Nuclet: Receiving electronic invoices - XRechnung and ZUGFeRD

Language: Deutsch · **English**

Nuclet: Receiving electronic invoices - XRechnung and ZUGFeRD

> **Data sheet and documentation entry point for the Electronic invoices Nuclet (receiving and processing electronic invoices (XRechnung and ZUGFeRD)).**

CONCEPT | INTEGRATORS | UPDATED: JUL 2026 | APPLIES TO NUCLOS 4.2026.X

On this page

Machine-translated – under review

This page was machine-translated from German as a first draft and is currently under review. The authoritative source remains the **German original** (see the language switch above).

- Nuclet: Receiving electronic invoices - XRechnung and ZUGFeRD
  - Releases
  - Overview
  - Notes
  - Integration
  - Related pages

Releases

Version	Date	Notes	Compatibility	Note
2.1.2	03.12.2024	First published version	from Nuclos 4.2022.41.2	
2.1.3	09.01.2025	Minor layout changes and moved to menu *Interfaces* adaptation to export version	from Nuclos 4.2022.41.2	
2.1.4	07.02.2025	Standalone import without *Import e-invoice* enabled	from Nuclos 4.2022.41.2	
2.1.5	29.07.2025	adaptation to export version	from Nuclos 4.2022.41.2	
2.1.6	10.10.2025	adaptation to export version	from Nuclos 4.2022.41.2	

Overview

The Nuclet enables the automated receipt of electronic invoices (e-invoices) in XML or PDF format, their official validation and presentation in an HTML file as well as the preparation for conversion into a suitable business object (hereafter: incoming invoice).

The following format is supported: XRechnung in version 3.0.2 (20.06.2024) in the XML syntax Universal Business Language (UBL) of OASIS in version 2.1 and the XML syntax Cross Industry Invoice (CII) of UN/CEFACT in version D16B.
Furthermore, the specification ZUGFeRD is covered with its five profiles in version 2.3.2 (15.11.2024).

The receipt of electronic invoices in outdated versions of the mentioned formats is also possible in many cases, but usually results in additional validation messages.

E-invoices can be imported automatically via the job *Importiere ERechnungen* from a specified directory (Nuclet parameter *Ablagepfad*).

- In doing so, a business object *ERechnungsimport* is created with corresponding messages. After that - if possible - an incoming invoice is created from the data.

- The files are then moved to corresponding subfolders so that they are not processed multiple times.

E-invoices can also be imported manually via the form *Interfaces -> Import e-invoice* and converted into an incoming invoice.

The example Nuclet under Nuclet: Electronic invoices - XRechnung and ZUGFeRD can also be used here.

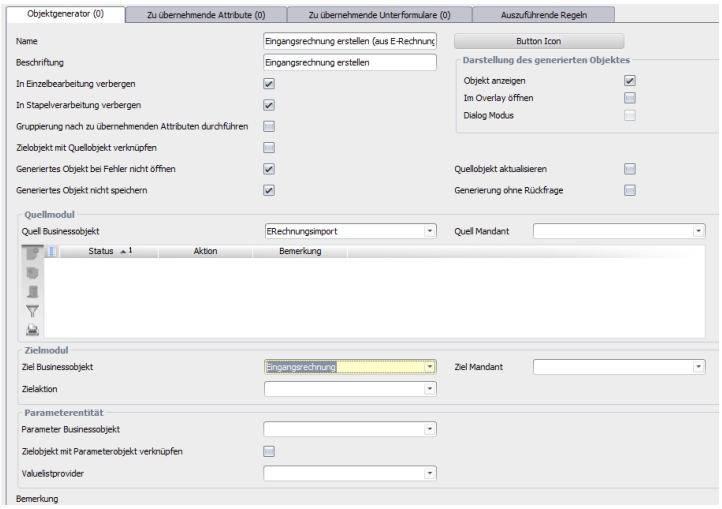
Notes

> **Dependencies**
>
> The Nuclet requires the two Nuclets described under Nuclet: Electronic invoices - XRechnung and ZUGFeRD .

1. To create incoming invoices, a project-specific object generator rule must be created that handles the assignment to the fields of the incoming invoice. At this point, the interface `org.nuclet.xrechnung.imp.ImportiereERechnungen.InvoiceHelper` can also be implemented (the assignment takes place via the Nuclet parameter *ImplKlasseEingangsrechnung*), which provides the methods required in the job for validating uniqueness (`getEingangsrechnung`), creating a new incoming invoice (`newEingangsrechnung`) and assigning the fields (`setFields`). A complete example is given under Integration .
2. Via the method `org.nuclet.xrechnung.imp.AktualisiereERechnungsimport#getXRechnung` the general Java object based on the XRechnung standard is made available. If fields are required that only exist in one of the two above-mentioned XML syntaxes, the respective complete Java transformations can be obtained via `org.nuclet.xrechnung.imp.AktualisiereERechnungsimport#getCompleteInvoice` .

