

# Mustang project user documentation

Jochen Stärk

For Mustangproject 1.1.2, 2015-06-15

<http://www.mustangproject.org>

## Inhaltsverzeichnis

Mustang project user documentation.....	1
About Mustangproject.....	1
Overview of ZUGFeRD-Solutions.....	1
Download/Project setup.....	2
Source code.....	2
Project setup without Maven.....	2
With Maven.....	3
Reading ZUGFeRD data.....	3
Complete sample source code for reading ZUGFeRD data.....	3
Writing a ZUGFeRD-PDF file.....	4
Complete source code example for writing ZUGFeRD PDFs.....	7
Writing custom XML-Data.....	11
Supplementary functions.....	11

## About Mustangproject

Mustangproject is a Java-Library for extended („ZUGFeRD“-)metadata in PDF-invoices. It requires the Apache PDFBox library, uses PDF/A files as input and is, like Apache PDFBox subject to the APL-License and can therefore, within the terms of the Apache Public License, be used for free in commercial and noncommercial projects as long as e.g. a according „Notice“-file is placed.

## Overview of ZUGFeRD-Solutions

	Platform	License	Functionality				Viable for			Price
			Read PDF	create XML	write PDF	PDF/A-Conversion	Commercial software	Freeware	Open Source	
intarsys	Java	proprietary	Yes	Yes	Yes	Yes	Yes	Yes	No	On request
Konik	Java	AGPL	Yes	Yes	Yes	No	No	No	Yes	0 €
Mustang	Java	APL	Yes	Yes	Yes	No	Yes	Yes	Yes	0 €

<a href="https://github.com/stephanstapel/ZUGFeRD-csharp">https://github.com/stephanstapel/ZUGFeRD-csharp</a>	C#	APL	Yes	Yes	No	No	Yes	Yes	Yes	0 €
---	----	-----	-----	-----	----	----	-----	-----	-----	-----

## ***Download/Project setup***

### **Source code**

Home of the Mustangprojekt source code is <https://github.com/Rayman2200/PDFA3>

### **Project setup without Maven**

With installed OpenOffice.org or LibreOffice and Eclipse for Java.

1. Start Eclipse, create a new Java-Eclipse-project, e.g. „MustangSample“.
2. Change to that folder.
3. Download
  1. Apache PDFBox
    1. from <http://apache.openmirror.de/pdfbox/1.8.8/pdfbox-1.8.8.jar>
    2. from <http://apache.openmirror.de/pdfbox/1.8.8/preflight-app-1.8.8.jar>
    3. from <http://apache.openmirror.de/pdfbox/1.8.8/xmpbox-1.8.8.jar>
  2. Mustang
    1. the JAR file <http://mustangproject.org/deploy/mustang-1.1.2.pdf>
    2. the notice file <http://mustangproject.org/deploy/NOTICE>
  3. Download the sample
    1. from [http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20150613\\_503.pdf](http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20150613_503.pdf)
    2. Either
      1. Download [http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20140613\\_503.pdf](http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20140613_503.pdf)
      2. Open this OpenOffice.org source file in Writer
      3. File|Export as PDF: Set the Checkbox PDF/A-1a in the export options
      4. Save the PDF-Datei as „[MustangGnuaccountingBeispielRE-20150613\\_503blanko.pdf](http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20150613_503blanko.pdf)“

### 3. alternatively

1. download blank PDF without ZUGFeRD metadata from [http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20150613\\_503blanko.pdf](http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20150613_503blanko.pdf)
4. Switch back to Eclipse. Add all four downloaded JAR files to your project (right click on project name, Properties) add as „external Jar“ to the „Build Path“ in the „libraries“ tab.

