The zugferd package*

Creating electronic and hybrid invoices using LaTeX

Marei Peischl
marei@peitex.de

September 11, 2024

Abstract

Invoicing is getting more and more automated. Starting with public sector, within Germany there already is a requirement to stick to the Faktur-X Standard. First Invoices based on this implementation here have been created back in 2021. And this is now the trial to create a more universal and public package to support the current Version of ZUGFeRD and therefore also X-Rechung and Faktur-X.
The fundamental idea of this package was to use the calculation within LaTeX as well. So it also creates the XML filefor the attachment on the fly. To match typical setups there is a wrapper package which usually would also hold the personal Invoicing layout configuration.

Sponsors & Supporters

Most of this package has been created within my free time and for my personal use. At start, it was not a paid project at all. Since it is addressing business users it would be great if we could keep this actively maintained. If you are able to support this either financially for the maintenance effort, a custom extension, I'd love to hear from you.

Tthis project was financially supported by:

- Pengutronix e.K., https://pengutronix.de
  Special thanks to them, as they also sponsored the minimal portable TeX Live setup.

*This document corresponds to zugferd v0.8, dated 2024-09-11.

Contents

1 Quick start

This package is still in development and does not provide any validation. To ensure your invoice is created correctly you should also validate the output files. There are tools like the [7] providing an easy-to-use interface for the validation. In the appendix I will add some notes on my setup and how I use it within pipelines.

The Bundle provides an example file called `DEMO-rechnung-zugferd.tex`. This includes a basic setup for a valid X-Rechnung currently matching Version 3.0.1 of the standard. Details on the requirements can be found in the documentation at [1].

1.1 Disclaimer concerning the zugferd-invoice Package

The included package zugferd-invoice is an example project which might match your own invoicing structure. It holds all the layout information which is static across all the invoices. This package is an example implementation and should not be used in production. It is published as a part of the documentation.

The idea is to create your own version of this package to use your own layout and internally load the zugferd package that way. Of course, it's possible to use a copy of this package within your personal setup. But the syntax used in the DEMO file may change, so you have to ensure yourself to be compatible with updates.

The interfaces for zugferd will hopefully stay the same. At least changes will be announced and build compatible during a deprecation period.

</package>

2 Package Options

The package supports a few fundamental settings. These have to be set when the package is loaded as they are used internally to setup the scheme or activate the XML mechanism.

`format=` (`xrechnung/xrechnung3.0/xrechnung2.3/basic`) (default: `xrechunng`)

`format` selects the scheme to be used for the zugferd invoice. Currently xrechnung3.0, xrechnung2.3 and the basic scheme are supported.

The value `xrechnung` is set as an alias to `xrechnung3.0` and will always use the latest version supported by zugferd.

`zugferd=` ⟨*boolean*⟩ (default: `true`)

This option can be used to deactivate the XML embedding. It would also disable the the `write-xml` option. This can be used to create a package which can use the same structure to also create invoices without XML attachment. It can also be used with older LaTeX releases than this package requires. There will be a warning, but the visible part should be okay.

`write-xml=` ⟨*boolean*⟩ (default: `true`)

Disable the XML output. This can be used if you want to create the XML attachment with different software than this package.

In that case you can either rename your file to ⟨`\jobname_`*zugferd.xml*⟩ or also adjust the `xml-file` option.

`xml-file=` ⟨*filename*⟩ (default: `\jobname_zugferd.xml`)

Adjust the file name of the created or loaded XML file.

The option `xrechnung` is only used internally to set the global parameters for all `xrechnung` variants.

`auto-exemption=` (⟨*boolean*⟩) (default: `true`)
zugferd tries to automatically add an exemption-reason for the most common VAT categories. In case a more specific reason is required this setting can be disabled and everything should be configured manually. See subsubsection 5.3.2 for more explanation of this feature and the categories this applies to.

3 User Commands

The end user is only asked to set or access the data to be used by zugferd.

```
\SetZUGFeRDData*{⟨key value list⟩}
```

`\SetZUGFeRDData` `\SetZugferdData` The two modes of `\SetZUGFeRDData` control if the argument is expanded before the fields are set. Depending on the source of the data this might be necessary. Fields which are involved in the calculation will be expanded anyway, but the text fields will not, to support special characters.