Integration

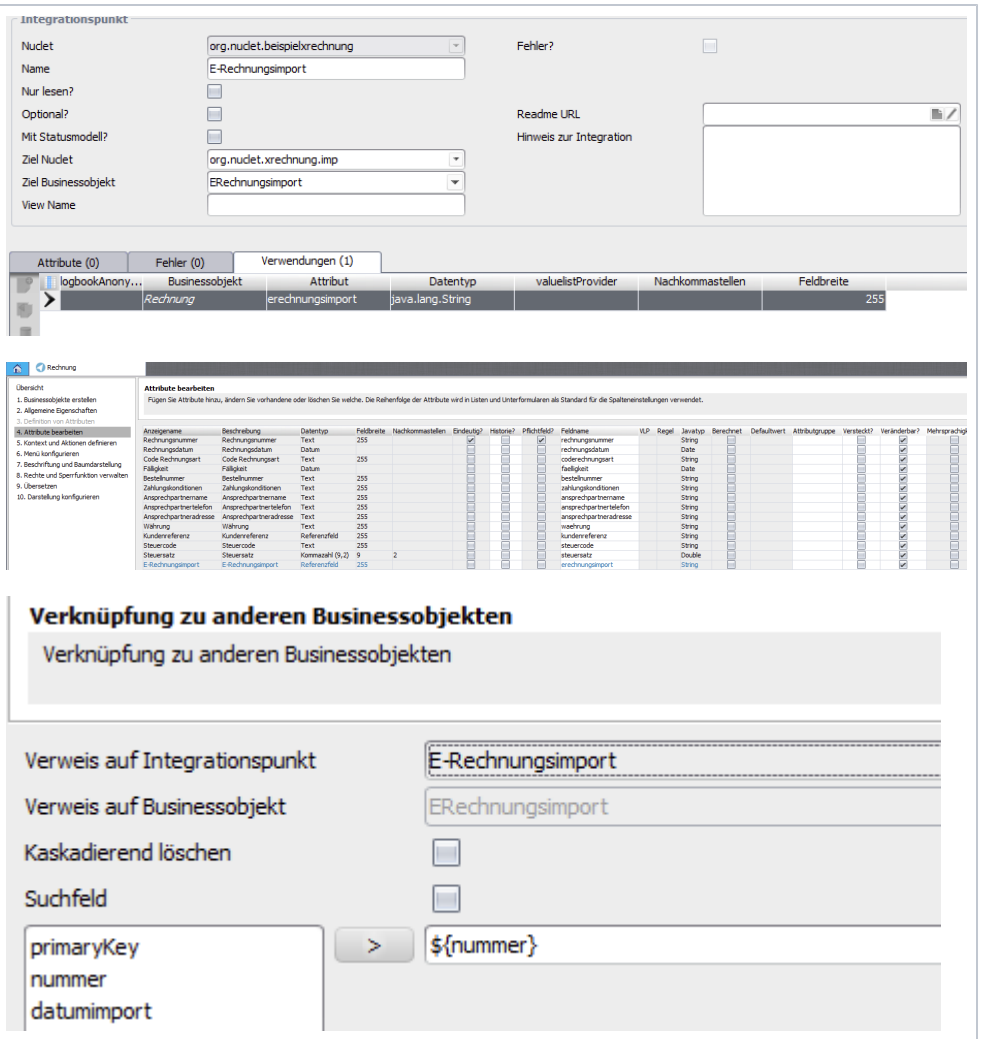
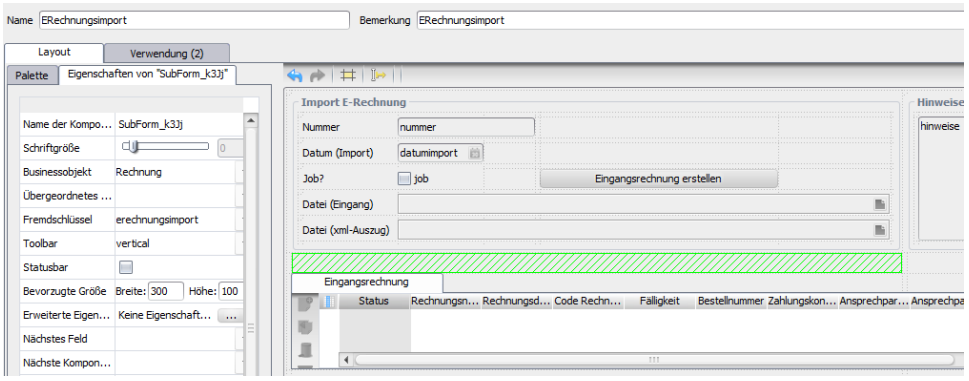
Step	Description	Screenshot / description
1	If not already done, import the two Nuclets described under Nuclet: Electronic invoices - XRechnung and ZUGFeRD (see Nuclet import). After importing these Nuclets, the server instance must be restarted.	
2	Carry out the import of the Nuclet, then the server instance must be restarted again, as the Nuclet contains its own extensions.	

