Peppol Document Format(s)

2025-02-20

Jelte Jansen, Ionite B.V.

1 Introduction

This document aims to provide an introduction to the Peppol BIS Invoice document specification, by discussing a brief overview of the history of business document standards as it pertains to Peppol BIS, a reading guide for the specification documents, and some suggestions on how to get started when adding Peppol BIS support to an existing (invoice) software system.

While the focus is on Peppol BIS (UBL) Invoices, a lot of the underlying principles also go for other Peppol BIS documents, and potentially non-peppol business documents as well. We will occasionally mention some other document types, but the main goal is providing information about Peppol BIS UBL invoices, as these are the main documents used on the Peppol network.

This document will *not* discuss other technical topics regarding the peppol network, such as transport protocols and business level processes.

2 A (brief) history of structured business document types

When data is exchanged between two parties, they need to agree upon the format and the meaning of that data. For a bilateral exchange, this agreement could be made in an ad-hoc manner; the parties have a few meetings, agree on some data structure, and both implement that data structure into their systems.

In a bilateral exchange between two parties, this could be done in an ad-hoc manner: the parties have a few meetings, decide on a data format, and implement it in their respective systems. This process, however, does not scale: if there are multiple parties communicating with each other, each with their own formats, the number of formats in use rises exponentially, and in many cases, which increases the amount of work that needs to be done, as well as the number of mistakes that are made in implementation and operation.

This is where standards are useful: in a bilateral exchange agreement, two parties can decide on which standards to use, thereby reducing the amount of work they need to do to implement the exchange. A community of parties can also agree to use the same standards, which would reduce that amount of work for everyone involved, including future community members.

For instance, they can decide that all data exchange is done using XML, so that they can use standard software for building and parsing the data structure. Taking this one step further, they could also agree that the XML should have a fixed structure, with restrictions on the elements that may occur and what the values of those elements may be. If that is standardized, the parties know exactly which features they need to implement in order to exchange data with other parties that are using the same standard.

In the world of business document exchange, there have always been a lot of bilateral agreements, and small communities creating unique data formats for their data exchange needs. But often, the same *type* of information is exchanged; think of catalogues, orders, and invoices, to name a few. The realization that this led to a lot of duplicate effort gave rise to the creation of several efforts to define a standard format for business documents. Two of the more widely known are:

- Universal Business Language (UBL) from Oasis [UBL]

- The Cross Industry formats from UN/CEFACT, such as the Cross Industy Invoice (CII) [CII]

We'll focus on UBL, but will occasionally mention UN/CEFACT and other standards.

2.1 UBL

The Universal Business Language [UBL] defines a number of XML schemas for business documents. In the first version of UBL, these were Order, Billing, and Filfulment documents only, but many more schemas were added in later versions.

XML Schemas

An XML Schema is a structural specification for XML documents: it defines which XML tags may occur, in which order, their cardinality (e.g. how often they must or may appear), which XML attributes the tags may have, and basic type information on their values (numbers, text, date, time, etc.).

Many XML processing libraries and tools provide functionality to validate a given XML document against a specific schema. Section 5 contains more information on document validation.

The UBL specification provides a number of basic XML types ('common basic components'), aggregates those into complex types ('common aggregated components'), and finally defines business documents that contain a number of these basic and aggregated components.

You'll recognize this structure when you see the XML of UBL documents, as it is a convention (though not a rule!) to use XML namespace prefixes `cbc:` for basic components, and `cac:` for aggregated components.

The latest version of UBL is 2.4. However, we shall refer to UBL 2.1 [UBL2.1], as that is the version that most Peppol BIS documents are currently based on. For us, for now, the most important one is the UBL Invoice [UBL-2.1-INVOICE]

If you want to see a full overview of all elements of the UBL 2.1 invoice, there is a helpful online view at [DATYPIC-UBL2.1-INVOICE].

The UBL invoice should contain just about everything you might need to commonly exchange invoice information. And even in the case it's not, it has a field 'UBLExtensions', where you can add your own additional data structure.

Sample of a very basic UBL invoice:

```xml
    <Invoice>
       <cbc:ID>123</cbc:ID>
       <cbc:IssueDate>2011-09-22</cbc:IssueDate>
       <cac:InvoicePeriod>
          <cbc:StartDate>2011-08-01</cbc:StartDate>
          <cbc:EndDate>2011-08-31</cbc:EndDate>
       </cac:InvoicePeriod>
       <cac:AccountingSupplierParty>
          <cac:Party>
             <cac:PartyName>
                <cbc:Name>Custom Cotter Pins</cbc:Name>
             </cac:PartyName>
          </cac:Party>
       </cac:AccountingSupplierParty>
       <cac:AccountingCustomerParty>
          <cac:Party>
             <cac:PartyName>
                <cbc:Name>North American Veeblefetzer</cbc:Name>
             </cac:PartyName>
          </cac:Party>
       </cac:AccountingCustomerParty>
       <cac:LegalMonetaryTotal>
          <cbc:PayableAmount currencyID="CAD">100.00</cbc:PayableAmount>
       </cac:LegalMonetaryTotal>
       <cac:InvoiceLine>
          <cbc:ID>1</cbc:ID>
          <cbc:LineExtensionAmount
currencyID="CAD">100.00</cbc:LineExtensionAmount>
          <cac:Item>
             <cbc:Description>Cotter pin, MIL-SPEC</cbc:Description>
          </cac:Item>
       </cac:InvoiceLine>
    </Invoice>
```

2.2 The problem with UBL

The problem with UBL (and CII) is that it is *too* broad; since they aim to cover most, if not all, business use-cases, they allow for many optional elements and data constructs that *most* businesses will never need.

That makes it easy to put whatever data you have into a valid UBL document, since you can choose the xml elements you want to use for that data. There is no need to use any of the other fields, so your implementation can be limited to those fields you have chosen.

However, if you're the one receiving arbitrary UBL documents, you should be able to process all possible elements, as you don't know which fields the sender uses, and what their values or constraints may be.

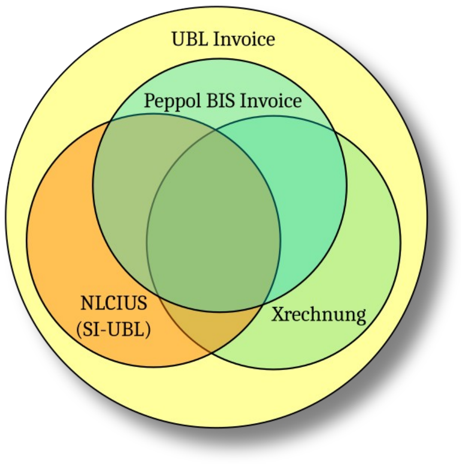
What happened next was that users of UBL would realize they really only used a small subset of UBL, say 100 to 200 possible elements. They'd agree amongst each other that certain fields were not to be used, others would be mandatory, and for some the values would be restricted to a limited set of options.

UBL Customizations

UBL provides support for defining subsets of its elements, or other restrictions to valid documents. This is called a *customization*, and indicated through the UBL Element `cbc:CustomizationID`, which contains an identifier indication which customization is in use.

Essentially, this is the element that shows which rules and restrictions the document is supposed to be compliant to, and each document type has a unique CustomizationID. A document can only have one CustomizationID.

This happened in many places, some because the creators were not aware of others, others because the creators did not agree on the choices of other dialects, and yet more simply because of the Not-Invented-Here effect. Just to name a few, there were UBL-OHNL, Peppol BIS (1/2), SI-UBL, XRechnung, OIOUBL, and many, many more.



Venn Diagram of some of the customizations

While the ‘general’ structure of these formats was the same, there were many, many different customizations of these in use (and there still are!), and software that supports one is not guaranteed to work with the others; they may use elements that another format has excluded, or leave out elements that another format has made mandatory.

2.3 European Norm EN-16931

At some point, there was a directed effort to improve this, at least for invoices at EU government organizations. This effort covered not just UBL-based invoice formats, but other formats as well: European Norm EN-16931 [EN-16931].

EN-16931 describes a semantic data model for “Core Invoices”; e.g. it defines a set of business terms for an invoice, and a set of business rules that compliant documents must adhere to. It also defines mappings for that semantic model to UBL and CII (and EDIFACT, but EDIFACT is very different from the others).

EN-16931 came with EU Directive 2014/55, which dictates that member states adopt it so that it will become mandatory for all public contracting authorities and contracting entities to receive and process eInvoices complying with EN-16931.

EN-16931 comprises several documents:

Part	Identifier	Name
1	EN 16931-1:2017	Semantic data model of the core elements of an electronic invoice
2	CEN/TS 16931-2:2017	List of syntaxes that comply with EN 16931
3-1	CEN/TS 16931-3-1:2017	Methodology for syntax bindings of the core elements of an electronic invoice
3-2	CEN/TS 16931-3-2:2017	Syntax binding for ISO/IEC 19845 (UBL 2.1) invoice and credit note
3-3	CEN/TS 16931-3-3:2017	Syntax binding for UN/CEFACT XML Industry Invoice D16B
3-4	CEN/TS 16931-3-4:2017	Syntax binding for UN/EDIFACT Invoice D16B

2.4 Semantic models and syntax mappings

Many business document specifications make a distinction between the semantic model (what data there is) and the syntax (how that data is written down), and specify these in separate documents. This choice is usually made because the model can be mapped to multiple syntaxes. For instance, EN-16931 and Peppol BIS both provide mappings to UBL as well as UN/CEFACT CII, as do a number of other document types, such as XRechnung. Note that while Peppol BIS does provide a mapping to UN/CEFACT CII, the UBL syntax is the default and mandatory one. EN-16931 does not choose any particular syntax over the other.