`\InsertZUGFeRDData` `\InsertZugferdData`

```
\InsertZugferdData[⟨special mode option⟩]{⟨data-selection⟩}
```

ZUGFerd uses the same data as the XML fileinside the PDF. To simplify the reuse of data this command is designed to simplify the access to data fields, for example:

```
\InsertZUGFeRDData{id}
{\InsertZUGFeRDData[set-today]{date}\today}
\InsertZUGFeRDData[AddressData]{seller}
```

As special modes the command currently supports the following:

By default zugferd tries to find the variable holding the data itself. First a token list is tried, afterwards a string. Global variables are prefered over local ones.

As the variable names may container underscores and the option usually prefers dashes, dashes are converted to underscores for the detection.

`AddressData` Allows `seller` or `buyer` for the data selection. Will print the address, to be used in letters.

`set-today` For dates there also exists the variant which will not print the variable but parse the variable to be used as `\today`. Using this the date format can be controlled easier using the language setting of the document. Here you should take care to use it within a group to restore the real value of `\today` afterwards.

4 Commands for template authors

`ZUGFeRD` (*env.*) To simplify the structure of the wrapper package, zugferd provides an environment for the XML mechanism and does the attachment to the PDF file (of course only, if enabled, see section 2). This provides the public interface bundling some steps together to reduce maintenance effort for any template maintainer using this package. It also avoids the use of internal commands.

This environment opens the XML file using `\startWritingZUGFeRDxml` and afterwards writes the XML header including the File and Scheme information, the `ExchangedDocumentContext` and information of the `ExchangedDocument`. Notes will also be written within this step. Afterwards the environment should include all the mechanisms to write the invoice positions as well as summation.

At the end of the environment the footer is inserted, before the output stream is closed using `\stopWritingZUGFeRDxml`. Which also attaches the XML fileto the PDF.

`\startWritingZUGFeRDxml` `\startWritingZUGFeRDxml` is opening the output stream for the XML file. It also adjusts the indentation. If `write-xml` is false, this option only opens a group to achieve the same structure in both modes.

`\stopWritingZUGFeRDxml` Here the output stream is closed and the XML fileis attached. In case `write-xml` is not active, the attachment will be made if that's not deactivated separately using `zugferd`. It also ends the group started by `\startWritingZUGFeRDxml`.

4.1 Interfaces to write the XML contents

In case you are using `write-xml=true` (which is the default) You need to ensure to call the XML writing functions in the correct order. For example after setting the global invoice data, like it's done in the example file. The minimal example below would create a valid XML. The interface commands are described afterwards.

```
\begin{ZUGFeRD}
  \zugferd_write_Item:nnnnnn {1} {} {Plushie \TeX{} lion} {31.98} {2}
  ↪  {63.78}
  \zugferd_startInvoiceSums:
  \zugferd_write_TaxEntry:nnnn {S} {19} {63.78} {12.12}
  \zugferd_write_Summation:nnnnnnnn {63.78} {0} {0} {63.78} {12.12}
  ↪  {75.90} {0} {75.90}
  \zugferd_stopInvoiceSums:
\end{ZUGFeRD}
```

`\zugferd_write_Item:nnnnnn` This command is the interface to write invoice items to the XML file. If the XML interface is enabled this is a reference to the internal command `\__zugferd_insert_TradeLineItem:nnnnnn`.

```
\zugferd_write_Item:nnnnn
{⟨LineID⟩}{⟨optional: item id ("SellerAssignedID")⟩}{⟨item name⟩}
{⟨NetPriceProductTradePrice⟩}
{⟨BilledQuantity⟩}
{⟨LineTotalAmount⟩}
```

Within the product name macros are disabled using `\zugferd_disable_macros:`, see subsection 4.3.

This command is using the local values of tax information as well as the unit code. If you want to overwrite them, adjust them locally using the corresponding options, e.g.:

```
\begingroup
  \keys_set:nn {zugferd}{tax/rate=19, tax/category=S}
  \zugferd_write_Item:nnnnnn {1} {} {Plushie \TeX{} lion} {31.98} {2}
  ↪  {63.78}
  % Code using the data for visible representation
\endgroup
```

This will set the tax rate to 19 % unregarding the global setting.