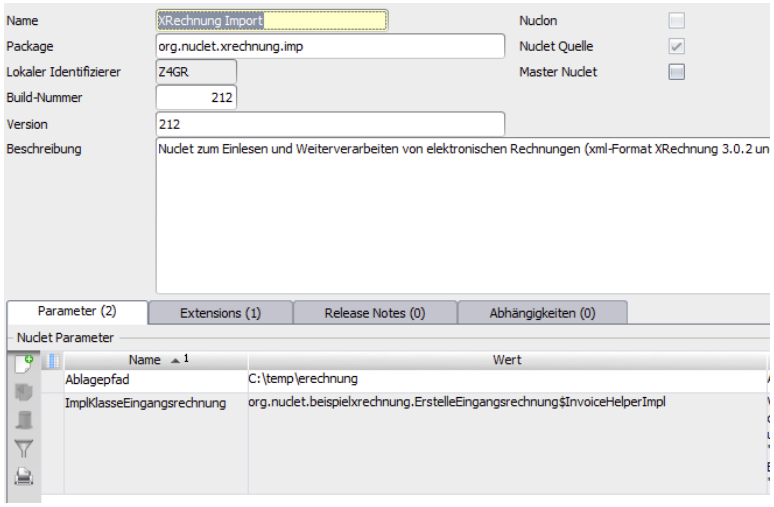
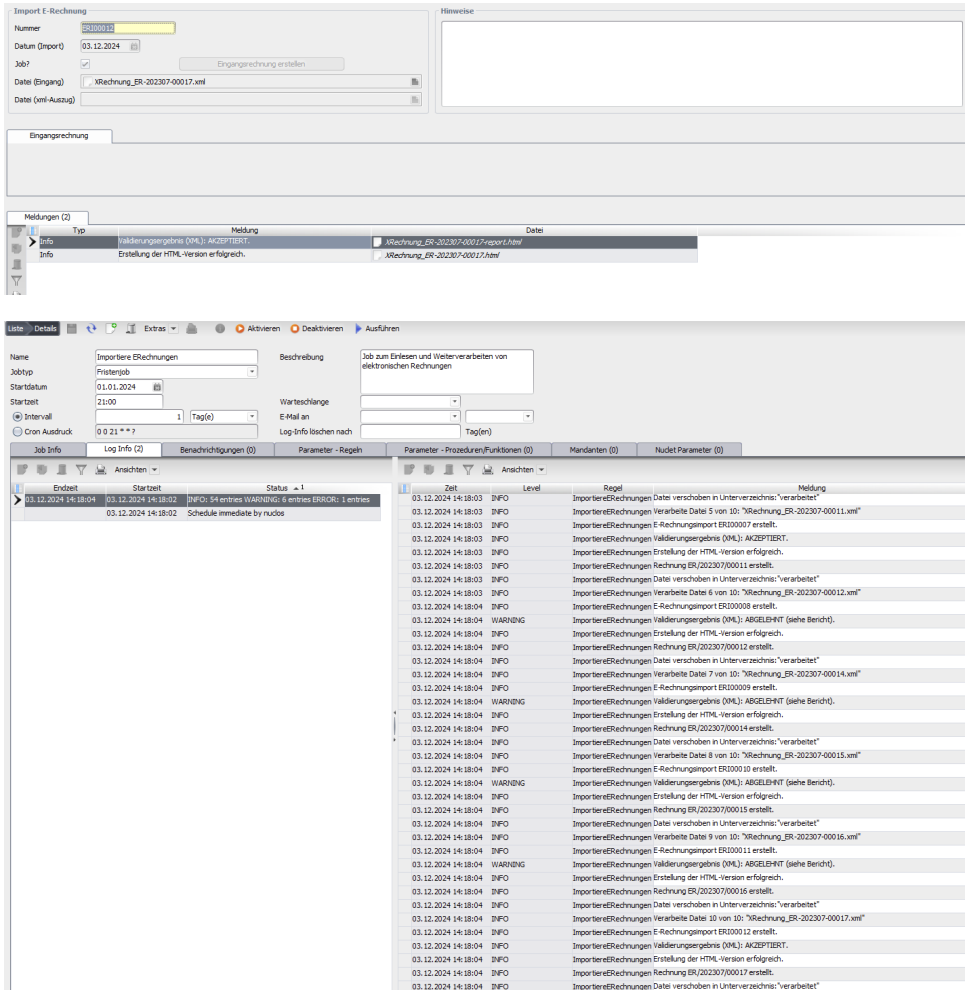
3	Create an object generator for the incoming invoices with a corresponding rule.	
		ErstelleEingangsrechnung.java
		`package org.nuclet.beispielxrechnung;`
		
