscript.
eInvoiceBridge.Cli		
eInvoiceBridge.Cli:InvoiceOutputFile		Name of the output file for the invoice (<i>PDF</i>).
eInvoiceBridge.Cli:XRechnungOutputFile		Name of the output file for the XRechnung file (<i>XML</i>).
eInvoiceBridge.Cli:ValidationOutputPath		Output path for validation reports; the name is generated automatically by the KoSIT tools.
eInvoiceBridge.Cli:MaxDegreeOfParallelism		Maximum number of simultaneous invoice creations when multiple invoices are processed. See Parallel Processing .
eInvoiceBridge.Cli:Create*		See Output Options .

8. Database Connection

The connection to the GnuCash database can be configured in two ways.

8.1 Direct Settings

Supported database types: `SQLite`, `MySQL`, `PostgreSQL`.

For MySQL/PostgreSQL, the following are additionally required: `Host`, `Port`, `DatabaseName`, `User`, `Password`. Missing values are filled with sensible defaults:

Setting	Default value
Host	localhost
Port (MySQL)	3306
Port (PostgreSQL)	5432
DatabaseName	internal default (gnucash)

CLI parameter	Configuration path	Description
-DBType	eInvoiceBridge.Core:Connection:Type	Database type
-DBFile	eInvoiceBridge.Core:Connection:File	File path (SQLite only)
-DBHost	eInvoiceBridge.Core:Connection:Host	Host
-DBPort	eInvoiceBridge.Core:Connection:Port	Port
-DBName	eInvoiceBridge.Core:Connection:DatabaseName	Database name
-DBUser	eInvoiceBridge.Core:Connection:User	User name
-DBPassword	eInvoiceBridge.Core:Connection:Password	Password

8.2 Encrypted Connection Key

For production use, it is recommended not to store credentials in plain text. When starting with complete connection data and the additional parameter `-DBCcreateKey`, the application outputs an encrypted key to the console.

This value can then be stored permanently.

The corresponding CLI parameter for direct use is `-DBKey <key>`. If `EncryptedKey` is set, it takes precedence over all other connection settings.

Only simple, computer-specific encryption is used.

9. Filters

Without at least one filter being set, no invoices are processed – this is a deliberate safety measure to prevent accidental mass processing.

CLI parameter	Configuration path	Description
<code>-FI</code>	<code>eInvoiceBridge.Core:Filter:InvoiceId</code>	Filter by invoice number
<code>-FC</code>	<code>eInvoiceBridge.Core:Filter:Customer</code>	Filter by customer name
<code>-FD</code>	<code>eInvoiceBridge.Core:Filter:Date</code>	Filter by date

All filters support two modes:

- **Exact/wildcard search** (default): e.g. `-FD 202609*` (all invoices from September 2026). `?` and `*` are supported as in file searches.
- **Regular expression** with the `r:` prefix: e.g. `-FI r:0*[23]$` (invoice numbers ending in 2 or 3).

To process all invoices, the following expression can be used: `-FI *`.

10. Output Options and Their Dependencies

CLI parameter	Meaning
<code>-OT+</code> / <code>-OT-</code>	Create and retain expanded template / delete it again
<code>-OX+</code> / <code>-OX-</code>	Create XRechnung document / do not create
<code>-OZ+</code> / <code>-OZ-</code>	Create ZUGFeRD (embedded in PDF) / do not create
<code>-OP+</code> / <code>-OP-</code>	Create and retain original PDF / delete it again
<code>-OV+</code> / <code>-OV-</code>	Create validation report / do not create

10.1 Implicit Dependencies

Since some outputs depend on others, missing prerequisites are automatically enabled – but only temporarily (the intermediate file is deleted after use unless it was explicitly requested with `+`):

- `-OZ+` implicitly enables `-OP` (if not set); the generated PDF is deleted after embedding unless `-OP+` was explicitly specified.
- `-OP+` implicitly enables `-OT` (if not set); the template file is then deleted unless `-OT+` was explicitly specified.
- `-OV+` implicitly enables `-OX` (if not set); the XRechnung file is then deleted unless `-OX+` was explicitly specified.

10.2 Output Paths

The paths for the individual outputs are dynamically expanded per invoice using Fluid/Liquid placeholders.