`\zugferd_startInvoiceSums:`
`\zugferd_stopInvoiceSums:`
There is some global data which is placed in the XML fileafter the invoice items have been placed. Typically, in LaTeX this block is started after the items have been printed and will enclose the summation block.

The starting includes the so called "ApplicableHeaderTradeAgreement" which contains the address data of both trade parties, see subsection 5.2 And this will also print the "SpecifiedTradeSettlementPaymentMeans", see subsubsection 5.2.2.

`\zugferd_write_TaxEntry:nnnn` This command is writing the sum over a tax rate. This command has to be used once per rate applied to the items.

```
\zugferd_write_TaxEntry:nnnn {⟨tax category code⟩} {⟨tax rate in %⟩} {⟨basis
amount the tax applies to⟩} {⟨tax amount⟩}
```

The tax amount could of course be calculated internally. In the example package this is done automatically, but the interface needs to support manual input as a lot of use cases for LaTeX invoicing use it only to create the output file.

`\zugferd_write_Summation:nnnnnnnn`
The total values are all collected with a single macro.

```
\zugferd_write_Summation:nnnnnnnn
{⟨LineTotalAmount⟩}{⟨ChargeTotalAmount⟩}{⟨AllowanceTotalAmount⟩}
{⟨TaxBasisTotalAmount⟩}{⟨TaxTotalAmount⟩}
{⟨GrandTotalAmount⟩}{⟨TotalPrepaidAmount⟩}{⟨DuePayableAmount⟩}
```

This commnd is also writing the payment terms to the XML file. Please be aware that it's in general not possible to calculate the tax values in here, as there might be multiple tax rates applied. This is only taking the sums over all tax entries.

In case you are using some specials like category "E" the exemption reason will also be written at that point. For that it is referencing the current value of the setting.

4.2 Commands to temporary disable/re-enable the XML writing interfaces

`\zugferd_enable_XML_interfaces:`
`\zugferd_disable_XML_interfaces:`

As there are a lot of usecases where code is processed multiple times, it's necessary to provide an interface to temporary disable the XML writing mechanism. A lot of these situations appear within table structures whereas a local adjustment would not be helpful. Therefore these adjustments have to be done globally.

The example package zugferd-invoice provides an example for this to ensure the XML data is not written multiple times. The `ZUGFeRD` environment has been constructed that way, that it would automatically enable the interface when it begins and also when it ends, to write the data. So you should ensure this environment is only processed once or use the lower level interfaces directly. Setting up the catcodes to simplify the XML indentation.

4.3 Escaping macros inside XML data

`\zugferd_disable_macros:` Since we allow the use of LaTeX code in some fields there has to be a mechanism to disable macros inside the XML output. The mechanism is created similar to the one by hyperref, and we also use some definitions from there to use those as a starting point. To have a detailed list of the redefinition, please have a look at the implementation of this command.

There exists a hook to extend or overwrite these definitions `zugferd/disable-macros`. You can add own redefinitions using this. For example if you want to overwrite the setting mapping a `\newline` to a new line char instead of space, you could add the following to your setup:

```
\hook_gput_code:nnn {zugferd/disable-macros}
    {newline-to-LF}
    {\def\newline{\iow_newline:}}
```

5 Adding data to the XML

All data which does not directly depend on amounts or specific items is provided using a key-value interface. For some fields there is the option to define a global preset but locally overwrite it for a specific item. This only applies to data fields used by the writing interfaces described in subsection 4.1.

This package is using the UN/CEFACT Cross Industry Invoice Syntax for the data. Currently it is not planned to implement the UBL syntax as well, but generally this would be possible.

In most cases this functionality will be used to change the tax setting or unit for a single item. subsection 4.1 also provided an example for this.

This section will now take all data which can be set using `\SetZUGFeRDData`.

5.1 General Invoicing Data

Some of the general data currently supports only one value, which is alreay selected by default. The interface already exists and may be extended later.

`document-type=` (`commercial-invoice`) (default: `commercial-invoice`)

Select the document type. The only supported value currently is `commercial-invoice`. This will select the corresponding type code, which is 380.