		`import eu.ce.en16931._2017.xoev_de.kosit.standard.xrechnung_1.*;`
		`import org.nuclet.xrechnung.imp.AktualisiereERechnungsimport;`

```
import org.nuclet.xrechnung.imp.ERechnungsimport;
import org.nuclet.xrechnung.imp.ImportiereERechnungen;
import org.nuclos.api.businessobject.Flag;
import org.nuclos.api.businessobject.facade.Modifiable;
import org.nuclos.api.provider.QueryProvider;
import org.nuclos.api.rule.GenerateRule;
import org.nuclos.api.context.GenerateContext;
import org.nuclos.api.annotation.Rule;
import org.nuclos.api.exception.BusinessException;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import static org.nuclet.xrechnung.imp.ImportiereERechnungen.*;
import static org.nuclet.xrechnung.imp.ImportiereERechnungen.getTextValue;

/** @name
  * @description
  * @usage
  * @change
*/
@Rule(name="ErstelleEingangsrechnung", description="
ErstelleEingangsrechnung")
public class ErstelleEingangsrechnung implements GenerateRule {

        public void generate(GenerateContext context) throws
BusinessException {
                ERechnungsimport imp = context.getSourceObject
(ERechnungsimport.class);
                Rechnung rechnung = context.getTargetObject(Rechnung.class);
                setFields(imp, rechnung);
        }

        public static void setFields(ERechnungsimport imp, Rechnung
rechnung) throws BusinessException {
                AktualisiereERechnungsimport.checkUpdate(imp, "Zu diesem
Import wurde bereits eine Eingangsrechnung erstellt.");
                Invoice xRechnung = AktualisiereERechnungsimport.getXRechnung
(imp);
                rechnung.setErechnungsimportId(imp.getId());
                rechnung.setRechnungsnummer(xRechnung.getInvoiceNumber().
getValue());
                rechnung.setRechnungsdatum(getDate(xRechnung.
getInvoiceIssueDate()));
        rechnung.setCoderechnungsart(getCodeValue(xRechnung.
getInvoiceTypeCode()));
                rechnung.setFaelligkeit(getDate(xRechnung.
getPaymentDueDate()));
                DocumentReference purchaseOrderReference = xRechnung.
getPurchaseOrderReference();
                if (purchaseOrderReference != null) {
                        rechnung.setBestellnummer(purchaseOrderReference.
getValue());
                }
                rechnung.setZahlungskonditionen(getTextValue(xRechnung.
getPaymentTerms()));
                SELLERType seller = xRechnung.getSELLER();
        if (seller != null) {
                        SELLERCONTACTType sellerContact = seller.
getSELLERCONTACT();
            if (sellerContact != null) {
                                rechnung.setAnsprechpartnername(getTextValue
(sellerContact.getSellerContactPoint()));
                                rechnung.setAnsprechpartnertelefon
(getTextValue(sellerContact.getSellerContactTelephoneNumber()));
                                rechnung.setAnsprechpartneradresse
(getTextValue(sellerContact.getSellerContactEmailAddress()));
                        }
                        String ustIdNr = getIdValue(seller.
getSellerVATIdentifier());
```

```
            if (ustIdNr != null) {
                                List<Kunde> kundeList = QueryProvider.execute
(QueryProvider.create(Kunde.class).where(Kunde.Ustidnr.eq(ustIdNr)).
setChunkSize(2L));
                                int size = kundeList.size();
                                Long kundeId = null;
                                if (size == 1) {
                    kundeId = kundeList.iterator().next().getId();
                } else if (size == 0) {
                                        Kunde kunde = new Kunde();
                                        kunde.setName(getTextValue(seller.
getSellerName()));
                                        kunde.setLeitweg(getTextValue
(xRechnung.getBuyerReference()));
                                        SELLERPOSTALADDRESSType
sellerpostaladdress = seller.getSELLERPOSTALADDRESS();
                    if (sellerpostaladdress != null) {
                                                kunde.setStrasse(getTextValue
(sellerpostaladdress.getSellerAddressLine1()));