Note: `InvoiceOutputFile` does not require a file extension – it is automatically appended based on the template found (e.g. `.html` or `.tex`).

11. Fluid Language

Fluid is an implementation of the `Liquid` language originally developed by Shopify. It is a powerful way to expand text files.

The relevant constructs are enclosed either in `{% ... %}` for more complex language constructs such as loops and conditions, or in `{{ ... }}` for variables.

A good overview of the available features can be found here: [Fluid Language](#).

Here are two examples from the TeX template to explain some points:

```
\newcommand{\InvoiceNotes}{\% for entry in Notes.List %}{\entry|escape_ext}}\% endfor %}}
\textbf{\{_account_holder\}:} & \SellerOwner & \textbf{\{_account_bank_name\}:} & \SellerBank \\\
```

TeX syntax is black

Fluid syntax is blue

The special points are:

- First, the strange characters `{}` and `}` in TeX stand out. Since both TeX and Liquid use curly braces in special ways, they must be distinguished. The TeX braces are therefore replaced by two Unicode characters. You can replace their definitions with other characters in *appsettings.json* if necessary. After the Fluid call, all characters are replaced. Incorrect use can cause compiler errors in both Fluid and TeX, especially in combination with `%`, because `%` also has a special meaning in TeX (comments).
- `{% for ... %}` is, like `{% if ... %}` and many others, Fluid syntax for language constructs such as conditions and loops.
- `Notes.List` is a variable coming from the invoice. Since *List* is a collection, it can be iterated over. A list of variables is provided later. Variable names are capitalized here.
- `_account_bank_name` is a variable coming from *appsettings.json*. There are other ways to inject variables as well. Examples include `_logo`, `_background`, etc. More on this later.
- `escape_ext` is a filter defined by the program. There are many filters included in the language ([Fluid Language](#)) and some defined by the program. See the table below.

Here is an extreme example from the configuration for the output file:

This is not TeX, but pure Fluid.

In the current configuration file, the date has been removed. It is still included here as an example.

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

- `CommonApplicationData` is one of several predefined variables. Under Windows, it expands to `C:\ProgramData`.
- `_extension` is a variable defined in the `Fluid:Assignments` section of *appsettings.json*. This means the template extension only has to be defined in one place.
- `Buyer.Company.Name` is again a variable from the invoice.
- `remove_blanks` and `replace_invalid_filename_chars` are filters defined by the eInvoice Toolkit. As their names suggest, they remove whitespace and invalid characters from the company name so that it can be used as a file name.
- `format_date` is a filter provided by Fluid.
- `InvoiceId` is the invoice number from the invoice.
- `CI` is the language that was determined. There are several ways to determine it.

12. Invoice Templates

The invoice data is inserted into a template using the **Fluid** template engine (Liquid syntax). The search for the appropriate template uses a configurable list of file paths.

Template search procedure:

1. The list is searched in order.
2. If an entry contains placeholders (`{}`), it is first expanded with the invoice data (e.g. to find a culture-specific file such as `invoice_de-AT.html`).
3. The first existing file is used.
4. If no file is found, processing fails with an error message listing all file names that were searched.

The template file found is completely expanded with the invoice data (see the appendix for available placeholders) and then saved as a file.

The extension of the output file is based on the extension of the template file (e.g. `.html`, `.tex`).

13. PDF Creation

PDF creation is performed in one of two possible ways, depending on the extension of the expanded template.

13.1 Using an External Shell Command (recommended for LaTeX etc.)

A separate command can be configured for each file extension.

Properties of the shell command:

Property	Description
ExePath	Directory of the executable (optional)
Cmd	Name/path of the executable
Args	Arguments; they support Fluid placeholders including <code>{{ srcFile }}</code>
ExitCodeOnSuccess	Expected exit code on success
RunTwice	Execute command twice (e.g. necessary for LaTeX references)
RemoveLogFilesOnSuccess	Delete temporary log files after success
TempfileExtension	Space-separated list of file extensions to delete after success

The standard output and error output of the process are automatically logged to `<InvoiceNo>.log` and `<InvoiceNo>_err.log` in the template's working directory.

13.2 Internally via Playwright (for HTML Templates)

If no external command is configured for the `.html` extension, **Playwright/Chromium** is used internally to render the HTML document directly to a PDF (A4 format, with margins of 20 mm top/bottom and 10 mm left/right). The PDF is then converted to a PDF/A-3-compliant document using **Ghostscript** (required for ZUGFeRD).