5.1.1 Invoice number/document ID

`id=` (`komavar`/⟨*document ID/invoice number*⟩) ⟨*initially unset*⟩

This has to be set. Leaving it empty will lead to an invalid XML file.

The value `komavar` would reference the data provided the KOMA-Script letter variabe `invoice`. In case you don't use scrletter you should not use this setting. More information can be found in the documentation [4].

5.1.2 Currency

`currency=` (`EUR`/`USD`/`CHF`/`€`) (default: `EUR`)

Currently zugferd only supports one currency for an invoice. This might be extended later. The currency is pre-configured to use Euro.

5.1.3 Dates

`date=` (`auto`/⟨*date formatted as YYYYMMDD*⟩) (default: `auto`)
`delivery-date=` (`auto`/⟨*date formatted as YYYYMMDD*⟩) (default: `auto`)
`due-date=` ⟨*date formatted as YYYYMMDD*⟩ ⟨*initially unset*⟩

Currently there are three kinds of dates implemented. The XML-Standard requires them to use the structure ⟨*YYYYMMDD*⟩. For the day this document was compiled this would be: "20240911" (September 11, 2024).

Instead of providing a date value directly it's also possible to use `\today`. This is done using the which is the default setting for `date` and `delivery-date`. Please be aware, that this would change if you rebuild the document later. So you might want to use an actual value here.

5.1.4 Payment terms

`payment-terms=` (⟨*string*⟩) ⟨*initially unset*⟩

One option to set payment terms is the `due-date` mentioned before. If this is not set or the setting is more complex one can use `payment-terms` to add more information.

This setting is a string. In case there is expansion required this has to be done before.

5.1.5 Notes: Adding additional information

`subject=` (`komavar`/⟨*Tokenlist*⟩) ⟨*initially unset*⟩
`fromaddress=` (`komavar`/⟨*Tokenlist*⟩) ⟨*initially unset*⟩
`add-note=` ⟨*Tokenlist*⟩ ⟨*initially unset*⟩

The ZUGFeRD example files[10] use all visible data to add them to the XML as a note. `subject` and `fromaddress` are used to support this. The data should not be too relevant but zugferd want's to support adding additional data to the XML using the note element. So these fields can be left out but in case they are not empty, they will also be written to the XML.

The corresponds to the mechanism provided by scrletter. It accesses the variable expands it to be used directly. If you don't use this package, you can ignore this setting or add content manually.

5.2 Trade parties

The XML scheme knows 6 different Trade Parties:

- Seller
- Buyer
- Payee
- ShipTo
- SellerTaxRepresentative

Currently zugferd supports only Buyer, Seller and ShipTo, but can be easily extended to support the others as well. The data for each party follows the same structure, except the “BuyerReference” which is described later in this section.

Some of the data is optional for specific parties. As this also depends on the selected scheme and version we will not list the details. All fields for a trade party can be set using the “group” named by the party. For example setting all the seller data is done in the following listing:

```
\SetZUGFeRDData{
  seller/name = {peiTeX (Marei Peischl)},
  seller/email = {invoicing@peitex.de},
  seller/vatid = {DE123456789},
  seller/contact= {Marei\\+4900000000\\marei@peitex.de},
  seller/address = {Address Line 1\\Address Line 2},
  seller/postcode = {20253},
  seller/city ={Hamburg},
  seller/country = {DE},
}
```

All this data is saved within a property list, which is internally called `\g__zugferd_`⟨*seller/buyer/shipto*⟩`_prop`. By default this property list is empty. The users themselves have to ensure to add the required data.

The outer braces are not required, if the data does not container an equal sign or a comma. In case the final data is unknown, it’s recommended to use them anyway.

⟨*party*⟩`/name=` ⟨*name*⟩ ⟨*initially unset*⟩

⟨*party*⟩`/email=` ⟨*email address*⟩ ⟨*initially unset*⟩

⟨*party*⟩`/vatid=` ⟨*VAT ID*⟩ ⟨*initially unset*⟩

⟨*party*⟩`/address=` ⟨*address*⟩ ⟨*initially unset*⟩

As shown in the example `address` can use two lines separted by `\\`. It’s possble to set all fields for all trade contacts, but e. g. for the `shipto`-party email and vatid will not be used in the XML.

