

# INTEGRĀCIJAS INSTRUKCIJA E-RĒŅINU API BIBLIOTĒKAI

*TILDE SIA*  
*VERSIJA 1.0*

*RĪGA*

Šī dokumenta autoru personiskās tiesības pieder tā izstrādātājiem. Pieļaujama dokumentā iekļautās informācijas citēšana un izmantošana atvasinātu darbu veidošanai, iekļaujot atsauci uz šo dokumentu.

Visas tekstā izmantotās tirdzniecības zīmes pieder to īpašniekiem un ir izmantotas tikai kā atsauces.

Darbi tiek līdzfinansēti no Eiropas infrastruktūras savienošanas instrumenta (Connecting Europe Facility) (tupmāk – CEF) projektā “Eiropas elektroniskais rēķins” (e-Invoices CEF Project) īstenošanu līdzekļiem.

### ***Dokumenta autori***

Viesturs Slaidiņš

Zigmunds Beļskis

### ***Kontaktpersona***

Viesturs Slaidiņš

SIA Tilde

Vienības gatve 75A

Rīga, LV-1004

E-pasts: viesturs.slaidins@tilde.lv

### ***Izmaiņu lapa***

Versija	Datums	Apraksts	Autors
1.0	20.12.2019.	Izveidota dokumenta sākotnējā versija.	Dokumenta autori

---

# Satura rādītājs

<b>Satura rādītājs .....</b>	<b>3</b>
<b>1. Ievads .....</b>	<b>4</b>
1.1. Nolūks .....	4
<b>2. Integrācijas instrukcija .....</b>	<b>4</b>
2.1. Izveidot e-rēķinu .....	4
2.2. Pārbaudīt savienojumu ar E-adrese .....	8
2.3. E-rēķina sūtīšana caur E-adrese .....	8
2.1. E-rēķinu saņemšana caur e-adreses kanālu .....	8
2.2. E-rēķinu saņemšanas apstiprinājums .....	8
2.3. E-adreses pārbaude, izmantojot reģistrācijas numuru vai personas kodu .....	8
2.4. E-adrešu saraksta izguve .....	9
2.5. E-rēķina izgūšana HTML formatējumā .....	9
2.6. E-rēķina izgūšana PDF formatējumā .....	9
2.7. E-rēķina saglabāšana XML formātā .....	9
2.8. E-rēķina izveide no XML formāta .....	9
2.9. E-rēķina validācija .....	10
2.9.1. XML .....	10
2.9.2. Invoice .....	10

# 1. Ievads

## 1.1. Nolūks

Šis dokuments ir veidots, lai apkopotu integrācijas piemērus publiskai e-rēķinu API bibliotēkai.

# 2. Integrācijas instrukcija

Šajā nodaļā ir apkopoti integrācijas piemēri e-rēķinu publiskai API programmatūras bibliotēkai.

