readme.md for @fin.cx/einvoice

The Ultimate TypeScript E-Invoicing Library for Europe - Now with **100% EN16931 Compliance** □

TypeScript EN16931 Standards License

Transform the chaos of European e-invoicing into pure TypeScript elegance. **@fin.cx/einvoice** is your battle-tested solution for creating, validating, and converting electronic invoices across all major European standards - with blazing fast performance and enterprise-grade reliability.

□□ Why @fin.cx/einvoice?

- □□ **100% EN16931 Compliant**: Full implementation of all 162 Business Terms and 32 Business Groups
- ⚡ **Blazing Fast**: Validate invoices in ~2.2ms, convert formats in ~0.6ms
- □□ **Enterprise Security**: XXE prevention, resource limits, path traversal protection
- □□ **Multi-Standard Support**: ZUGFeRD, Factur-X, XRechnung, PEPPOL BIS 3.0, UBL, and more
- □□ **Decimal Precision**: Arbitrary precision arithmetic for perfect financial calculations
- □□ **Lossless Conversion**: 100% data preservation in round-trip conversions
- □□ **PDF Magic**: Extract and embed XML in PDF/A-3 documents seamlessly
- □□ **TypeScript First**: Fully typed with IntelliSense support throughout

□□ Quick Start

```
# Using pnpm (recommended)
pnpm add @fin.cx/einvoice

# Using npm
npm install @fin.cx/einvoice
```

```
# Using yarn
yarn add @fin.cx/einvoice
```

One-Minute Example

```
import { EInvoice } from '@fin.cx/einvoice';

// Load from any source
const invoice = await EInvoice.fromFile('invoice.xml');      // From file
const invoice2 = await EInvoice.fromXml(xmlString);          // From XML string
const invoice3 = await EInvoice.fromPdf(pdfBuffer);          // From PDF with embedded XML

// Validate with comprehensive EN16931 rules
const validation = await invoice.validate();
console.log(`Valid: ${validation.valid}`);

// Convert between any formats - losslessly!
const xrechnung = await invoice.exportXml('xrechnung');      // For German B2G
const peppol = await invoice.exportXml('ubl');               // For PEPPOL network
const facturx = await invoice.exportXml('facturx');          // For France/Germany
const zugferd = await invoice.exportXml('zugferd');          // For German standard

// Embed into PDF for hybrid invoices
const pdfWithXml = await invoice.exportPdf('facturx');
```

□□ Complete Invoice Creation

```
import { EInvoice } from '@fin.cx/einvoice';

// Create a fully compliant invoice from scratch
const invoice = new EInvoice();

// Essential metadata
invoice.accountingDocId = 'INV-2025-001';
invoice.issueDate = new Date('2025-01-15');
invoice.accountingDocType = 'invoice';
```

```
invoice.currency = 'EUR';
invoice.dueInDays = 30;

// Seller information
invoice.from = {
  type: 'company',
  name: 'Tech Solutions GmbH',
  address: {
    streetName: 'Innovation Street',
    houseNumber: '42',
    city: 'Berlin',
    postalCode: '10115',
    country: 'DE'
  },
  registrationDetails: {
    vatId: 'DE123456789',
    registrationId: 'HRB 123456',
    registrationName: 'Tech Solutions GmbH'
  },
  status: 'active'
};

// Buyer information
invoice.to = {
  type: 'company',
  name: 'Customer Corp SAS',
  address: {
    streetName: 'Rue de la Paix',
    houseNumber: '10',
    city: 'Paris',
    postalCode: '75001',
    country: 'FR'
  },
  registrationDetails: {
    vatId: 'FR987654321',
    registrationId: 'RCS Paris 987654321'
  }
};
```

```
// Payment details - SEPA ready
invoice.paymentAccount = {
  iban: 'DE89370400440532013000',
  bic: 'COBADEFFXXX',
  accountName: 'Tech Solutions GmbH',
  institutionName: 'Commerzbank'
};

// Line items with automatic calculations
invoice.items = [
  {
    position: 1,
    name: 'Cloud Infrastructure Services',
    description: 'Monthly cloud hosting and support',
    articleNumber: 'CLOUD-PRO-001',
    unitQuantity: 1,
    unitNetPrice: 2500.00,
    vatPercentage: 19,
    unitType: 'MON'  // Month
  },
  {
    position: 2,
    name: 'Professional Consulting',
    description: 'Architecture review and optimization',
    articleNumber: 'CONSULT-001',
    unitQuantity: 16,
    unitNetPrice: 150.00,
    vatPercentage: 19,
    unitType: 'HUR'  // Hour
  }
];

// Export to any format you need
const zugferdXml = await invoice.exportXml('zugferd');
const pdfWithXml = await invoice.exportPdf('facturx');
```