Note: For file extensions other than `.html`, an external command **must** be configured because no internal support exists.

14. ZUGFeRD Creation and Embedding

The ZUGFeRD configuration determines the version, profile, and format of the XRechnung XML that is subsequently embedded in the PDF:

CLI parameter	Description
-ZV	ZUGFeRD version (currently supported: <code>Version23</code>)
-ZP	ZUGFeRD profile (Comfort, XRechnung)
-ZF	ZUGFeRD format (CII)

Procedure when `-OZ+` is enabled:

1. The PDF is created (if it does not already exist).
2. XRechnung XML is generated.
3. If generation is successful, the XML is embedded in the PDF (resulting in a hybrid ZUGFeRD document).
4. If XRechnung creation fails, embedding is skipped; a PDF previously created only for ZUGFeRD is deleted unless `-OP+` was explicitly set.

15. XRechnung Validation

Validation is performed using an external, Java-based validator (e.g. the official KoSIT Validator). Requirements:

- A working **Java runtime environment**. The path is determined via:
 1. Configuration value `eInvoiceBridge.Core.Validation:JavaHome`
 2. System environment variable `JAVA_HOME`
- A **validator directory** containing the validation JAR. The path is determined via:
 1. Configuration value `eInvoiceBridge.Core.Validation:ValidatorPath`
 2. Automatic search for a directory named `Validator` in the parent directories of the program directory

15.1 Configuration

Setting	Description
JavaHome	Path to the Java installation (overrides <code>JAVA_HOME</code>)
JavaOptions	Additional JVM options

Setting	Description
ValidatorPath	Directory containing the validator JAR
ValidatorJarMask	Search pattern for the JAR file (if several matches exist, the alphabetically last match is used)
ValidatorExtOptions	Additional command-line options for the validator
ValidatorResultsPath	Output path for intermediate results (default: same as report path)
ValidatorScenarioFile	Scenario file defining the validation rules to apply

15.2 Procedure and Result

The validator is started as an external Java process. The standard output is searched for the pattern `Acceptable: <n> Rejected: <n>`:

- **Rejected > 0** or an **exit code > 0** results in a validation error.
- On success, an **HTML report** (`<filename>-report.html`) is optionally generated (with `-OV+`), in addition to the XML report.

16. Localization / Culture Settings / Translations

There are two important settings in `appsettings.json` for the language of the invoice.

- In CultureMode `Force`, the value configured in `Culture` is always used (the default in the configuration file is no language). In this example, all invoices would therefore be generated in French.
- In CultureMode `Auto`, there is a defined search order. First, the customer address is used and the most commonly used language for the country is selected. This does not always work in multilingual countries such as Switzerland or Belgium. For this reason, you can select another language in the customer notes, and even this can be overridden per invoice.

Here is an example from invoice `000003`. The same applies to customer notes.

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

The characters `:=` are important here. They have a special meaning and can be defined in the configuration file. They are removed from the notes, so only the first line remains on the invoice.

- `culture` and `action` are predefined and allow the language to be overridden in `Auto` mode and the `Action` column to be interpreted. Case is ignored on the left-hand side.
- `_logo` overrides the variable defined in `appsettings.json`. You can define additional variables if you wish. The examples use these for layout settings (e.g. `_tex_font`).

The values can also be overridden via the command line.

CLI parameter	Description
<code>-CC</code>	Culture in "Force" mode, e.g. <code>de-AT</code>
<code>-CM</code>	Culture mode: <code>Auto</code> (derived from invoice data/notes) or <code>Force</code> (forced via <code>-CC</code>)
<code>-Ext</code>	File extension for generated templates, e.g. <code>tex</code> or <code>html</code> (configuration path: <code>eInvoice:Fluid:Assignments:_extension</code>)

The actual culture (CI) used for an invoice is either derived from GnuCash notes/address data (`Auto` mode) or explicitly specified with `-CC` (`Force` mode). It affects both the selection of the invoice template (see chapter 12) and the formatting of numbers and dates in the generated document.

Now that it is clear how the language is set, it is useful to know where translations are applied.