## 2.1. Izveidot e-rēķinu

```
var invoice = new EInvoice.EInvoice(new InvoiceType()
{
    ID = new IdentifierType() { Value = "TEST-" + Guid.NewGuid().ToString().ToUpper() },
    CustomizationID = "urn:cen.eu:en16931:2017#compliant#urn:fdc:peppol.eu:2017:poacc:billing:3.0",
    ProfileID = "urn:fdc:peppol.eu:2017:poacc:billing:01:1.0",
    IssueDate = new DateType() { Value = DateTime.Now },
    DueDate = new DateType() { Value = DateTime.Now },
    InvoiceTypeCode = new CodeType()
    {
        Value = "380"
    },
    BuyerReference = "43203002696",
    DocumentCurrencyCode = "EUR",
    Note = new List<TextType> { new TextType() { Value = "Šis ir tikai tests. Maksāt nevajag!" } },
    AccountingSupplierParty = new SupplierPartyType()
    {
        Party = new PartyType()
        {
            EndpointID = new IdentifierType()
            {
                schemeID = "9939",
                Value = "LV43203002696"
            },
            PostalAddress = new AddressType()
            {
                Country = new CountryType()
                {
                    IdentificationCode = "LV"
                },
                AddressLine = new List<AddressLineType>() {
                    new AddressLineType() {
                        Line = "Vienības gatve 75A, LV-1004, Rīga"
                    }
                }
            },
            PartyName = new List<PartyNameType>() { new PartyNameType() { Name = "SIA Tilde" } },
            PartyLegalEntity = new List<PartyLegalEntityType>() {
                new PartyLegalEntityType() {
                    CompanyID = "43203002696",
                    RegistrationName = "SIA Tilde",
                }
            },
            PartyTaxScheme = new List<PartyTaxSchemeType>() {
                new PartyTaxSchemeType() {
                    CompanyID = "LV43203002696",
```

```

        TaxScheme = new TaxSchemeType(){
            ID = "VAT"
        }
    },
}
},
AccountingCustomerParty = new CustomerPartyType()
{
    Party = new PartyType()
    {
        EndpointID = new IdentifierType()
        {
            schemeID = "9939",
            Value = "LV43203002696"
        },
        PostalAddress = new AddressType()
        {
            Country = new CountryType()
            {
                IdentificationCode = "LV"
            },
            AddressLine = new List<AddressLineType>() {
                new AddressLineType(){
                    Line = "Vecstāmeriena, \"Lāčplēši\", Stāmerienas pag., Gulbenes nov., Latvija,
LV-4406"
                }
            },
            PartyName = new List<PartyNameType>() { new PartyNameType() { Name = "SIA DEMO-LITION" } },
            PartyLegalEntity = new List<PartyLegalEntityType>() {
                new PartyLegalEntityType() {
                    CompanyID = "40003240524",
                    RegistrationName = "SIA DEMO-LITION",
                }
            },
            PartyTaxScheme = new List<PartyTaxSchemeType>() {
                new PartyTaxSchemeType() {
                    CompanyID = "LV40003240524",
                    TaxScheme = new TaxSchemeType(){
                        ID = "VAT"
                    }
                }
            },
        },
    },
},
InvoiceLine = new List<InvoiceLineType>() {
    new InvoiceLineType() {
        ID = "1",
        Item = new ItemType() {
            Name = "Pakalpojumu Cloud",
            ClassifiedTaxCategory = new List<TaxCategoryType>() {
                new TaxCategoryType() {
                    ID = "S",
                    Percent = new PercentType(){ Value = 21m },
                    TaxScheme = new TaxSchemeType(){
                        ID = "VAT"
                    }
                }
            },
        },
    },
}

```

```

    }
},
InvoicedQuantity = new QuantityType () { unitCode = "HUR", Value = 5m },
Price = new PriceType(){ PriceAmount = new AmountType() { currencyID = "EUR", Value = 100m }
},

AllowanceCharge = new List<AllowanceChargeType>() {
    new AllowanceChargeType() {
        ChargeIndicator = true,
        AllowanceChargeReasonCode = new CodeType(){
            Value = "ABK" /*Miscellaneous*/
        },
        BaseAmount = new AmountType() { currencyID = "EUR", Value = 500m },
        MultiplierFactorNumeric = 100,
        Amount = new AmountType() { currencyID = "EUR", Value = 500m },
    }
},
LineExtensionAmount = new AmountType() { currencyID = "EUR", Value = 1000m },
},
new InvoiceLineType() {
    ID = "2",
    Item = new ItemType() {
        Name = "Tildes Jumis Pro",
        ClassifiedTaxCategory = new List<TaxCategoryType>() {
            new TaxCategoryType() {
                ID = "E",
                Percent = new PercentType(){ Value = 0m },
                TaxScheme = new TaxSchemeType(){
                    ID = "VAT"
                }
            }
        }
    }
},
InvoicedQuantity = new QuantityType () { unitCode = "H87", Value = 2m },
Price = new PriceType(){ PriceAmount = new AmountType() { currencyID = "EUR", Value = 10m }
},

AllowanceCharge = new List<AllowanceChargeType>() {
    new AllowanceChargeType() {
        AllowanceChargeReasonCode = new CodeType(){
            Value = "95" /*Discount*/
        },
        ChargeIndicator = false,
        BaseAmount = new AmountType() { currencyID = "EUR", Value = 20m },
        MultiplierFactorNumeric = 20m,
        Amount = new AmountType() { currencyID = "EUR", Value = 4m }
    }
},
LineExtensionAmount = new AmountType() { currencyID = "EUR", Value = 16m },
},
new InvoiceLineType() {
    ID = "3",
    Item = new ItemType() {
        Name = "Tildes Jumis Standarta",
        ClassifiedTaxCategory = new List<TaxCategoryType>() {
            new TaxCategoryType() {
                ID = "E",
                Percent = new PercentType(){ Value = 0m },
                TaxScheme = new TaxSchemeType(){
                    ID = "VAT"
                }
            }
        }
    }
}
}