## With Maven

The following repository

```
<repositories>
  <repository>
    <id>mustang-mvn-repo</id>
    <url>https://raw.githubusercontent.com/Rayman2200/PDFA3/mvn-repo/</url>
  </repository>
</repositories>
```

serves the following dependency

```
<dependency>
  <groupId>org.mustangproject.ZUGFeRD</groupId>
  <artifactId>mustang</artifactId>
  <version>1.1.2</version>
</dependency>
```

## Reading ZUGFeRD data

5. Create a new class in the src folder, called Reader. Check the „Public static void main()“ checkbox.
6. Within the main method, enter „ZUGFeRDImporter zi=**new** ZUGFeRDImporter();“ and add the import by pressing STRG+SHIFT+O
7. use zi.extract(PDF-filename) and canParse() to find out if ZUGFeRD-Data is present.
8. After invoking zi.parse() you can access the getter-Methods like getAmount()
9. There are only getters for few properties but additional ones can be added easily. Which data is available can be seen in the ZUGFeRD-invoice.xml file embedded any ZUGFeRD compliant PDF

## Complete sample source code for reading ZUGFeRD data

```
package sample;

import org.mustangproject.ZUGFeRD.ZUGFeRDImporter;

public class Read {
```

```

public static void main(String[] args) {
    ZUGFeRDImporter zi=new ZUGFeRDImporter();
    zi.extract("./MustangGnuaccountingBeispielRE-20150613_503.pdf");
    System.out.println("Reading ZUGFeRD");
    if (zi.canParse()) {
        zi.parse();
        System.out.println("Due amount:"+zi.getAmount());
        System.out.println("BIC:"+zi.getBIC());
        System.out.println("IBAN:"+zi.getIBAN());
        System.out.println("Account holder name:"+zi.getHolder());
        System.out.println("Document:"+zi.getForeignReference());
    }
}
}
}

```

## Writing a ZUGFeRD-PDF file

A sample for writing ZUGFeRD PDFs is more comprehensive, because

- 1) more data is being written than read in the read example and
- 2) the exporter interacts via interfaces with your software in a kind of „pull-method“. While this avoids redundant data a sample is more exhaustive because the sample has to store the data in the memory, which any productive software already does.

The alternative ZUGFeRD-Open-Source-project Konik (<http://konik.io>) follows a more conventional „push-method“ in which data is stored redundantly (if used alongside a ordinary software) by using setter-methods but which conveniently does not require you to cater for the availability of the getter methods.

1. Create a new class in the src-folder, e.g. MustangWriter. Check the checkbox to generate „Public static void main()“.
2. Change **public class** MustangWriter to **public class** MustangWriter **implements** IZUGFeRDExportableTransaction
3. Add the following classes in in the same file:
  1. add **class** Contact **implements** IZUGFeRDExportableContact {}
  2. **class** Item **implements** IZUGFeRDExportableItem {
 

```

          private BigDecimal price, priceGross, quantity, totalGross;
          private Product product;
          }
          
```
  3. **class** Product **implements** IZUGFeRDExportableProduct {
 

```

          private String description, name, unit;
          private BigDecimal VATPercent;
          }
          
```
4. Generate the imports by pressing CTRL+SHIFT+O

5. Click left on MustangWriter and press ALT+SHIFT+S, select Override/Implement Methods and press return.
6. Click on Contact and repeat the last step.
7. Click Item, mark the variables, press ALT+SHIFT+S and select „Generate Getters and Setters“. Mark all members and press return.
8. Click again on Item, press ALT+SHIFT+S and select „Generate Constructor using Fields“. Choose again all member variables and press return.
9. Repeat the last two steps for „Product“: Click Product, mark the variables, press ALT+SHIFT+S and select „Generate Getters and Setters“. Choose all members and press return.
10. Click on Product again, press ALT+SHIFT+S and select „Generate Constructor using Fields“. Choose all members again and press return.
11. The following methods of Contact should return the following:

```

1. getCountry(): "DE"
2. getLocation(): "Spielkreis"
3. getName(): "Theodor Est"
4. getStreet(): "Bahnstr. 42"
5. getVATID(): "DE999999999"
6. getZIP(): "88802";