- **Translation tables**
All tables matching the pattern `*Translations.json` in the `config` directory are interpreted as translation tables. The following files are included by default (all keys start with an underscore and a lowercase letter to distinguish the texts from other content, such as invoice contents, in the template):

- *InvoiceTranslations.json* for translating headings in the template
- *OtherTranslations.json* examples for translating invoice items and notes

- **Invoice items**

In theory, you can translate invoice-item descriptions. Examples are provided in the German and Austrian invoices, where *Tomaten* is translated differently. If problems occur, you can disable this option in *appsettings.json* (*TranslateItems*).

- **Invoice notes**

You may also want invoice notes to be multilingual. Example: Example from invoice notes: `'{{_invoice_notes_example | escape_ext}}'`. This option can also be disabled if necessary (*TranslateNotes*).

- **Payment terms**

The example invoices contain a translation for payment terms named *Common* with the content `{{_pay_until_duedate | escape_ext}}`. The text and translations are found in the *InvoiceTranslations.json* file.

For all translations, the text must be enclosed in `{{ . . }}` and, if it contains special characters, the `escape_ext` filter should also be used. This replaces special characters for TeX and HTML with the appropriate escape sequences.

17. GnuCash-Specific Settings

CLI parameter	Configuration path	Description
<code>-Action</code>	<code>eInvoice.Connector.GnuCash:ActionInterpretation</code>	Interpretation of the GnuCash "Action" column per invoice item

Possible values for *ActionInterpretation*:

Value	Meaning
<code>None</code>	No special interpretation (default)
<code>Unit</code>	Content of the Action column is interpreted as a unit (e.g. "pcs", "h")
<code>Image</code>	Content is interpreted as an image reference (e.g. product image in the template)
<code>Text</code>	Content is interpreted as additional free text for the item
<code>Link</code>	Description receives the content as a link

The resulting invoice properties *ActionIsUnit*, *ActionIsImage*, *ActionIsText*, *ActionIsLink* are available in the template (see appendix).

18. Address Export

In addition to invoice creation, a list of all addresses known to the system can be exported (e.g. for comparison with other systems):

Setting	Description
<code>Dump</code>	Enable address export
<code>OutputFile</code>	Destination file (supports Fluid placeholders and paths using <code>/</code>)
<code>Delimiter</code>	CSV delimiter (default: <code>;</code>)
<code>Encoding.Name</code>	Character encoding: <code>utf-8</code> , <code>utf-16</code> , <code>utf-32</code> , or any code-page name supported by .NET (e.g. <code>windows-1252</code>)
<code>Encoding.WithBom</code>	Add byte-order mark (relevant only for UTF encodings)
<code>Encoding.BigEndian</code>	Big-endian byte order (relevant only for UTF-16/32)

Address export is performed independently of the filter settings and takes place before actual invoice processing.

19. Parallel Processing

Multiple invoices are processed in parallel to efficiently handle large quantities. The degree of parallelism is controlled by `eInvoiceBridge.Cli:MaxDegreeOfParallelism` (default: 1):

- The value is automatically limited to at least 1 and at most the number of available processor cores.

- Invoices are fully loaded from the database in advance (to avoid parallel database access) before parallel processing begins.
- If an error occurs for an individual invoice, it is logged; processing of the remaining invoices continues.

20. Logging

The application uses **log4net**, configured via `config/log4net.config`. Log files are stored by default under:

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

Dedicated loggers also exist for specific parts of the program:

Logger name	Purpose
CreatePDF	Logs PDF creation
Validation	Logs the validation process

The log levels (Debug, Info, Warn, Error) can be configured entirely via `log4net.config` and independently for each logger.

21. Error Handling and Return Values

- Invoices with validation errors (`invoice.Valid == false`) are skipped; XRechnung creation is not performed for these invoices.
- If XRechnung creation fails, an embedding operation already started for ZUGFeRD is aborted.
- A PDF created only as an intermediate step for ZUGFeRD is automatically deleted if XRechnung creation fails (unless `-OP+` was explicitly set).
- Temporary intermediate files (expanded template, XRechnung XML) are automatically removed after use unless they were explicitly requested permanently with `-OT+` or `-OX+`.
- **Exit code:**
 - `0` – complete success
 - `1` – at least one error occurred during processing (e.g. failed validation, file access error, template not found)