```

```

    },
    InvoicedQuantity = new QuantityType () { unitCode = "H87", Value = 1m },
    Price = new PriceType(){ PriceAmount = new AmountType() { currencyID = "EUR", Value = 1m }
},
    LineExtensionAmount = new AmountType() { currencyID = "EUR", Value = 1m }
},
},
LegalMonetaryTotal = new MonetaryTotalType()
{
    LineExtensionAmount = new AmountType() { currencyID = "EUR", Value = 1017.00m },
    TaxExclusiveAmount = new AmountType() { currencyID = "EUR", Value = 1017.00m },
    TaxInclusiveAmount = new AmountType() { currencyID = "EUR", Value = 1227.00m },
    PayableAmount = new AmountType() { currencyID = "EUR", Value = 1227.00m }
},
TaxTotal = new List<TaxTotalType>() {
    new TaxTotalType(){
        TaxAmount = new AmountType() { currencyID = "EUR", Value = 210m },
        TaxSubtotal = new List<TaxSubtotalType>() {
            new TaxSubtotalType(){
                TaxableAmount = new AmountType() { currencyID = "EUR", Value = 1000m },
                TaxAmount = new AmountType() { currencyID = "EUR", Value = 210m },
                TaxCategory = new TaxCategoryType (){
                    ID = "S",
                    Percent = new PercentType(){ Value = 21m },
                    TaxScheme = new TaxSchemeType(){
                        ID = "VAT"
                    }
                }
            },
            new TaxSubtotalType(){
                TaxableAmount = new AmountType() { currencyID = "EUR", Value = 17m },
                TaxAmount = new AmountType() { currencyID = "EUR", Value = 0m },
                TaxCategory = new TaxCategoryType (){
                    TaxExemptionReason = new List<TextType>(){
                        new TextType(){
                            Value = "Nav apliekams"
                        }
                    },
                    ID = "E",
                    Percent = new PercentType(){ Value = 0m },
                    TaxScheme = new TaxSchemeType(){
                        ID = "VAT"
                    }
                }
            }
        },
    }
},
};

invoice.Document.Xmlns = new System.Xml.Serialization.XmlSerializerNamespaces(new[]{
    new XmlQualifiedName("cbc","urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2"),
    new XmlQualifiedName("cac","urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2"),
});

```

## 2.2. Pārbaudīt savienojumu ar E-adrese

```
var proxy = EInvoiceService.ConfigureEAddressConnection(new ConfigurationOptions(  
    thumbprint: "5101A94FB35A271468A92DF4F07B9F6D1F8331D9",  
    serviceAddress: test.vraa.gov.lv/Vraa.Div.WebService.UnifiedInterface/UnifiedService.svc"  
));  
var success = proxy.TestEAddressConnection();
```

## 2.3. E-rēķina sūtīšana caur E-adrese

```
var proxy = EInvoiceService.ConfigureEAddressConnection(new ConfigurationOptions(  
    thumbprint: "5101A94FB35A271468A92DF4F07B9F6D1F8331D9",  
    serviceAddress: test.vraa.gov.lv/Vraa.Div.WebService.UnifiedInterface/UnifiedService.svc"  
));  
var messageId = proxy.SendEInvoice(new EInvoiceMessage()  
{  
    MessageTitle = "Testa ziņojums Nr.1",  
    Description = "apraksts",  
    FromTitle = "TILDE Testētājs",  
    MessageFrom = "_DEFAULT@40003027238",  
    MessageTo = "_DEFAULT@40003027238",  
    EInvoice = new EInvoice.EInvoice(/*Saturš no "Izveidot e-rēķinu"*/)  
});
```