Keep in mind that this is not always the case; some specifications have only a single syntax, and the semantic model and syntax mapping are a single specification.

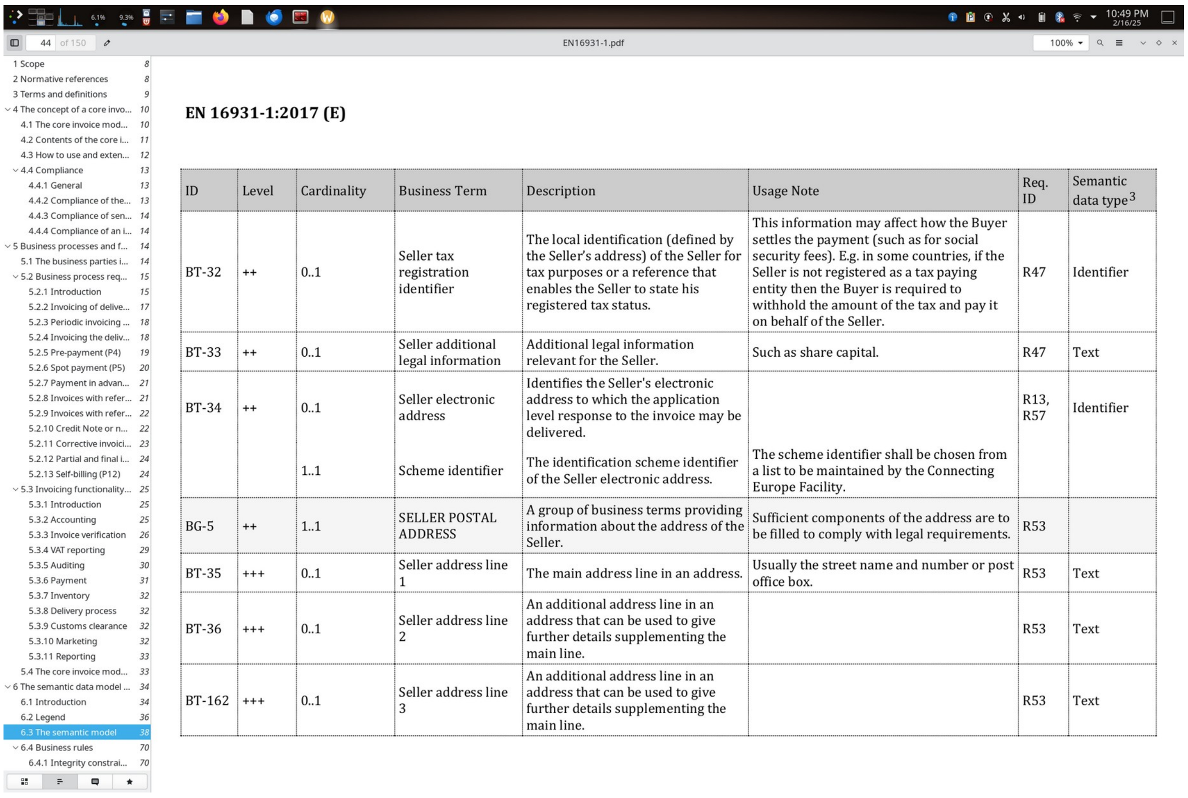
Usually, when speaking about a specific document type or format, we mean both the semantic model and the mapping together.

2.4.1 Semantic models

A semantic model describes what data can be modeled, what that data represents within its context, what the limitations on the data sets are, and up to a point, how the data is to be treated or processed.

A Semantic model contains The following:

1. A list of data elements (“Business Terms”) and data groups (“Business Groups”)
2. What those data elements and groups mean (hence *semantic* model)
3. How often they must/may occur (their cardinality)
4. Additional rules and restrictions (e.g. the total of an invoice must equals the sum of the invoice lines) (“Business Rules”)
5. How the data should, on a high level, be processed, and how it may relate to business processes


44 of 150

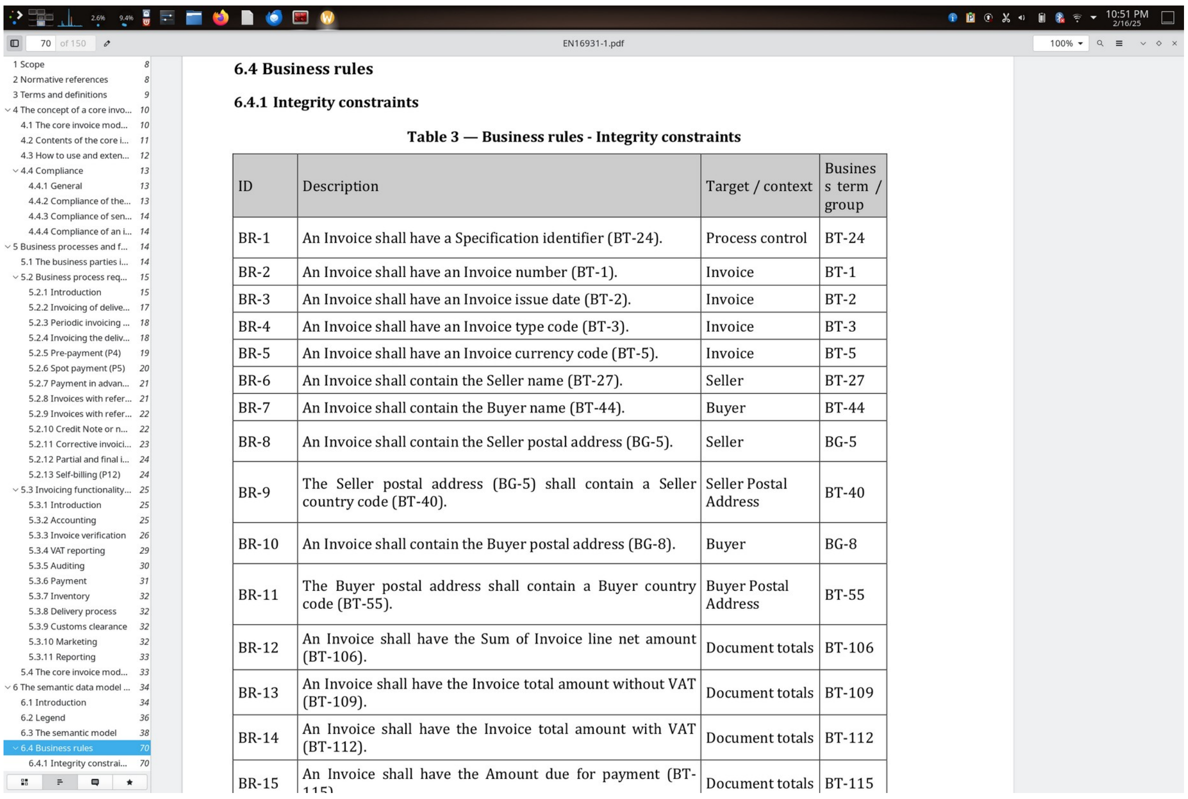
EN16931-1.pdf

100%

EN 16931-1:2017 (E)

ID	Level	Cardinality	Business Term	Description	Usage Note	Req. ID	Semantic data type[3]
BT-32	++	0..1	Seller tax registration identifier	The local identification (defined by the Seller's address) of the Seller for tax purposes or a reference that enables the Seller to state his registered tax status.	This information may affect how the Buyer settles the payment (such as for social security fees). E.g. in some countries, if the Seller is not registered as a tax paying entity then the Buyer is required to withhold the amount of the tax and pay it on behalf of the Seller.	R47	Identifier
BT-33	++	0..1	Seller additional legal information	Additional legal information relevant for the Seller.	Such as share capital.	R47	Text
BT-34	++	0..1	Seller electronic address	Identifies the Seller's electronic address to which the application level response to the invoice may be delivered.		R13, R57	Identifier
		1..1	Scheme identifier	The identification scheme identifier of the Seller electronic address.	The scheme identifier shall be chosen from a list to be maintained by the Connecting Europe Facility.		
BG-5	++	1..1	SELLER POSTAL ADDRESS	A group of business terms providing information about the address of the Seller.	Sufficient components of the address are to be filled to comply with legal requirements.	R53	
BT-35	+++	0..1	Seller address line 1	The main address line in an address.	Usually the street name and number or post office box.	R53	Text
BT-36	+++	0..1	Seller address line 2	An additional address line in an address that can be used to give further details supplementing the main line.		R53	Text
BT-162	+++	0..1	Seller address line 3	An additional address line in an address that can be used to give further details supplementing the main line.		R53	Text