```

12. The following methods of the main class should return the following:

```

1. getDeliveryDate(): new GregorianCalendar(2014,Calendar.JULY,3).getTime()
2. CTRL+SHIFT+O will import the necessary GregorianCalendar class
3. getDueDate(): new GregorianCalendar(2014,Calendar.JULY,24).getTime()
4. getIssueDate(): new GregorianCalendar(2014,Calendar.JULY,3).getTime()
5. getNumber(): "RE-20140703/502"
6. getOwnBIC(): "COBADEFXXX"
7. getOwnBankName(): "Commerzbank"
8. getOwnCountry() "DE"
9. getOwnIBAN(): "DE88 2008 0000 0970 3757 00"
10. getOwnLocation() "Stadthausen"
11. getOwnOrganisationName(): "Bei Spiel GmbH"
12. getOwnStreet() "Ecke 12"
13. getOwnTaxID(): "22/815/0815/4"
14. getOwnVATID(): "DE136695976"
15. getOwnZIP() "12345"
16. getRecipient(): new Contact()
17. getTotal(): new BigDecimal("496.00")
18. getTotalGross(): new BigDecimal("571.04")

```

19. getZFItems() of the main class can now create products and return them as a array of items:

```

Item[] allItems=new Item[3];
Product designProduct=new Product("", "Künstlerische Gestaltung
(Stunde)", "HUR", new BigDecimal("7.000000"));
Product balloonProduct=new Product("", "Luftballon", "C62", new
BigDecimal("19.000000"));

```

```

        Product airProduct=new Product("", "Heiße Luft pro Liter", "LTR", new
BigDecimal("19.000000"));

        allItems[0]=new Item(new BigDecimal("160"), new BigDecimal("171.20"),
new BigDecimal("1"), new BigDecimal("171.20"), designProduct);
        allItems[1]=new Item(new BigDecimal("0.79"), new BigDecimal("0.94"),
new BigDecimal("400"), new BigDecimal("376.04"), balloonProduct);
        allItems[2]=new Item(new BigDecimal("0.10"), new BigDecimal("0.12"),
new BigDecimal("200"), new BigDecimal("23.80"), airProduct);
        return allItems;

```

20. Now create a private void apply method

21. Please instantiate this main MustangWriter class in the main method and invoke the apply() function.

22. In the apply-method you can now

1. load a PDDocument
2. instantiate a ZUGFeRDEExporter ,
3. invoke the ZUGFeRDEExporter's PDFmakeA3compliant (including the „Producer“, i.e. Application- and „Creator“ ,i.e. Author name parameters) and
4. finally use the PDFattachZugferdFile-method (with the IZUGFeRDEExportableTransation, i.e. „this“ as parameter) and
5. save the PDDocument again. The apply-method then looks – with according try/catch-blocks- as follows:

```

PDDocument doc;

try {
    System.out.println("Reading blank PDF");
    doc = PDDocument.load("./MustangGnuaccountingBeispielRE-
20150613_503blanko.pdf");
    // automatically add Zugferd to all outgoing invoices
    ZUGFeRDEExporter ze = new ZUGFeRDEExporter();
    System.out.println("Converting to PDF/A-3u");
    ze.PDFmakeA3compliant(doc, "My Application",
        System.getProperty("user.name"), true);
    System.out.println("Generating and attaching ZUGFeRD-Data");
    ze.PDFattachZugferdFile(doc, this);
    System.out.println("Writing ZUGFeRD-PDF");
    doc.save("./MustangGnuaccountingBeispielRE-20150613_503new.pdf");
    System.out.println("Done.");
} catch (IOException e) {
    e.printStackTrace();
} catch (TransformerException e) {
    e.printStackTrace();
} catch (COSVisitorException e) {
    e.printStackTrace();
}

```

23. CTRL+SHIFT+O again helps with the imports

24. „My Application“ and System.getProperty("user.name") are stored in the meta

data as „Producer“ (producing application) respectively „Creator“ (author). Please adjust accordingly.