22. Complete Command-Line Reference

Parameter	Description
<code>-CF <path></code>	Configuration file (JSON)
<code>-DBKey <key></code>	Use encrypted connection key (overrides other DB settings)
<code>-DBType <type></code>	Database type: SQLite, MySQL, PostgreSQL
<code>-DBFile <path></code>	Database file path (SQLite only)
<code>-DBHost <host></code>	Database host (default: localhost)
<code>-DBPort <port></code>	Database port (default: 3306/5432)
<code>-DBName <name></code>	Database name
<code>-DBUser <user></code>	Database user
<code>-DBPassword <pwd></code>	Database password
<code>-DBCreateKey</code>	Generate encrypted connection key (for <code>appsettings.json</code>)
<code>-FI <id></code>	Filter by invoice number (<code>r:</code> for regular expressions)
<code>-FC <customer></code>	Filter by customer name (<code>r:</code> for regular expressions)
<code>-FD <date></code>	Filter by date, e.g. 2024-01-01 (<code>r:</code> for regular expressions)
<code>-OT+ / -OT-</code>	Create and retain expanded template / delete it again
<code>-OX+ / -OX-</code>	Create XRechnung document / do not create
<code>-OZ+ / -OZ-</code>	Create ZUGFeRD (embedded in PDF) / do not create
<code>-OP+ / -OP-</code>	Create and retain original PDF / delete it again
<code>-OV+ / -OV-</code>	Create validation report / do not create

Parameter	Description
-CC <culture>	Culture in "Force" mode
-CM <mode>	Culture mode: Auto or Force
-Ext <extension>	File extension for generated templates, e.g. <code>tex</code> or <code>html</code>
-ZV <version>	ZUGFeRD version (Version23)
-ZP <profile>	ZUGFeRD profile (Comfort, XRechnung)
-ZF <format>	ZUGFeRD format (CII)
-Action <action>	GnuCash action interpretation (None, Unit, Image, Text, Link)
-h, --help, -?	Display help

23. Common Problems (Troubleshooting)

Problem	Possible cause	Solution
"No invoices found matching the specified filter criteria"	No matching filter or incorrect filter values	Check filter values; test with <code>-FI *</code> if necessary
"No default database connection configured"	Neither <code>EncryptedKey</code> nor <code>Type</code> is set in the configuration	Add connection data in <code>appsettings.json</code> or via CLI
"JAVA_HOME not configured. Validation cannot continue"	Java is not installed or the environment variable is missing	Install Java and set <code>JAVA_HOME</code> , or use <code>-OV-</code>
"Validator folder not found"	No <code>Validator</code> directory in the program directory or its parent directories	Configure <code>ValidatorPath</code> explicitly
"No template file found for invoice ... with Culture ..."	No suitable template exists for the detected culture	Create a template for the corresponding culture or add a fallback entry without a culture placeholder to <code>TemplateFiles</code>
"Failed to get pdf creation command. No command defined and no internal support for extension"	Template has an extension other than <code>.html</code> , but no <code>PDFCreationCommands</code> entry is configured	Add the appropriate shell command to <code>PDFCreationCommands</code>
Validation fails despite an apparently correct XRechnung	Incorrect or outdated <code>ValidatorScenarioFile</code>	Use the current scenario file of the validator
Invoice is not processed despite a filter	<code>invoice.Valid == false</code> (data error in GnuCash)	Check the error message in the log (<code>invoice.ErrorMessage</code>) and correct the accounting data

24. Appendix: Available Template Placeholders

The following properties of an invoice are available in Fluid/Liquid templates. Access is via Liquid syntax, e.g. `{{ InvoiceId }}`. Most of them are used in the templates.