Sample of EN-16931 Business Terms


70 of 150

EN16931-1.pdf

100%

1 Scope 8
2 Normative references 8
3 Terms and definitions 9
4 The concept of a core invo... 10
4.1 The core invoice mod... 10
4.2 Contents of the core i... 11
4.3 How to use and exten... 12
4.4 Compliance 13
4.4.1 General 13
4.4.2 Compliance of the... 13
4.4.3 Compliance of sen... 14
4.4.4 Compliance of an i... 14
5 Business processes and f... 14
5.1 The business parties i... 14
5.2 Business process req... 15
5.2.1 Introduction 15
5.2.2 Invoicing of delive... 17
5.2.3 Periodic invoicing ... 18
5.2.4 Invoicing the deliv... 18
5.2.5 Pre-payment (P4) 19
5.2.6 Spot payment (P5) 20
5.2.7 Payment in advan... 21
5.2.8 Invoices with refer... 21
5.2.9 Invoices with refer... 22
5.2.10 Credit Note or n... 22
5.2.11 Corrective invoici... 23
5.2.12 Partial and final i... 24
5.2.13 Self-billing (P12) 24
5.3 Invoicing functionality... 25
5.3.1 Introduction 25
5.3.2 Accounting 25
5.3.3 Invoice verification 26
5.3.4 VAT reporting 29
5.3.5 Auditing 30
5.3.6 Payment 31
5.3.7 Inventory 32
5.3.8 Delivery process 32
5.3.9 Customs clearance 32
5.3.10 Marketing 32
5.3.11 Reporting 33
5.4 The core invoice mod... 33
6 The semantic data model ... 34
6.1 Introduction 34
6.2 Legend 36
6.3 The semantic model 38
6.4 Business rules 70
6.4.1 Integrity constrai... 70

6.4 Business rules

6.4.1 Integrity constraints

Table 3 — Business rules - Integrity constraints

ID	Description	Target / context	Business term / group
BR-1	An Invoice shall have a Specification identifier (BT-24).	Process control	BT-24
BR-2	An Invoice shall have an Invoice number (BT-1).	Invoice	BT-1
BR-3	An Invoice shall have an Invoice issue date (BT-2).	Invoice	BT-2
BR-4	An Invoice shall have an Invoice type code (BT-3).	Invoice	BT-3
BR-5	An Invoice shall have an Invoice currency code (BT-5).	Invoice	BT-5
BR-6	An Invoice shall contain the Seller name (BT-27).	Seller	BT-27
BR-7	An Invoice shall contain the Buyer name (BT-44).	Buyer	BT-44
BR-8	An Invoice shall contain the Seller postal address (BG-5).	Seller	BG-5
BR-9	The Seller postal address (BG-5) shall contain a Seller country code (BT-40).	Seller Postal Address	BT-40
BR-10	An Invoice shall contain the Buyer postal address (BG-8).	Buyer	BG-8
BR-11	The Buyer postal address shall contain a Buyer country code (BT-55).	Buyer Postal Address	BT-55
BR-12	An Invoice shall have the Sum of Invoice line net amount (BT-106).	Document totals	BT-106
BR-13	An Invoice shall have the Invoice total amount without VAT (BT-109).	Document totals	BT-109
BR-14	An Invoice shall have the Invoice total amount with VAT (BT-112).	Document totals	BT-112
BR-15	An Invoice shall have the Amount due for payment (BT-115)	Document totals	BT-115

Sample of EN-16931 Business Rules

2.4.2 Cardinality

Both semantic models and syntax mappings define a cardinality for each of the elements. The cardinality defines how many times any given data element may occur. It is often expressed using the following notation:

```
<Minimum number>..<Maximum number, or n if there is no limit>
```

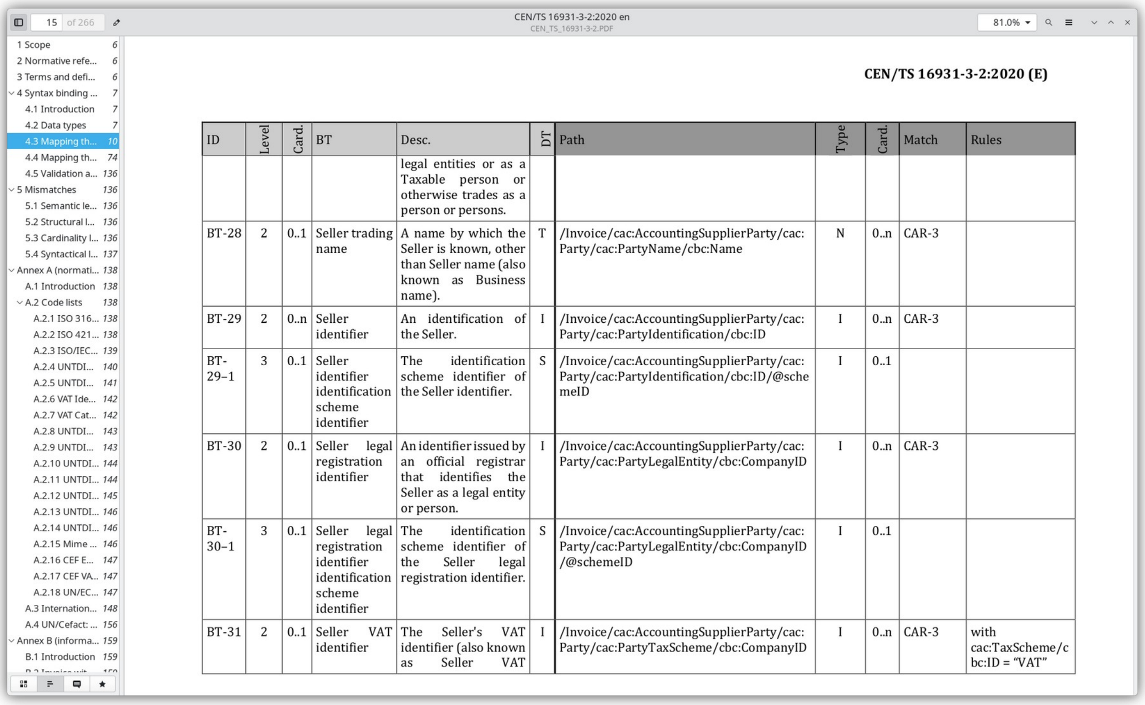
For example:

- **0..1**: Optional element, may occur only once
- **1..1**: Mandatory element, may occur only once
- **0..n**: Optional element, may occur multiple times
- **1..n**: Mandatory element, may occur multiple times
- **1..2**: Mandatory element, may occur once or twice
- etc.

Note that the cardinality specifies how many times that element may occur *within its parent element*. For instance, the cardinality of a the invoice line amount is 1..1, meaning that there must be exactly one line amount per invoice line, not that there must be exactly one line amount in the entire document.

2.4.3 Syntax mappings

A Syntax Mapping (often just called mapping or syntax) maps the business terms and groups to a specific format, e.g. XML, or more specifically, UBL, CII, or EDIFACT. This usually adds a fixed order of the elements as well.



15 of 266

CEN/TS 16931-3-2:2020 en
CEN_TS_16931-3-2.PDF

81.0%

CEN/TS 16931-3-2:2020 (E)

ID	Level	Card.	BT	Desc.	DT	Path	Type	Card.	Match	Rules
				legal entities or as a Taxable person or otherwise trades as a person or persons.						
BT-28	2	0..1	Seller trading name	A name by which the Seller is known, other than Seller name (also known as Business name).	T	/Invoice/cac:AccountingSupplierParty/cac:Party/cac:PartyName/cbc:Name	N	0..n	CAR-3	
BT-29	2	0..n	Seller identifier	An identification of the Seller.	I	/Invoice/cac:AccountingSupplierParty/cac:Party/cac:PartyIdentification/cbc:ID	I	0..n	CAR-3	
BT-29–1	3	0..1	Seller identifier identification scheme identifier	The identification scheme identifier of the Seller identifier.	S	/Invoice/cac:AccountingSupplierParty/cac:Party/cac:PartyIdentification/cbc:ID/@schemeID	I	0..1		
BT-30	2	0..1	Seller legal registration identifier	An identifier issued by an official registrar that identifies the Seller as a legal entity or person.	I	/Invoice/cac:AccountingSupplierParty/cac:Party/cac:PartyLegalEntity/cbc:CompanyID	I	0..n	CAR-3	
BT-30–1	3	0..1	Seller legal registration identifier identification scheme identifier	The identification scheme identifier of the Seller legal registration identifier.	S	/Invoice/cac:AccountingSupplierParty/cac:Party/cac:PartyLegalEntity/cbc:CompanyID/@schemeID	I	0..1		
BT-31	2	0..1	Seller VAT identifier	The Seller's VAT identifier (also known as Seller VAT	I	/Invoice/cac:AccountingSupplierParty/cac:Party/cac:PartyTaxScheme/cbc:CompanyID	I	0..n	CAR-3	with cac:TaxScheme/cbc:ID = "VAT"

Sample of EN-16931 syntax mapping to UBL

In this syntax mapping sample, you see the elements from the semantic model on the left, and how they map to UBL on the right. Note that where there is not a 1:1 match, the EN also specifies what limited mismatches are allowed, and how these should be treated, in part 3-1.