□□ Supported Standards & Formats

Standard	Version	Status	Use Case
EN16931	2017	100% Complete	Core European standard
ZUGFeRD	1.0, 2.0, 2.1	Full Support	German B2B/B2C
Factur-X	1.0 (all profiles)	Full Support	France/Germany
XRechnung	2.0, 3.0	Full Support	German public sector
PEPPOL BIS 3.0	3.0	Full Support	Cross-border B2G
UBL	2.1	Full Support	International
CII	D16B	Full Support	Cross Industry

Factur-X Profile Support

```
// Automatic profile detection and validation
const profiles = {
  MINIMUM: 'Essential fields only (BT-1, BT-2, BT-3)',
  BASIC: 'Core invoice with line items',
  BASIC_WL: 'Basic without lines (summary invoices)',
  EN16931: 'Full EN16931 compliance',
  EXTENDED: 'Additional structured data'
};
```

Power Features

Decimal Precision for Financial Accuracy

No more floating-point errors! Built-in arbitrary precision arithmetic:

```
// Perfect financial calculations every time
const calculator = new DecimalCurrencyCalculator('EUR');
const result = calculator.calculateLineNet(
  '3.14159', // Quantity
  '999.99',  // Unit price
  '0'        // Discount (optional)
```

```
);
// Result: 3141.56 (correctly rounded for EUR)
```

□□ Multi-Level Validation

```
// Three-layer validation with detailed diagnostics
const syntaxResult = await invoice.validate(ValidationLevel.SYNTAX);
const semanticResult = await invoice.validate(ValidationLevel.SEMANTIC);
const businessResult = await invoice.validate(ValidationLevel.BUSINESS);

// Get specific rule violations
businessResult.errors.forEach(error => {
  console.log(`Rule ${error.ruleId}: ${error.message}`);
  console.log(`Business Term: ${error.btReference}`);
  console.log(`Field: ${error.field}`);
});
```

□□ Format Detection & Conversion

```
// Automatic format detection
const format = FormatDetector.detectFormat(xmlString);
console.log(`Detected: ${format}`); // 'zugferd', 'facturx', 'xrechnung', etc.

// Intelligent conversion preserves all data
const zugferd = await EInvoice.fromFile('zugferd.xml');
const xrechnung = await zugferd.exportXml('xrechnung');
const backToZugferd = await EInvoice.fromXml(xrechnung);
// All data preserved through round-trip!
```

□□ PDF Operations

```
// Extract XML from PDF invoices
const extractor = new PDFExtractor();
const result = await extractor.extractXml(pdfBuffer);
if (result.success) {
  console.log(`Found ${result.format} invoice`);
```

```
  const invoice = await EInvoice.fromXml(result.xml);
}

// Embed XML into PDF for hybrid invoices
const embedder = new PDFEmbedder();
const pdfWithXml = await embedder.createPdfWithXml(
  existingPdf,
  xmlContent,
  'factur-x.xml',
  'Factur-X Invoice'
);
```

□□ Country-Specific Requirements

□□□ German XRechnung

```
invoice.metadata = {
  customizationId: 'urn:cen.eu:en16931:2017#compliant#urn:xeinkauf.de:kosit:xrechnung_3.0',
  extensions: {
    leitwegId: '991-12345-67',        // Required routing ID
    buyerReference: 'DE-BUYER-REF',    // Mandatory
    sellerContact: {
      name: 'Max Mustermann',
      phone: '+49 30 12345678',
      email: 'invoice@company.de'
    }
  }
};
```

□□□ PEPPOL BIS 3.0

```
invoice.metadata = {
  profileId: 'urn:fdc:peppol.eu:2017:poacc:billing:01:1.0',
  extensions: {
    endpointId: '0088:1234567890128',  // GLN with checksum
```

```
    documentTypeId: 'urn:oasis:names:specification:ubl:schema:xsd:Invoice-
2::Invoice##urn:cen.eu:en16931:2017',
    processId: 'urn:fdc:peppol.eu:2017:poacc:billing:01:1.0'
  }
};
```