24.1 General Invoice Data

Property	Type	Description
Validity		
<code>Valid</code>	<code>bool</code>	Indicates whether the invoice is valid and can be processed completely.
<code>ErrorMsg</code>	<code>string</code>	Error message if the invoice is invalid.
Presence of specific data		

Property	Type	Description
HasReference	bool	Indicates whether at least one invoice reference is present.
HasVat	bool	Indicates whether VAT data is present.
HasMultipleVat	bool	Indicates whether multiple different VAT rates are used.
HasDeliveryAddress	bool	Indicates whether a delivery address is present.
HasItemsDiscount	bool	Indicates whether at least one invoice item contains a discount.
HasTaxCodes	bool	Indicates whether tax categories or tax codes are present.
NeedsTaxCodesColumn	bool	Indicates whether the output needs an additional column for tax codes.
Basic data		
InvoiceId	string	Unique invoice number or invoice identifier.
InvoiceType	string	Invoice type as a translated string.
CI	CultureInfo	Culture information for language, date, number, and currency formatting.
Symbol	string	Currency symbol, e.g. €.
Currency	string	Currency code, e.g. EUR.
Notes	NotesParser	Parsed invoice notes, including free text and key-value entries.
DueDate	DateTime?	Payment due date.
InvoiceDate	DateTime?	Invoice date.
DeliveryDate	DateTime?	Delivery or service date.
TermsDescription	string	Description of the payment terms.
Payed	decimal?	Amount already paid.
Actions	Dictionary<string, bool>	Action types for which data is present in the invoice.
References		
ProjectId	string	Project number or identifier.
ProjectName	string	Project name.
Contract	string	Contract reference.
PurchaseOrder	string	Purchase order number or reference.
SalesOrder	string	Sales order number.
Nested objects		
Seller	SellerData	Seller or invoice issuer data.
Buyer	CustomerData	Buyer or invoice recipient data.
Items	List<IInvoiceEntry>	List of invoice items.
Totals and other data		
IsCreditNote	bool	Indicates whether the invoice is a credit note or negative invoice.
Terms	BillTerms	Invoice payment terms. The terms defined on the invoice take precedence over the customer's payment terms.
AllPaid	bool	Indicates whether the amount due has been paid in full.
ItemsCount	int	Number of invoice items.
TermsDueDays	int	Number of days until payment is due. Returns 0 if Terms is not specified.

Property	Type	Description
SumItems	List<InvoiceEntry>	List of accumulated invoice items. Each item contains the totals accumulated up to and including that item.
TaxEntries	List<InvoiceTaxEntry>	List of tax data grouped by tax rate.
TaxCodes	List<TaxData>	List of all tax data used in invoice items. A separate entry may be included for tax-exempt or non-taxable items.
NetAmountSum	decimal	Total net amount of the invoice before tax.
AmountToPay	decimal?	Amount still payable after deducting payments already made.
TaxSum	decimal	Total tax amount for the invoice.
RoundingDiff	decimal	Rounding difference between individual item totals and the invoice total.
GrossAmountSum	decimal	Gross invoice total including tax.

24.2 SellerData / CustomerData / Addresses

Member	Data type	Description
SellerData		
Company	CompanyData	Seller's company data.
Address	AddressData	Seller's address.
Account	AccountData	Seller's bank and payment details.
CustomerData		
BuyerReference	string	Buyer reference according to BT-10, e.g. a Leitweg-ID.
Terms	BillTerms	Customer's payment terms.
Company	CompanyData	Customer's company data.
Address	AddressData	Customer's billing address.
ShipAddress	AddressData	Delivery address.
Notes	NotesParser	Parsed customer notes containing additional configuration values.
CompanyData		
Name	string	Legal or general company name.
TradingName	string	Trading name or alternative company name.
Vat	string	VAT identification number.
Tin	string	Tax identification number, particularly for B2C invoices.
Url	string	Company website.
TaxRegistration	string	Additional tax registration information.
LegalInformation	string	Legal company information, e.g. commercial register details.
ElectronicAddress	string	Company's electronic address, e.g. an e-invoicing address.
AddressData		
OrgAddrLines	List<string>	Original address lines, without further processing.
AddrLines	List<string>	Cleaned address lines without trailing blank lines.
HasAddress	bool	Indicates whether usable address or contact data is available (particularly useful for delivery addresses).
Contact	string	Contact person.

Member	Data type	Description
Phone	string	Telephone number.
Fax	string	Fax number.
Mail	string	Email address.
PostCode	string	Postal code.
City	string	City or town.
Street	string	Street and house number.
Country	string	Country name.
SubCountry	string	State, region, or province.
CountryCode	string	Country code, e.g. DE.
Culture	string	Assigned culture code, e.g. de-DE.
AccountData		
Type	string	Payment method or Payment Means.
Owner	string	Account holder.
BIC	string	Bank BIC or SWIFT code.
IBAN	string	International bank account number.
Bank	string	Bank name.
BillTerms		
Name	string	Name of the payment term.
Description	string	Description of the payment term.
DueDays	int	Number of days until payment is due.