For example, the cardinality of `BT-28`, the Seller Trading Name, is 0..1 in EN-16931, meaning that specifying the name of the seller is optional, and a seller can have only one name. In UBL, the name is optional too, but the seller can have multiple names. The final cardinality for the mapping is therefor 0..1, and this means that a validation rule must be added to UBL to check that a seller's name is limited to the one element (mismatch type `CAR-3`).

2.5 Customization of EN-16931

The European Norm also provides a structured way to create more specific variants, in the way of 'Core Invoice Usage Specifications' (CIUSes) and Extensions:

- A CIUS may add further restrictions on documents:
  - may remove optional fields
  - may make optional fields mandatory
  - may introduce stricter business rules

    - may choose subsets for code lists
    - a CIUS may *not* expand code lists, add elements, make mandatory elements optional, or make rules more lenient
- An Extension can do all of that: add new elements, expand code lists, increase cardinality, etc.

This means that any document that is compliant to a CIUS of the European Norm is also compliant to the European Norm, whereas documents that are compliant to the European Norm are not always compliant to a CIUS.

The extension mechanism is there for very specific sectors: it is a way to create a standard that adds some things, while still being ‘mostly’ compatible to the EN, without burdening other users of the EN with that highly specific data. Two examples from the Netherlands are the Standard Energy E-Invoice (which adds meter data for the energy sector in the netherlands), and the G-account extension, which adds a second set of payment instruction, for a tax construct that is very specific and almost exclusively used in the temporary employment industry.

If at all possible, it’s generally better to create a CIUS if you have specific requirements.

Peppol BIS 3 is a CIUS of the European Norm. Therefore, all valid Peppol BIS 3 documents are compliant to EN-16931.

There are other CIUSes as well; CEF maintains a registry of of CIUSes and Extensions at [EN-16931-CIUSes]

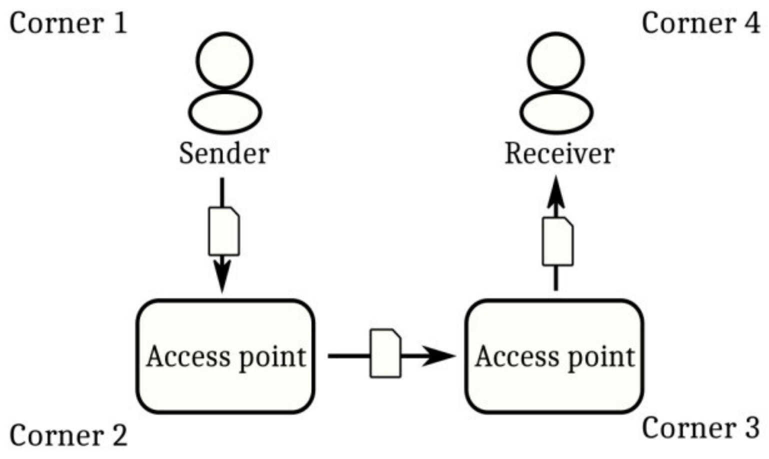
2.6 Peppol

Peppol (originally PEPPOL, an acronym for Pan-European Public Procurement On-Line, though the acronym has been dropped, and the name is just Peppol now) [PEPPOL], is actually three things in one:

1. The Peppol Interoperability framework, a set of legal agreements, policies, requirements and specifications. of functional and non-functional rules to which the participants agree, including:
2. The Peppol Network: the services and servers that perform the actual exchange of business documents
3. Peppol BIS (Business Interoperability Specifications): a set of document types for the exchange of business information; e-invoices, e-orders, catalogues, etc.

‘Peppol’ as a name, is often used as a shorthand for all three of these things, so depending on the context, Peppol may mean any of them.

To connect to the Peppol network, you either become a Peppol Service Provider, or you sign a contract with one. Peppol Service Providers operate Peppol Access Points, which are the servers that perform the actual exchange of documents.



The Peppol 4-corner model

Through a Peppol Access Point, any connected user can send documents to, and receive documents from, any other connected user, regardless of who their Peppol Service Provider is.

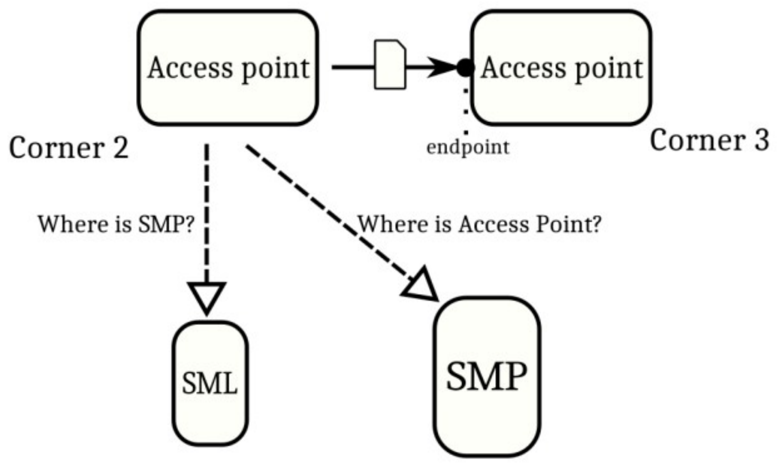
A Peppol Service Provider signs a contract with a national Peppol Authority (or the central Peppol Coordinating Authority, if there is no national authority), to join the agreement framework. This contract contains a number of legal and operational requirements. For example, there are service level agreements regarding the availability of access points, the service provider may not send invalid documents on the network, and the service provider must check that a participant is allowed to send a particular document (e.g. that they represent the company they claim to represent).

There are also requirements to support specific document types. The Peppol BIS documents are not the only types that are allowed, there are many more (see [PEPPOL-DOCUMENT-TYPES] for a full list).

However, as per the agreement, if, as a participant, you support a specific document *class* (such as e-invoices) of which there is a Peppol BIS equivalent, you MUST also support the Peppol BIS version. So for instance, you are allowed to use Xrechnung invoices, but if you do, you must also support Peppol BIS invoices. This ensures the ‘connect once, reach anyone’ principle.

You are free to use the Peppol BIS format outside of the Peppol network as well, but in that case neither you nor the party you exchange the document with are bound by the other requirements from the Peppol Interoperability Framework.

The Peppol network provides a way to publish which document types (and related business processes) a participant supports. See [ION-SMP-ROLE]. The Service Provider generally takes care of this publication.