□□□ French Chorus Pro

```
invoice.metadata = {
  extensions: {
    siret: '12345678901234',
    serviceCode: 'SERVICE-2025',
    engagementNumber: 'ENG-123456',
    marketReference: 'MARKET-REF-001'
  }
};
```

⚡ Performance Metrics

Lightning-fast operations with minimal memory footprint:

Operation	Speed	Memory
Format Detection	~0.1ms	Minimal
XML Parsing	~0.5ms	~100KB
Full Validation	~2.2ms	~136KB
Format Conversion	~0.6ms	~150KB
PDF Extraction	~5ms	~1MB
PDF Embedding	~10ms	~2MB

□□ Architecture

Plugin-Based Design

```
// Factory pattern for extensibility
DecoderFactory.getDecoder(format) → BaseDecoder
EncoderFactory.getEncoder(format) → BaseEncoder
ValidatorFactory.getValidator(format) → BaseValidator
```

Data Flow

```
Input (XML/PDF) → Format Detection → Decoder → EInvoice Model
                                                    ↓
                                                Validation
                                                    ↓
                                          Encoder → Output (XML/PDF)
```

□□ Security Features

- **XXE Prevention**: External entities disabled by default
- **Resource Limits**: Max 100MB XML, max 100 nesting levels
- **Path Traversal Protection**: Sanitized filenames in PDFs
- **SSRF Mitigation**: Entity blocking in XML processing
- **Input Validation**: Comprehensive input sanitization

□□ Testing

```
# Run all tests
pnpm test

# Run specific test suites
pnpm test test/test.peppol-validator.ts
pnpm test test/test.facturx-validator.ts
pnpm test test/test.semantic-model.ts

# Run with verbose output
pnpm test -- --verbose
```

□□ Advanced Examples

Batch Processing with Concurrency Control

```
import pLimit from 'p-limit';

const limit = pLimit(5); // Max 5 concurrent operations
const files = ['invoice1.xml', 'invoice2.xml', /* ... */];

const results = await Promise.all(
  files.map(file =>
    limit(async () => {
      const invoice = await EInvoice.fromFile(file);
      const validation = await invoice.validate();
      return { file, valid: validation.valid };
    })
  )
);
```

REST API Integration

```
app.post('/api/invoice/convert', async (req, res) => {
  try {
    const { xml, targetFormat } = req.body;
    const invoice = await EInvoice.fromXml(xml);

    // Validate before conversion
    const validation = await invoice.validate();
    if (!validation.valid) {
      return res.status(400).json({
        error: 'Invalid invoice',
        violations: validation.errors
      });
    }
```

```
    const converted = await invoice.exportXml(targetFormat);
    res.json({ success: true, xml: converted });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

□□ What Makes Us Different

□□ 100% EN16931 Compliance

- All 162 Business Terms implemented
- All 32 Business Groups structured
- Complete semantic model with BT/BG validation
- Official Schematron rules integrated

□□ Production Excellence

- **500+ test cases** ensuring reliability
- **Battle-tested** with real-world invoice corpus
- **Memory efficient** - handles 1000+ line items
- **Thread-safe** for concurrent processing

□□ Developer Experience

- **IntelliSense everywhere** - fully typed API
- **Detailed error messages** with recovery hints
- **Static factory methods** for intuitive usage
- **Comprehensive documentation** with real examples

□□ Installation Requirements

- Node.js 18+ or modern browser
- TypeScript 5.0+ (for TypeScript projects)
- ~15MB installed size

- Zero native dependencies

□□ Standards Compliance

This library implements:

- **EN 16931-1:2017** - Core invoice model
- **CEN/TS 16931-3** - Syntax bindings
- **ISO 4217** - Currency codes
- **ISO 3166** - Country codes
- **UN/ECE Rec 20** - Units of measure
- **ISO 6523** - Organization identifiers

License and Legal Information

This repository contains open-source code that is licensed under the MIT License. A copy of the MIT License can be found in the license file within this repository.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH and are not included within the scope of the MIT license granted herein. Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines, and any usage must be approved in writing by Task Venture Capital GmbH.

Company Information

Task Venture Capital GmbH
Registered at District court Bremen HRB 35230 HB, Germany

For any legal inquiries or if you require further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture

Capital GmbH of any derivative works.

Revision #5
Created 2026-03-28 10:49:18 UTC by foss.global Team
Updated 2026-03-28 12:14:43 UTC by foss.global Team