25. Adjust the NOTICE-File and add it to your application.

## Complete source code example for writing ZUGFeRD PDFs

```
import java.io.IOException;
import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;

import javax.xml.transform.TransformerException;

import org.apache.pdfbox.exceptions.COSVisitorException;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.mustangproject.ZUGFeRD.IZUGFeRDExportableContact;
import org.mustangproject.ZUGFeRD.IZUGFeRDExportableItem;
import org.mustangproject.ZUGFeRD.IZUGFeRDExportableProduct;
import org.mustangproject.ZUGFeRD.IZUGFeRDExportableTransaction;
import org.mustangproject.ZUGFeRD.ZUGFeRDExporter;

public class MustangWriter implements IZUGFeRDExportableTransaction {

    @Override
    public Date getDeliveryDate() {
        return new GregorianCalendar(2014, Calendar.JULY, 3).getTime();
    }

    @Override
    public Date getDueDate() {
        return new GregorianCalendar(2014, Calendar.JULY, 24).getTime();
    }

    @Override
    public Date getIssueDate() {
        return new GregorianCalendar(2014, Calendar.JULY, 3).getTime();
    }

    @Override
    public String getNumber() {
        return "RE-20140703/502";
    }

    @Override
    public String getOwnBIC() {
        return "COBADEFXXX";
    }

    @Override
    public String getOwnBankName() {
        return "Commerzbank";
    }

    @Override
    public String getOwnCountry() {
        return "DE";
    }

    @Override
    public String getOwnIBAN() {
        return "DE88 2008 0000 0970 3757 00";
    }
}
```

```

@Override
public String getOwnLocation() {
    return "Stadthausen";
}

@Override
public String getOwnOrganisationName() {
    return "Bei Spiel GmbH";
}

@Override
public String getOwnStreet() {
    return "Ecke 12";
}

@Override
public String getOwnTaxID() {
    return "22/815/0815/4";
}

@Override
public String getOwnVATID() {
    return "DE136695976";
}

@Override
public String getOwnZIP() {
    return "12345";
}

@Override
public IZUGFeRDEExportableContact getRecipient() {
    return new Contact();
}

@Override
public BigDecimal getTotal() {
    return new BigDecimal("496.00");
}

@Override
public BigDecimal getTotalGross() {
    return new BigDecimal("571.04");
}

@Override
public IZUGFeRDEExportableItem[] getZFItems() {
    Item[] allItems=new Item[3];
    Product designProduct=new Product("", "Künstlerische Gestaltung (Stunde)", "HUR", new
BigDecimal("7.000000"));
    Product balloonProduct=new Product("", "Luftballon", "C62", new BigDecimal("19.000000"));
    Product airProduct=new Product("", "Heiße Luft pro Liter", "LTR", new
BigDecimal("19.000000"));

    allItems[0]=new Item(new BigDecimal("160"), new BigDecimal("171.20"), new
BigDecimal("1"), new BigDecimal("171.20"), designProduct);
    allItems[1]=new Item(new BigDecimal("0.79"), new BigDecimal("0.94"), new
BigDecimal("400"), new BigDecimal("376.04"), balloonProduct);
    allItems[2]=new Item(new BigDecimal("0.10"), new BigDecimal("0.12"), new
BigDecimal("200"), new BigDecimal("23.80"), airProduct);
    return allItems;
}

class Contact implements IZUGFeRDEExportableContact {

    @Override
    public String getCountry() {
        return "DE";
    }

    @Override