                                                kunde.setPlz(getTextValue
(sellerpostaladdress.getSellerPostCode()));
                                                kunde.setOrt(getTextValue
(sellerpostaladdress.getSellerCity()));
                                                kunde.setLaendercode
(getCodeValue(sellerpostaladdress.getSellerCountryCode()));
                                        }
                                        kunde.setUstidnr(ustIdNr);
                                        kunde.save();
                                        kundeId = kunde.getId();
                                }
                if (kundeId != null) {
                                        rechnung.setKundenreferenzId
(kundeId);
                }
                        }
                }
                rechnung.setWaehrung(getCodeValue(xRechnung.
getInvoiceCurrencyCode()));
                List<VATBREAKDOWNType> vatbreakdown = xRechnung.
getVATBREAKDOWN();
        if (vatbreakdown.size() == 1) {
                        VATBREAKDOWNType breakdown = vatbreakdown.iterator().
next();
                        rechnung.setSteuercode(getCodeValue(breakdown.
getVATCategoryCode()));
                        rechnung.setSteuersatz(getBigDecimal(breakdown.
getVATCategoryRate()));
        }
                for (INVOICELINEType line : xRechnung.getINVOICELINE()) {
                        Rechnungsposition pos = new Rechnungsposition();
                        ITEMINFORMATIONType iteminformation = line.
getITEMINFORMATION();
            if (iteminformation != null) {
                                pos.setBezeichnung(getTextValue
(iteminformation.getItemName()));
                        }
                        pos.setMenge(getBigDecimal(line.
getInvoicedQuantity()));
                        pos.setEinheit(getCodeValue(line.
getInvoicedQuantityUnitOfMeasureCode()));
                        pos.setNettobetrag(getBigDecimal(line.
getInvoiceLineNetAmount()));
                        pos.setBestellposition(getIdValue(line.
getInvoiceLineIdentifier()));
                        INVOICELINEPERIODType period = line.
getINVOICELINEPERIOD();
            if (period != null) {
                                pos.setLeistungszeitraumvon(getDate(period.
getInvoiceLinePeriodStartDate()));
                                pos.setLeistungszeitraumbis(getDate(period.
getInvoiceLinePeriodEndDate()));
```

```
                        }
                        rechnung.insertRechnungsposition(pos);
                }
        }

        public static class InvoiceHelperImpl implements InvoiceHelper {

                public Modifiable<?> newEingangsrechnung() {
                        return new Rechnung();
                }

                public List<? extends Modifiable<?>> getEingangsrechnung
(ERechnungsimport imp, Flag... flags) {
                        List<Rechnung> list = new ArrayList<>();
                        org.nuclet.beispielxrechnung.ERechnungsimport
rechnungsimport = org.nuclet.beispielxrechnung.ERechnungsimport.get(imp.
getId());
            if (rechnungsimport != null) {
                list = rechnungsimport.getRechnung(flags);
            }
                        return list;
                }

                public void setFields(ERechnungsimport imp, Modifiable<?>
eingangsrechnung) throws BusinessException {
                        ErstelleEingangsrechnung.setFields(imp, (Rechnung)
eingangsrechnung);
                }
        }
}
```

4	If needed, create an integration point in the incoming invoice Nuclet with a reference to the *ERechnungsimport* and create a reference field via the BO editor.	
5	If needed, extend the layout *ERechnungsimport* . • Embed the subform for the incoming invoice • Embed the object generator via a button (alternatively only via the menu without layout change)	

6	Set the Nuclet parameters appropriately.	
7	Check the functionality of the job and the manual import with suitable examples (e.g. xrechnung-testsuite/.../technical-cases or zugferd-2.3.2).	

Related pages

Interfaces

Back to Interfaces

Open

Nuclet Wiki

Nuclet Wiki home

Open