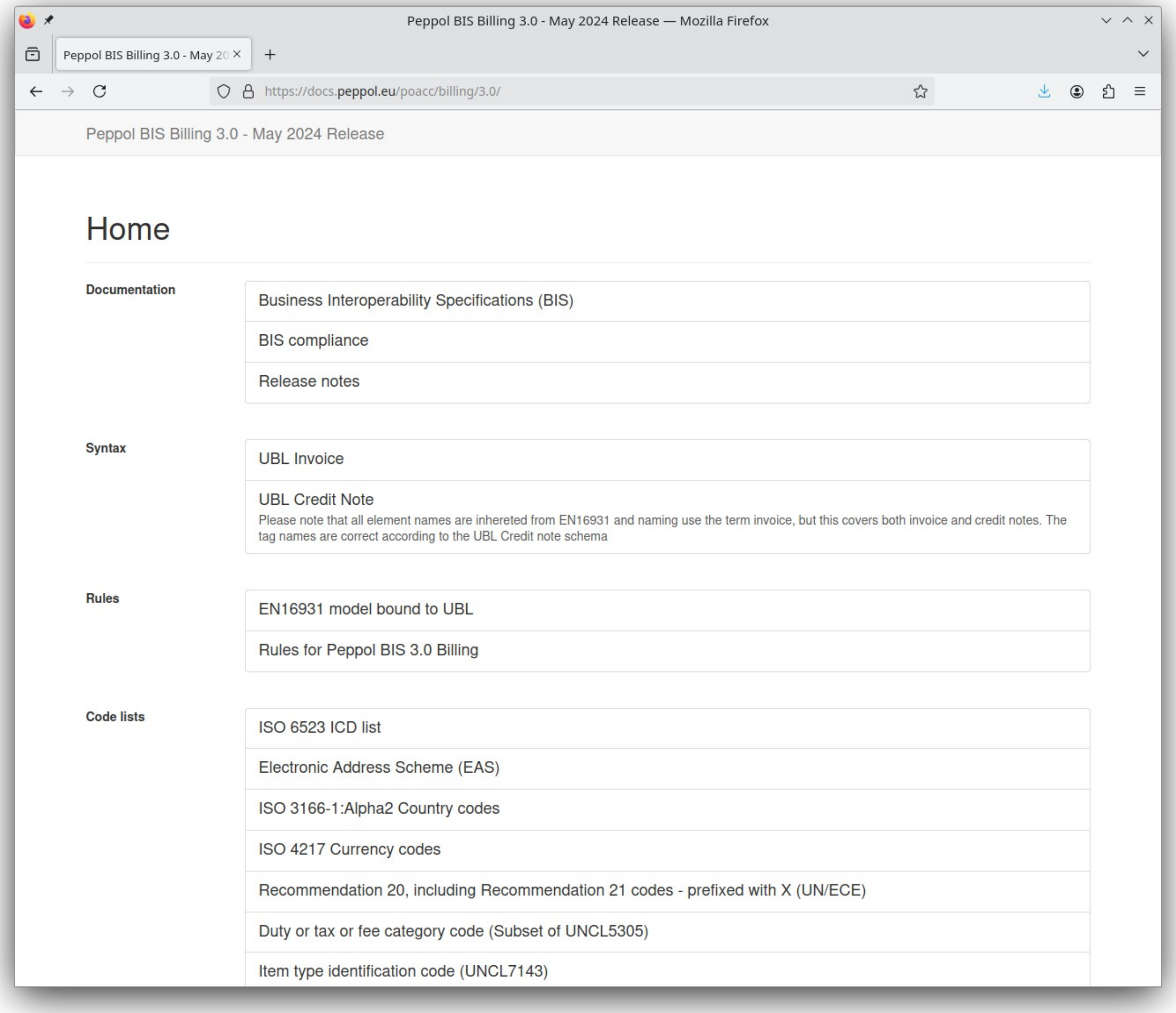


Peppol Capability Lookup

3 Peppol BIS Invoices - how to read the specification

3.1 Documentation overview

The Documentation page for Peppol BIS Invoices (and, actually, Credit Notes), comprises several different parts [PEPPOL-BIS-3-BILLING]:


Peppol BIS Billing 3.0 - May 2024 Release — Mozilla Firefox
Peppol BIS Billing 3.0 - May 20
https://docs.peppol.eu/poacc/billing/3.0/
Peppol BIS Billing 3.0 - May 2024 Release

Home

Documentation	Business Interoperability Specifications (BIS)
	BIS compliance
	Release notes
Syntax	UBL Invoice
	UBL Credit Note Please note that all element names are inhereted from EN16931 and naming use the term invoice, but this covers both invoice and credit notes. The tag names are correct according to the UBL Credit note schema
Rules	EN16931 model bound to UBL
	Rules for Peppol BIS 3.0 Billing
Code lists	ISO 6523 ICD list
	Electronic Address Scheme (EAS)
	ISO 3166-1:Alpha2 Country codes
	ISO 4217 Currency codes
	Recommendation 20, including Recommendation 21 codes - prefixed with X (UN/ECE)
	Duty or tax or fee category code (Subset of UNCL5305)
	Item type identification code (UNCL7143)

Screenshot of documentation overview page

1. The main BIS specification, which describes the surrounding processes, and business rules. This part also contains a number of examples for specific sections of the invoice.

2. The syntax. This part contains the documentation of the actual mapping, by listing all the allowed XML elements, their cardinality, their meaning, and the validation rules that are relevant to those elements.
3. The rules. This is a list of validation rules, containing the rule description, and the actual schematron definition of the rule (e.g. how it's validated when using schematron validtion)
4. Code lists. A number of elements are restricted to a specific code list. This part lists all the used code lists.
5. Downloads. The downloads section contains the standard validation files you can use (apart from the UBL XML Schema's, you'll need to procure those from Oasis), and a number of example files.

The following sections contain more information about each of these.

3.2 Peppol BIS main specification

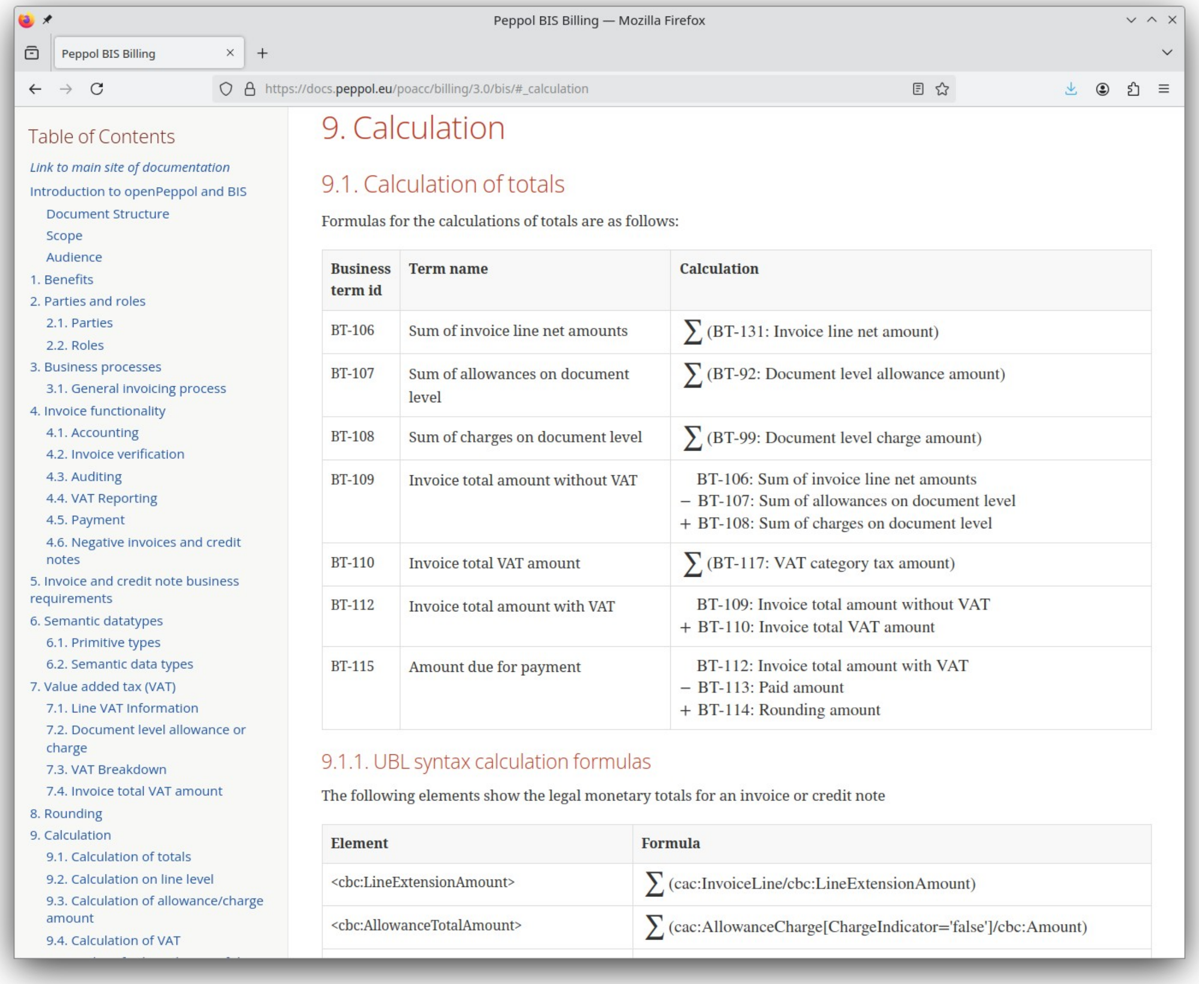

Peppol BIS Billing — Mozilla Firefox
Peppol BIS Billing
https://docs.peppol.eu/poacc/billing/3.0/bis/#_calculation

Table of Contents

9. Calculation

9.1. Calculation of totals

Formulas for the calculations of totals are as follows:

Business term id	Term name	Calculation
BT-106	Sum of invoice line net amounts	$\sum$(BT-131: Invoice line net amount)
BT-107	Sum of allowances on document level	$\sum$(BT-92: Document level allowance amount)
BT-108	Sum of charges on document level	$\sum$(BT-99: Document level charge amount)
BT-109	Invoice total amount without VAT	BT-106: Sum of invoice line net amounts − BT-107: Sum of allowances on document level + BT-108: Sum of charges on document level
BT-110	Invoice total VAT amount	$\sum$(BT-117: VAT category tax amount)
BT-112	Invoice total amount with VAT	BT-109: Invoice total amount without VAT + BT-110: Invoice total VAT amount
BT-115	Amount due for payment	BT-112: Invoice total amount with VAT − BT-113: Paid amount + BT-114: Rounding amount

9.1.1. UBL syntax calculation formulas

The following elements show the legal monetary totals for an invoice or credit note