Alternatively it’s also possible to use ⟨*party*⟩`/lineone` and ⟨*party/linetwo*⟩ separately. This may be helpful if you use a custom input format. In any way you should ensure that all macros used within the data either are expandable or disabled using `\zugferd_disable_macros:`.

⟨*party*⟩`/postcode=` ⟨*postal code*⟩ ⟨*initially unset*⟩

⟨*party*⟩`/city=` ⟨*city*⟩ ⟨*initially unset*⟩

⟨*party*⟩`/country=` ⟨*country code*⟩ ⟨*initially unset*⟩

The two letter country codes allowed here can be found in [2].

⟨*party*⟩`/contact=` ⟨*Combined contact data*⟩ ⟨*initially unset*⟩

The contact person can either be set using the combined structure similar to ⟨*party*⟩`/address`. It either consists of 3 or 4 entries, depending on if a department should be used or not.

```
\SetZUGFeRDData{
  seller/contact = {
     |\meta{contact-name}|\\
     |\meta{contact-phone}|\\
     |\meta{contact-email}|
  },
  seller/contact = {
     |\meta{contact-name}|\\
     |\meta{contact-department}|\\
     |\meta{contact-phone}|\\
     |\meta{contact-email}|
  }
}
```

As for `seller/address` it's also possible to set the keys directly:

```
\SetZUGFeRDData{
  seller/contact-name= {|\meta{contact-name}|},
  seller/contact-department = {|\meta{contact-department}|},
  seller/cotact-phone ={|\meta{contact-phone}|},
  seller/contact-email= {|\meta{contact-email}|}
}
```

5.2.1 Buyer Reference

`buyer/reference=` (`komavar/`⟨*Reference*⟩) ⟨*initially unset*⟩

The reference field only exists for the `buyer` trade party. Depending on the process it's required to use some unique identifier referring to the `buyer`. Within Germany these numbers are called “Leitweg-ID”[6].

In any way the `buyer` may choose what is used here. Also may be some PO number or similar reference.

As defined for other variables the `reference` can also use the value to refer to the value of komavar `yourref`[4].

5.2.2 Payment Means

The payment means are selected by numeric codes. Currently we support:

- 1 = Instrument not defined
- 10 = In cash
- 30 = Credit Transfer
- 31 = Debit Transfer
- 42 = Payment to bank account
- 48 = Bank card
- 49 = Direct Debit
- 57 = Standing agreement
- 58 = SEPA credit transfer
- 59 = SEPA direct debit
- 97 = Clearing between partners

Others may be added in the future but it's not planned to include a full list.

The codes will automatically add the corresponding string inside the "Information" field. The initial version only included German strings, but currently they are also included in English. It's possible to overwrite them using the same structure:

```
\setupZUGFeRDStrings{payment-means}{
  10 = Bargeld,
  58 = Zahlung per SEPA Überweisung.,
}
```

The language selection is done using at hook executed at `\begin{document}` and will try to use the document's language. If this is not defined English will be used.

Internally the commands are predefined as a key-value list like the argument in the example above. They macros are called `\zugferd@paymentMeans@`⟨*languagename*⟩. Currently zugferd defines these for `english` and `german` (also `ngerman` as an compatibility alias).

5.3 Variables which may be changed per invoice item

Some settings may have the same value for all invoice items. These are defined to take some preset but are set locally. So it's possible to adjust them for a single invoice item if necessary. An example is shown in subsection 4.1.

5.3.1 Units

`unit=` (`hour/day/one/piece/`⟨*unit code*⟩) ⟨*initially unset*⟩

The Faktur-X standard requires the unit to be selected. These are called "/UN/CEFACT Common Codes" and can be found withtin [8].

Currently zugferd supports `hour` (`HUR`), `day` (`DAY`), `one` (C62) and `piece` (H87). For these the corresponding codes have been implemented within the package. Other units can be selected using the codes listed in [8].

This option is not case sensitive The value is automatically converted to uppercase. If the selected option is different from the predefined ones, there will be a warning, as zugferd does not know if the selection is valid or not.

5.3.2 Tax category and rate

`tax/category=` ⟨*category code/alias*⟩ (default: `standard`)

The Tax data requires a category code. For details have a look at the Specification [e. g. at 5]. zugferd implements all of those, but the user has to take care to select the correct one for each invoice item. The example file includes 2 different VAT values using the same category.