```



```

    public String getLocation() {
        return "Spielkreis";
    }

    @Override
    public String getName() {
        return "Theodor Est";
    }

    @Override
    public String getStreet() {
        return "Bahnstr. 42";
    }

    @Override
    public String getVATID() {
        return "DE999999999";
    }

    @Override
    public String getZIP() {
        return "88802";
    }
}

class Item implements IZUGFeRDEExportableItem {

    public Item(BigDecimal price, BigDecimal priceGross,
                BigDecimal quantity, BigDecimal totalGross, Product product) {
        super();
        this.price = price;
        this.priceGross = priceGross;
        this.quantity = quantity;
        this.totalGross = totalGross;
        this.product = product;
    }

    private BigDecimal price, priceGross, quantity, totalGross;
    private Product product;

    public BigDecimal getPrice() {
        return price;
    }

    public void setPrice(BigDecimal price) {
        this.price = price;
    }

    public BigDecimal getPriceGross() {
        return priceGross;
    }

    public void setPriceGross(BigDecimal priceGross) {
        this.priceGross = priceGross;
    }

    public BigDecimal getQuantity() {
        return quantity;
    }

    public void setQuantity(BigDecimal quantity) {
        this.quantity = quantity;
    }

    public BigDecimal getTotalGross() {
        return totalGross;
    }

    public void setTotalGross(BigDecimal totalGross) {
        this.totalGross = totalGross;
    }
}

```

```

        public Product getProduct() {
            return product;
        }

        public void setProduct(Product product) {
            this.product = product;
        }
    }

    class Product implements IZUGFeRExportableProduct {
        private String description, name, unit;
        private BigDecimal VATPercent;

        public Product(String description, String name, String unit,
            BigDecimal VATPercent) {
            super();
            this.description = description;
            this.name = name;
            this.unit = unit;
            this.VATPercent = VATPercent;
        }

        public String getDescription() {
            return description;
        }

        public void setDescription(String description) {
            this.description = description;
        }

        public String getName() {
            return name;
        }

        public void setName(String name) {
            this.name = name;
        }

        public String getUnit() {
            return unit;
        }

        public void setUnit(String unit) {
            this.unit = unit;
        }

        public BigDecimal getVATPercent() {
            return VATPercent;
        }

        public void setVATPercent(BigDecimal vATPercent) {
            VATPercent = vATPercent;
        }
    }

    private void apply() {
        PDDocument doc;
        try {
            System.out.println("Reading blank PDF");
            doc = PDDocument.load("./MustangGnuaccountingBeispielRE-20150513_503blanko.pdf");
            // automatically add Zugferd to all outgoing invoices
            ZUGFeRExporter ze = new ZUGFeRExporter();
            System.out.println("Converting to PDF/A-3u um");
            ze.PDFmakeA3compliant(doc, "My Application",
                System.getProperty("user.name"), true);
            System.out.println("Generating and attaching ZUGFeRD-Data");
            ze.PDFattachZugferdFile(doc, this);
            System.out.println("Writing ZUGFeRD-PDF");
        }
    }

```

```

        doc.save("./MustangGnuaccountingBeispielRE-20150613_503new.pdf");
        System.out.println("Done.");
    } catch (IOException e) {
        e.printStackTrace();
    } catch (TransformerException e) {
        e.printStackTrace();
    } catch (COSVisitorException e) {
        e.printStackTrace();
    }
}

}

public static void main(String[] args) {
    MustangWriter mw=new MustangWriter();
    mw.apply();
}
}

```

## Writing custom XML-Data

If you create your own ZUGFeRD-XML you can attach them using `setZUGFeRDXMLData`, in this case `PDFattachZugferdFile` is invoked with a null argument as follows:

```

doc = PDDocument.load("./Source.pdf");
// automatically add Zugferd to all outgoing invoices
ZUGFeRDExporter ze = new ZUGFeRDExporter();
System.out.println("Converting to PDF/A-3u");
ze.PDFmakeA3compliant(doc, "My Application",
    System.getProperty("user.name"), true);
System.out.println("Attaching ZUGFeRD-Data");
String ownZUGFeRDXML = "<some><xml attrib='value'/></some>";
ze.setZUGFeRDXMLData(ownZUGFeRDXML.getBytes());
ze.PDFattachZugferdFile(doc, null);
System.out.println("Writing ZUGFeRD-PDF");
doc.save("./Target.pdf");

```

## Supplementary functions

`zugferdExporter->setTest()` sets the indicator in the xml structure that the invoice has been created in or is ment for nonproductive use only.