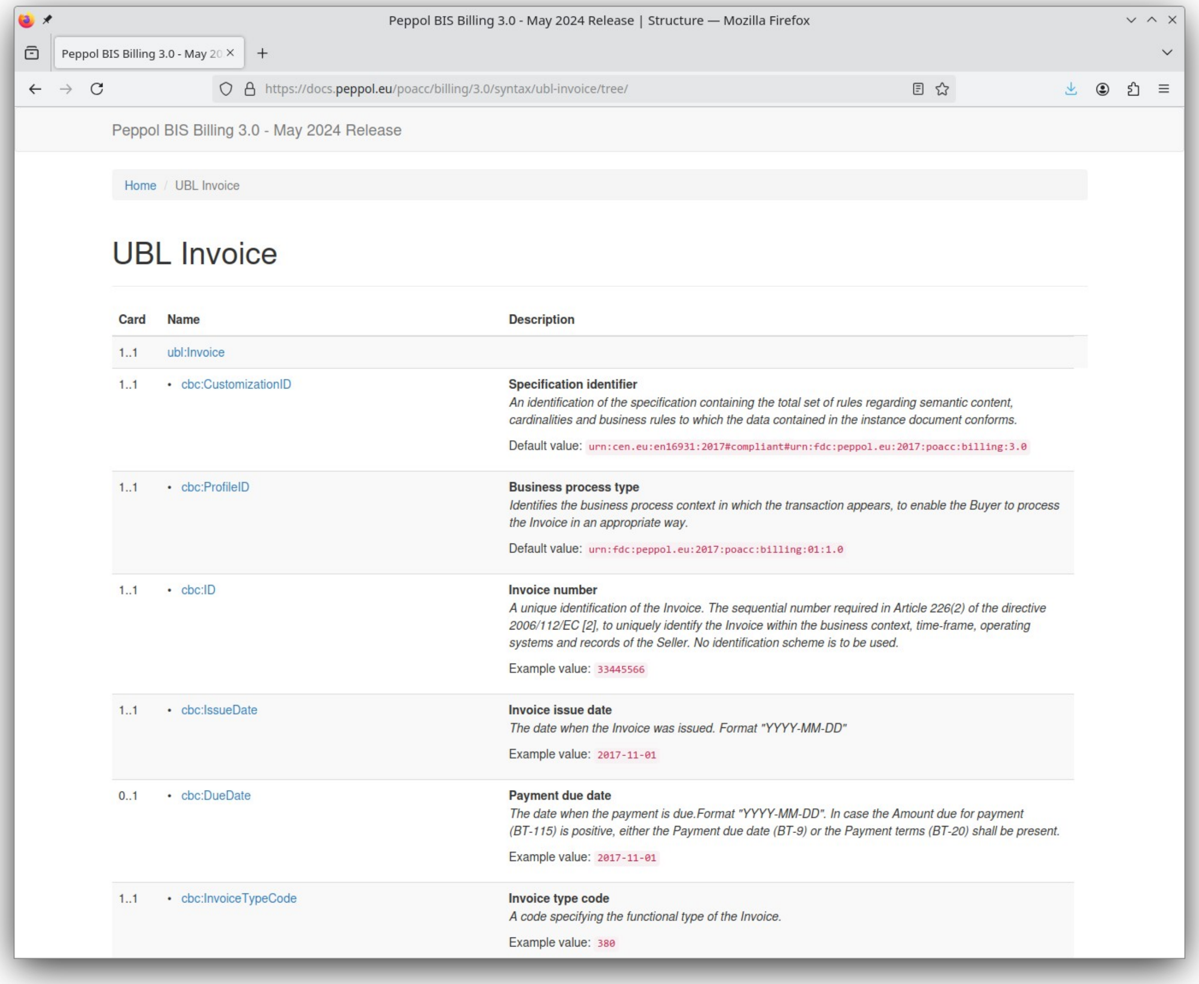
Element	Formula
<cbc:LineExtensionAmount>	$\sum$(cac:InvoiceLine/cbc:LineExtensionAmount)
<cbc:AllowanceTotalAmount>	$\sum$(cac:AllowanceCharge[ChargeIndicator='false']/cbc:Amount)

Screenshot of main documentation page

This is the main specification document for Peppol BIS Billing, i.e. for invoices and credit notes. It specifies the data model, and which elements of EN-16931's model are used for Peppol BIS. It describes the roles that it supports within business processes, and the functionality of Peppol BIS within those processes. It also defines the rules and restrictions for the data elements in the model.

3.3 Peppol BIS Billing syntax

The syntax documentation describes the XML elements that are used in the (mandatory) UBL syntax mapping.


Peppol BIS Billing 3.0 - May 2024 Release | Structure — Mozilla Firefox
Peppol BIS Billing 3.0 - May 20
https://docs.peppol.eu/poacc/billing/3.0/syntax/ubl-invoice/tree/
Peppol BIS Billing 3.0 - May 2024 Release
Home / UBL Invoice

UBL Invoice

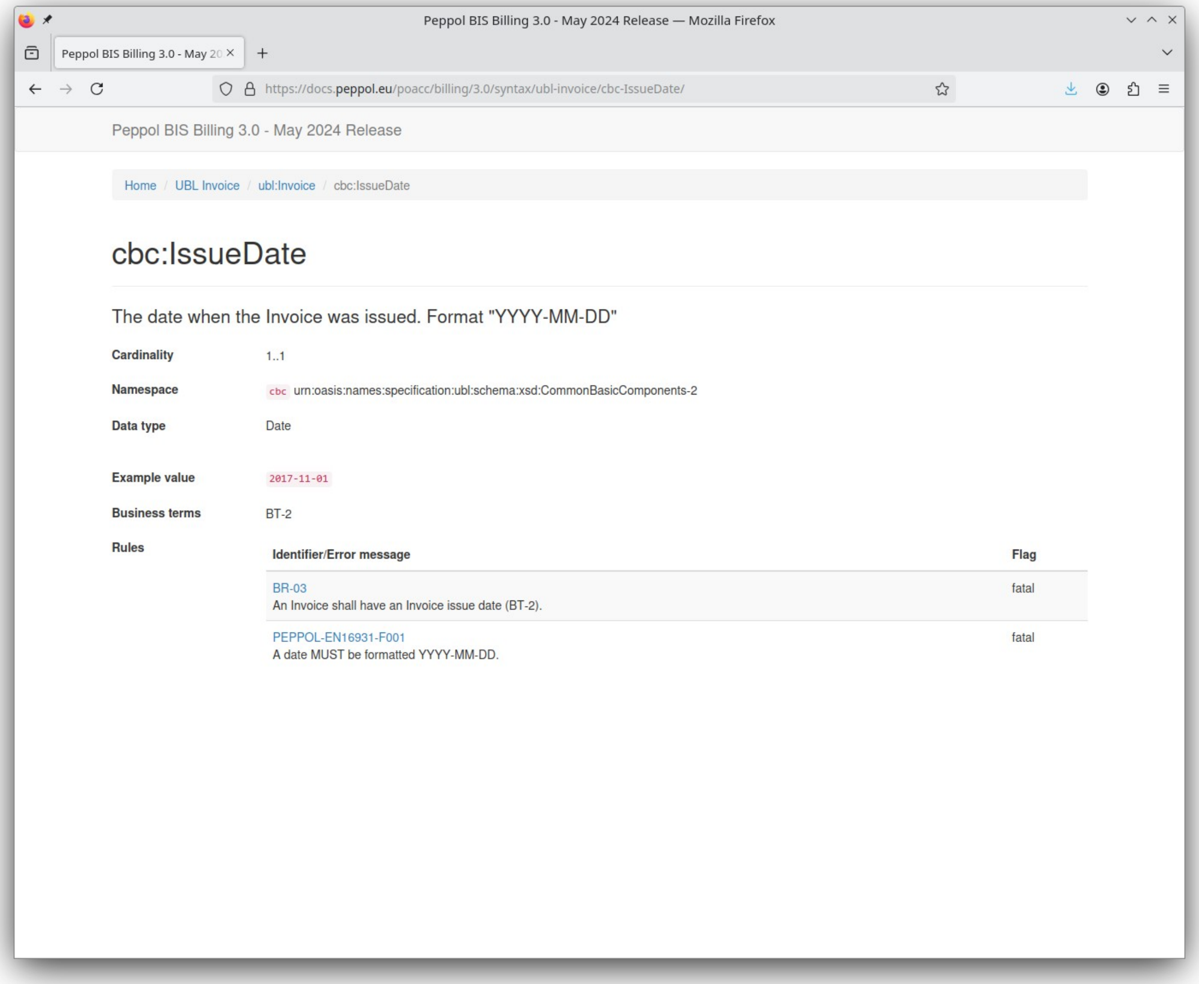
Card	Name	Description
1..1	ubl:Invoice	
1..1	• cbc:CustomizationID	**Specification identifier** *An identification of the specification containing the total set of rules regarding semantic content, cardinalities and business rules to which the data contained in the instance document conforms.* Default value: `urn:cen.eu:en16931:2017#compliant#urn:fdc:peppol.eu:2017:poacc:billing:3.0`
1..1	• cbc:ProfileID	**Business process type** *Identifies the business process context in which the transaction appears, to enable the Buyer to process the Invoice in an appropriate way.* Default value: `urn:fdc:peppol.eu:2017:poacc:billing:01:1.0`
1..1	• cbc:ID	**Invoice number** *A unique identification of the Invoice. The sequential number required in Article 226(2) of the directive 2006/112/EC [2], to uniquely identify the Invoice within the business context, time-frame, operating systems and records of the Seller. No identification scheme is to be used.* Example value: `33445566`
1..1	• cbc:IssueDate	**Invoice issue date** *The date when the Invoice was issued. Format "YYYY-MM-DD"* Example value: `2017-11-01`
0..1	• cbc:DueDate	**Payment due date** *The date when the payment is due.Format "YYYY-MM-DD". In case the Amount due for payment (BT-115) is positive, either the Payment due date (BT-9) or the Payment terms (BT-20) shall be present.* Example value: `2017-11-01`
1..1	• cbc:InvoiceTypeCode	**Invoice type code** *A code specifying the functional type of the Invoice.* Example value: `380`

Screenshot of syntax documentation page

Elements on this page:

1. Cardinality: The cardinality ('Card' in the table) specifies how many times the element may occur as a child of its parent element, see section 2.4 for more information on the notation of cardinality.
2. Level (e.g. is this element nested in the element of the previous level), represented as a number of bullets before the name
3. Element name and namespace, using the conventional UBL namespace prefixes of cbc and cac
4. Description of the element, its semantic meaning, and in some cases, an example value.