The labels have been chosen to simplify the usage. It's also possible to enter the codes directly. This option is not case sensitive.

`standard` Standard rate and reduced rate item, `category=S`

`zero` Zero rated sale, `category=Z`

`exempt` Exempted from VAT. This requires a reason via `exemption-reason`,`category=E`

`reverse-charge`	Reverse Charge, `category=AE`
`intra-community` or `EEA`	Intra-Community Supply, `category=K`
`export`	Free export item, tax not charged, `category=G`
`O`	Services outside scope of tax
`canary-islands`	Canary Islands general indirect tax, `category=L`
`ceuta` or `melilla`	Ceuta and Melilla, `category=M`

`tax/exemption-reason=` ⟨*Text*⟩ (⟨*initially unset*⟩)
`tax/exemption-reason-code=` ⟨*exemption reason code*⟩ (⟨*initially unset*⟩)

Add Reasons for a tax exempt, as required by `category=E,K,AE,G,O`. This can either be added using a text (`exemption-reason`) or a predefined code (`exemption-reason-code`). The codes are listed at [9].

In most common cases zugferd tries to automatically match them if the package option `auto-exemption` is enabled, which is the default. In that case the following settings would apply:

`S`	Exemption reason: ⟨*empty*⟩; Exemption reason code: ⟨*empty*⟩
`Z`	Not configured.
`E`	Not configured, as there are too many options.
`AE`	Exemption reason: Reverse Charge; Exemption reason code: vatex-eu-ae
`K`	Exemption reason: Intra-Community Supply; Exemption reason code: vatex-eu-ic
`G`	Exemption reason: Export outside the EU; Exemption reason code: vatex-eu-g
`O`	Exemption reason: No subject to VAT; Exemption reason code: vatex-eu-o

In case there is no pre-configured selection zugferd will create a warning to remind the user to add a selection themselves.

`tax/rate=` ⟨*floating point*⟩ (default: `19`)

The value given will be used for tax calculation. By default it's configured to `19` to match the German standard VAT rate.

Change History

v0.6
General: Provide public interfaces and first version of the documentation. 5

v0.8
General: First CTAN version 1

References

[1] URL: https://xeinkauf.de/dokumente/ (visited on 08/20/2024).

[2] *ECE/TRADE/201. ISO COUNTRY CODE for Representation of Names of Countries.* URL: https://unece.org/trade/documents/iso-country-code-representation-names-countries (visited on 08/20/2024).

[3] United Nations Economic Commission for Europe (UNECE). *Code List Recommendations.* URL: https://unece.org/trade/uncefact/cl-recommendations (visited on 08/20/2024).

[4] Markus Kohm. *The scrletter package. Letter extension to KOMA-Script classes.* Version 3.41. July 7, 2023. URL: https://komascript.de/ (visited on 09/03/2024).

[5] Koodinierungsstelle für IT-Standards. *Spezifikation Standard XRechnung. CIUS und Extension.* June 20, 2024. URL: https://xeinkauf.de/app/uploads/2024/07/302-XRechnung-2024-06-20.pdf (visited on 08/20/2024).

[6] Koordinierungsstelle für IT Standards (KoSIT). *Xeinkauf FAQ. Leitweg ID.* URL: https://xeinkauf.de/faq/xrechnung#leitweg-id (visited on 09/04/2024).

[7] *Mustangproject.* URL: https://github.com/ZUGFeRD/mustangproject (visited on 08/20/2024).

[8] *Rec 20 – Codes for Units of Measure Used in International Trade.* Link directly to xlsx. [see 3, for all revisions]. URL: https://unece.org/sites/default/files/2023-10/rec20_Rev17e-2021.xlsx (visited on 08/20/2024).

[9] *VAT exemption reason code list.* URL: https://www.xrepository.de/details/urn:xoev-de:kosit:codeliste:vatex_1 (visited on 08/20/2024).

[10] *ZUGFeRD 2.2. Download page.* URL: https://ferd-net.de/standards/zugferd-2.2/zugferd-2.2.html (visited on 09/02/2024).

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.