24.3 Tax Data

Member	Data type	Description
TaxData		
TaxCategory	string	Tax category, e.g. S, K, G, or AE.
VatexCode	string	VAT-EX code for tax-exempt or non-taxable transactions.
TaxReason	string	Translated explanation of the tax treatment.
TaxTableEntry		
Amount	decimal	Tax amount or tax factor. For percentage-based taxes, this is the tax rate.
TaxType	TaxTypes	Type of tax calculation, e.g. percentage or fixed amount.
InvoiceTaxEntry		
TaxId	string	Tax code or tax category, e.g. S, K, G, or AE.
TaxRate	decimal	Applied tax rate in percent. For tax-exempt or non-taxable transactions, the value is normally 0.
TaxSum	decimal	Total tax amount for the tax code and rate.
Netto	decimal	Total net amount of all invoice items with this tax code and rate.

24.4 Items and Totals

Property	Type	Description
Items	List	Invoice items
ItemsCount	int	Number of items
SumItems	List	Subtotal items
TaxEntries	List	Tax amounts grouped by tax rate
TaxCodes	List	Tax codes used
NetAmountSum	decimal	Net total
TaxSum	decimal	Total tax amount
RoundingDiff	decimal	Rounding difference
GrossAmountSum	decimal	Gross total
HasVat	bool	At least one tax rate other than 0%
HasMultipleVat	bool	Multiple different tax rates are present
HasTaxCodes	bool	At least one tax code is present
NeedsTaxCodesColumn	bool	More than one tax code is used in the items (a tax-code column may be useful)
HasItemsDiscount	bool	At least one item has a discount
HasDeliveryAddress	bool	Delivery address is present

24.5 Custom Filters

The following custom filters are available in Fluid templates. They are applied to a value using the pipe character (|), e.g. `{{ Filename | to_slash }}`.

Filter	Description	Example / Notes
replace_invalid_filename_chars	Replaces characters that are invalid in file names.	An optional first argument specifies the replacement string, e.g. <code>{{ Invoiceld</code>
escape_ext	Escapes a value according to the template file extension.	<code>.tex</code> : TeX escaping; <code>.ps</code> : PostScript escaping; <code>.html</code> : HTML encoding. For other extensions, the value remains unchanged.
to_slash	Replaces backslashes (\\) with forward slashes (/).	For consistent path specifications.
remove_blanks	Removes all whitespace characters from a value.	Removes spaces, tabs, and line breaks, among others.
find_file	Searches for a file using the configuration-file search and returns the found path.	If no file is found, <code>File <Name> not found</code> is returned.
to_backslash	Replaces forward slashes (/) with backslashes (\\).	For Windows paths.
to_pathchar	Replaces both slash variants with the platform-specific path separator.	\\ on Windows, / on Unix-like systems.

24.6 Predefined Variables

In addition to the properties of the `Invoice` object, `AddInternalConstants` provides the following variables. They can be used like other Fluid variables, e.g. `{{ BasePath }}`.

Variable	Description
CommonApplicationData	Path to the system-wide common application data directory.

Variable	Description
ApplicationData	Path to the current user's application data directory.
LocalApplicationData	Path to the current user's local application data directory.
Personal	Path to the current user's personal documents directory.
CommonDesktopDirectory	Path to the common desktop directory.
CommonDocuments	Path to the common documents directory.
BasePath	Application base directory.
ImagesPath	Path to the <code>Images</code> subdirectory of the application directory.
ConfigPath	Path to the <code>config</code> subdirectory of the application directory.
TemplatesPath	Path to the <code>templates</code> subdirectory of the application directory.
NOW	Current date and time (<code>DateTime.Now</code>).
CS	Currency symbol of the invoice; empty if there is no invoice.
CI	Name of the culture used (<code>CultureInfo.Name</code>).
DirSep	Platform-specific path separator.
CL	Two-letter ISO language code of the culture, e.g. <code>de</code> .
InvoiceDateCI	Invoice date formatted according to the culture used. If no invoice date is available, the current date is used.