Each element has its own page with a more detailed description:


Peppol BIS Billing 3.0 - May 2024 Release — Mozilla Firefox

https://docs.peppol.eu/poacc/billing/3.0/syntax/ubl-invoice/cbc-IssueDate/

Peppol BIS Billing 3.0 - May 2024 Release

Home / UBL Invoice / ubl:Invoice / cbc:IssueDate

cbc:IssueDate

The date when the Invoice was issued. Format "YYYY-MM-DD"

Cardinality	1..1
Namespace	`cbc` urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2
Data type	Date
Example value	`2017-11-01`
Business terms	BT-2
Rules	

Identifier/Error message	Flag
BR-03 An Invoice shall have an Invoice issue date (BT-2).	fatal
PEPPOL-EN16931-F001 A date MUST be formatted YYYY-MM-DD.	fatal

Screenshot of syntax detail page

This page has the following content:

- Next to Home (a link to the start) and UBL Invoice (the document type of the specification we're looking at) is a breadcrumb trail of the current element; i.e. the elements name, preceded by the parent element(s) up to the main element of this document (in this case ubl:Invoice).
- The name of the element again
- A description of the element
- Cardinality: specifies how many times this element may occur within its parent element, see section 2.4 for more information on the notation of cardinality.
- Namespace: the conventional namespace prefix (e.g. cac or cbc), followed by the full XML namespace of this element

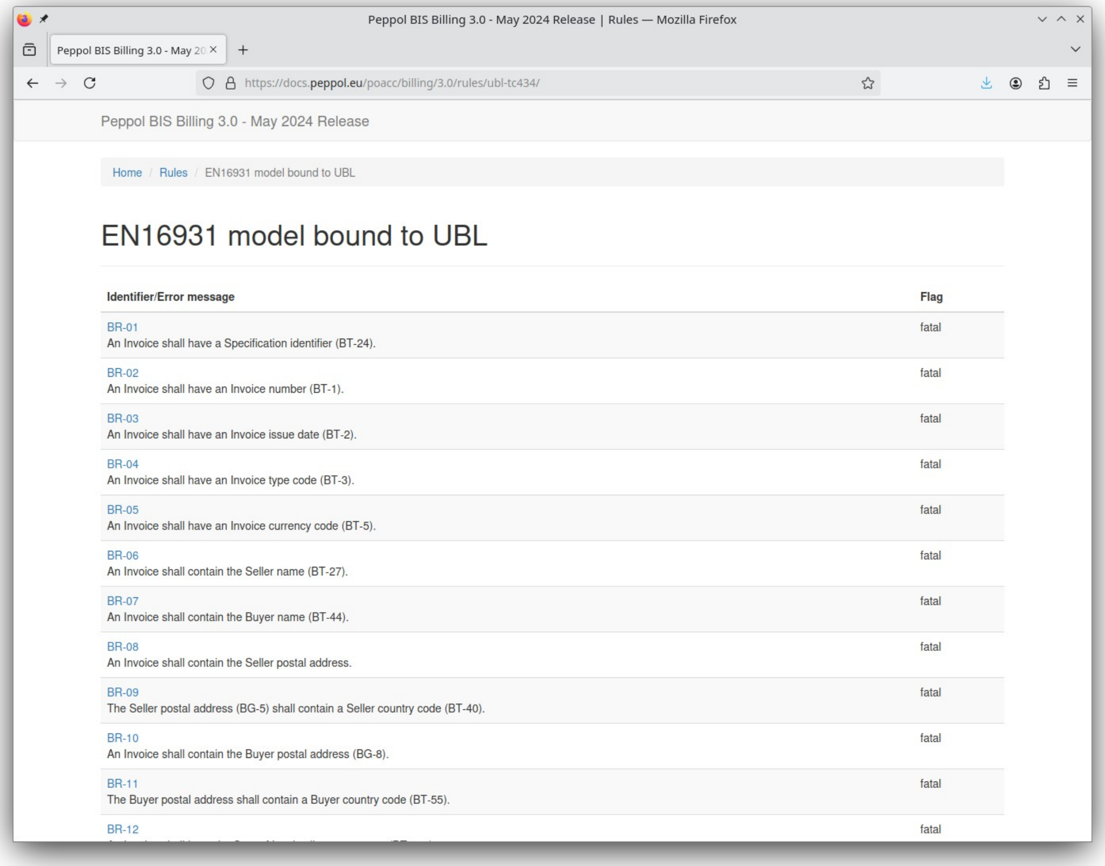
- Data Type: The UBL Data Type of data of this element's content, if this element has direct content. If the element has only child elements, the Type is ommitted. Example types are Text, Date, Amount, etc. UBL Data types are defined in [UBL2.1-Data-Types].
- Example Value: An example value for this element, if it is an element with a value
- Business Terms: The Business Term Identifier(s) this element relates to as per the syntax mapping of the European Norm
- Rules: all validation rules from the specification that are relevant for this element.

3.4 Peppol BIS validation rules

The validation rules were already mentioned in the previous section, but are presented as separate lists for reference. The Peppol documentation provides two lists of validation rules: the rules from EN-16931, and the peppol-specific rules. Since Peppol BIS is a CIUS of EN-16931, all Peppol documents must adhere to the rules from the EN itself. Valid documents must comply to both these rulesets.

Country-specific rules

The Peppol validation rules also contain a number of country-specific rules. These rules have a name that starts with the ISO-3166 Alpha2 country code (e.g. DK, NL, etc). Such rules are determined by the Peppol Authority for that country, but enforced for all validation. Country-specific rules are always limited to the country of the seller, and each country-specific rules must always start with 'For suppliers in *XX*, ...'.



Peppol BIS Billing 3.0 - May 2024 Release | Rules — Mozilla Firefox

Peppol BIS Billing 3.0 - May 20

https://docs.peppol.eu/poacc/billing/3.0/rules/ubl-tc434/

Peppol BIS Billing 3.0 - May 2024 Release

Home / Rules / EN16931 model bound to UBL

EN16931 model bound to UBL

Identifier/Error message	Flag
BR-01 An Invoice shall have a Specification identifier (BT-24).	fatal
BR-02 An Invoice shall have an Invoice number (BT-1).	fatal
BR-03 An Invoice shall have an Invoice issue date (BT-2).	fatal
BR-04 An Invoice shall have an Invoice type code (BT-3).	fatal
BR-05 An Invoice shall have an Invoice currency code (BT-5).	fatal
BR-06 An Invoice shall contain the Seller name (BT-27).	fatal
BR-07 An Invoice shall contain the Buyer name (BT-44).	fatal
BR-08 An Invoice shall contain the Seller postal address.	fatal
BR-09 The Seller postal address (BG-5) shall contain a Seller country code (BT-40).	fatal
BR-10 An Invoice shall contain the Buyer postal address (BG-8).	fatal
BR-11 The Buyer postal address shall contain a Buyer country code (BT-55).	fatal
BR-12	fatal