## 2.1. E-rēķinu saņemšana caur e-adreses kanālu

```
var proxy = EInvoiceService.ConfigureEAddressConnection(new ConfigurationOptions(  
    thumbprint: "5101A94FB35A271468A92DF4F07B9F6D1F8331D9",  
    serviceAddress: test.vraa.gov.lv/Vraa.Div.WebService.UnifiedInterface/UnifiedService.svc"  
));  
var messages = proxy.ReceiveEInvoice();
```

## 2.2. E-rēķinu saņemšanas apstiprinājums

```
var proxy = EInvoiceService.ConfigureEAddressConnection(new ConfigurationOptions(  
    thumbprint: "5101A94FB35A271468A92DF4F07B9F6D1F8331D9",  
    serviceAddress: test.vraa.gov.lv/Vraa.Div.WebService.UnifiedInterface/UnifiedService.svc"  
));  
//Saņemšana  
var messages = proxy.ReceiveEInvoice();  
//Apstiprināšana  
proxy.ConfirmEInvoice(messages);
```

## 2.3. E-adreses pārbaude, izmantojot reģistrācijas numuru vai personas kodu

```
var proxy = EInvoiceService.ConfigureEAddressConnection(new ConfigurationOptions(  
    thumbprint: "5101A94FB35A271468A92DF4F07B9F6D1F8331D9",  
    serviceAddress: test.vraa.gov.lv/Vraa.Div.WebService.UnifiedInterface/UnifiedService.svc"  
));  
var validationResult = proxy.ValidateEAddress(new List<string>() { "40003027238", "12109311111" });
```



## 2.4. E-adrešu saraksta izguve

- Pilnu sarakstu ir iespējams izgūt ne biežāk kā reizi 12 stundās.
  - 12 stundas ir kopīgas uz visām ierīcēm viena e-adreses konta ietvaros
  - To ir nepieciešams veikt ja:
    - Ļoti sen nav pieprasīti dati.
    - Ja šī ir pirmā reize, kad dati tiek pieprasīti uz dotā datora.
- E-adrešu saraksts tiek automātiski saglabāts uz diska, lai būtu iespēja veikt inkrementālu pieprasījumu.
  - %AppData%\Tilde\EInvoice\AddressCache.json

```
var proxy = EInvoiceService.ConfigureEAddressConnection(new ConfigurationOptions(  
    thumbprint: "5101A94FB35A271468A92DF4F07B9F6D1F8331D9",  
    serviceAddress: test.vraa.gov.lv/Vraa.Div.WebService.UnifiedInterface/UnifiedService.svc"  
));  
var addresses = proxy.GetEAddressList();
```

## 2.5. E-rēķina izgūšana HTML formatējumā

- Veidojot HTML var:
  - Lokalizēt/mainīt rēķina attēlojumu, padodot savu HTML sagatavi
  - Lokalizēt/mainīt mērvienību tekstu, padodot vārdnīcu

```
var eInvoice = new EInvoice.EInvoice(/*Saturis no "Izveidot e-rēķinu"*/)  
var htmlText = eInvoice.GetHTML();
```

## 2.6. E-rēķina izgūšana PDF formatējumā

- Veidojot PDF var:
  - Lokalizēt/mainīt rēķina attēlojumu, padodot savu HTML sagatavi
  - Lokalizēt/mainīt mērvienību tekstu, padodot vārdnīcu
  - Padot DinkToPdf iestatījumus, PDF izveidošanas korekcijai

```
var eInvoice = new EInvoice.EInvoice(/*Saturis no "Izveidot e-rēķinu"*/)  
var pdfBytes = eInvoice.GetPDF();
```

## 2.7. E-rēķina saglabāšana XML formātā

```
var eInvoice = new EInvoice.EInvoice(/*Saturis no "Izveidot e-rēķinu"*/)  
var xmlText = eInvoice.GetXML();
```

## 2.8. E-rēķina izveide no XML formāta

```
var eInvoice = new EInvoice.EInvoice()  
eInvoice.LoadXML("UBL XML teksts");
```

## 2.9. E-rēķina validācija

### 2.9.1. XML

```
var errorList = EInvoice.EInvoice.Validate("UBL XML teksts")
```

### 2.9.2. EInvoice

```
var eInvoice = new EInvoice.EInvoice(/*Saturis no "Izveidot e-rēķinu"*/)
var valid = eInvoice.Validate();
var errorList = eInvoice.ValidationError;
```