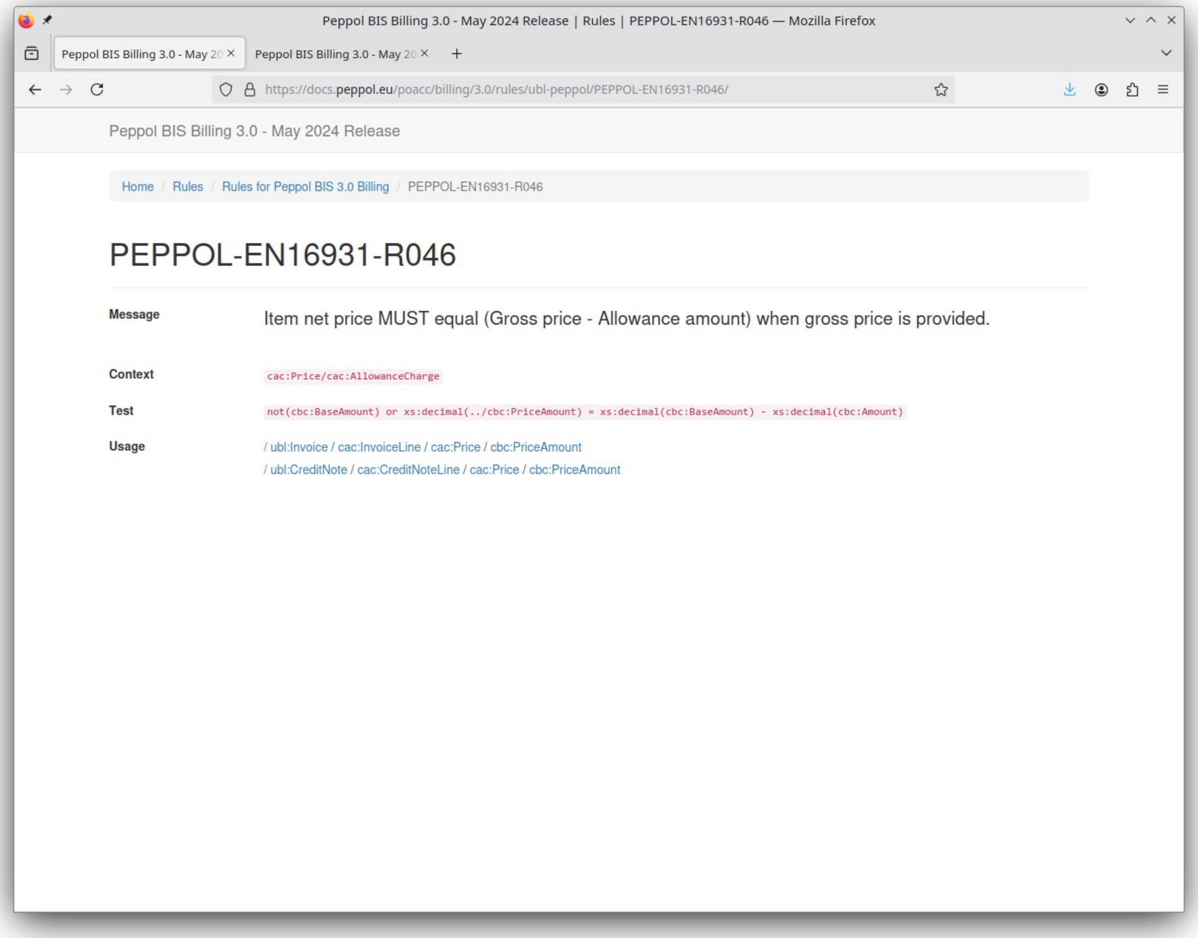
Screenshot of rule list page

The validation rules list pages show the following details for all rules:

- The identifier for the rule. There are a number of conventions used by the rule identifiers to show the origin and scope of the rule:
  - The initial prefix shows the origin of the rule:
    - `BR-` for rules from EN-16931
    - `UBL-` for rules that check for the absence of UBL elements that are **not** defined for Peppol BIS
    - `PEPPOL-` for rules that are Peppol-specific
    - `NL-`, `BE-`, etc, for Peppol's country-specific rules
  - The prefix may be followed by a denominator that indicated the type of rule, e.g. codelist rules have a denominator `-CL-`, and rules pertaining to specific tax categore have the tax category code here (such as `-S-`, `-AE-`, etc).
  - The last part of the rule is an integer to give the rule a unique identifier.

- A textual description of the rule. When using the standard validation files, this is also the error that is generated when the document does not conform to the rule.
- The rule list also contains an indicator whether a violation of the rule is fatal (i.e. the document is not compliant), or just a warning (the document is wrong, but still valid).

Clicking on the rule identifier takes you to the rule details page:



Peppol BIS Billing 3.0 - May 2024 Release | Rules | PEPPOL-EN16931-R046 — Mozilla Firefox

https://docs.peppol.eu/poacc/billing/3.0/rules/ubl-peppol/PEPPOL-EN16931-R046/

Peppol BIS Billing 3.0 - May 2024 Release

Home / Rules / Rules for Peppol BIS 3.0 Billing / PEPPOL-EN16931-R046

PEPPOL-EN16931-R046

Message	Item net price MUST equal (Gross price - Allowance amount) when gross price is provided.
Context	`cac:Price/cac:AllowanceCharge`
Test	`not(cbc:BaseAmount) or xs:decimal(../cbc:PriceAmount) = xs:decimal(cbc:BaseAmount) - xs:decimal(cbc:Amount)`
Usage	/ ubl:Invoice / cac:InvoiceLine / cac:Price / cbc:PriceAmount / ubl:CreditNote / cac:CreditNoteLine / cac:Price / cbc:PriceAmount

Screenshot of rule details page

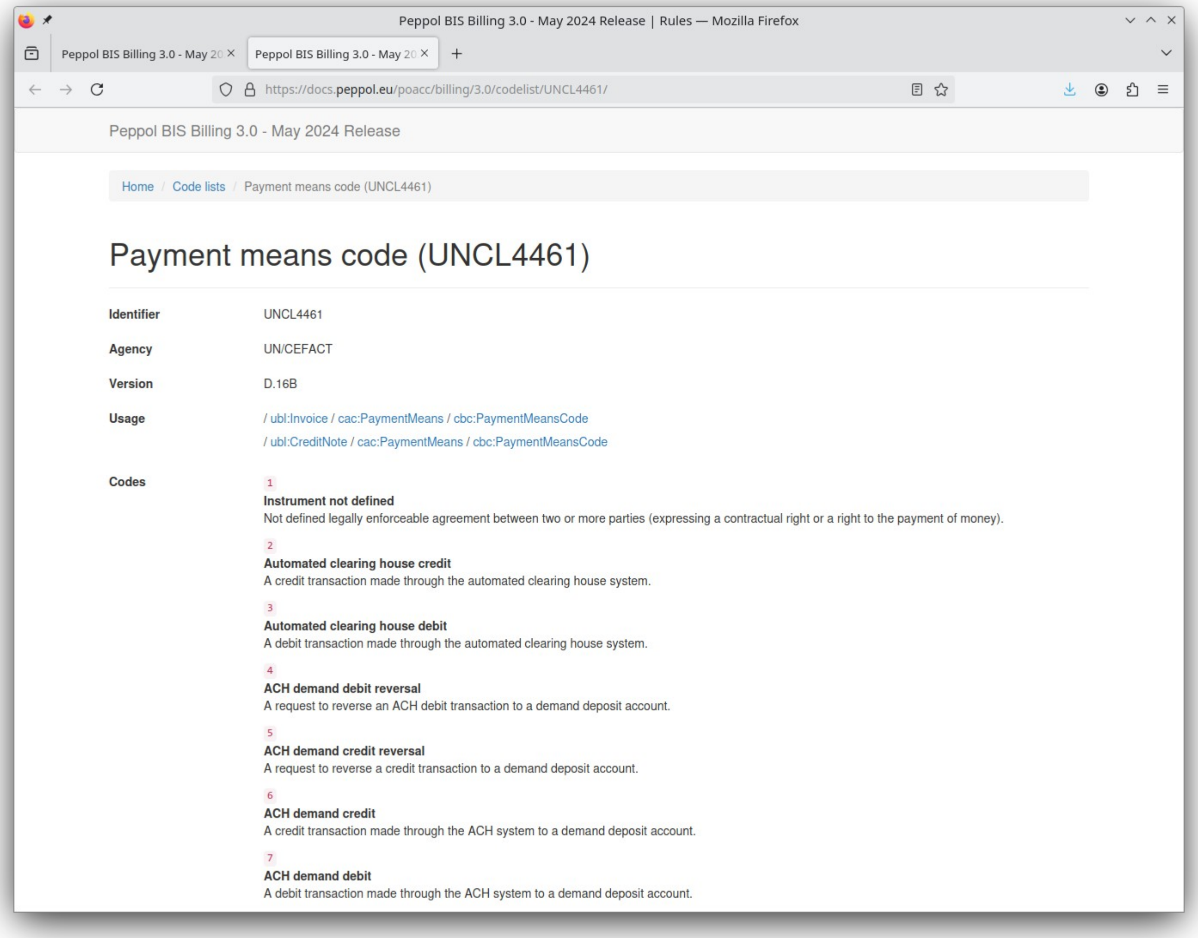
The rule details page shows the following details:

- The identifier of the rule.
- The description of the rule. When using the standard validation files, this is also the error that is generated when the document does not conform to the rule.
- The context of the rule. This relates to how the standard validation files are implemented using the Schematron standard, and is represented as

an XSLT 2 Selector. The rule is applied to all XML elements that match this selector.

- The XSLT Test that implements the rule. This relates to how the standard validation files are implemented using the Schematron standard, and is represented as an XLST 2 expression. The check fails if this expression, when evaluated in specified Context, returns False.
- Usage: this shows the elements where from the Syntax section for which this rule is relevant.

3.5 Peppol BIS code lists



Peppol BIS Billing 3.0 - May 2024 Release | Rules — Mozilla Firefox
Peppol BIS Billing 3.0 - May 20
Peppol BIS Billing 3.0 - May 20
https://docs.peppol.eu/poacc/billing/3.0/codelist/UNCL4461/
Peppol BIS Billing 3.0 - May 2024 Release
Home / Code lists / Payment means code (UNCL4461)

Payment means code (UNCL4461)

Identifier	UNCL4461
Agency	UN/CEFACT
Version	D.16B
Usage	/ ubl:Invoice / cac:PaymentMeans / cbc:PaymentMeansCode / ubl:CreditNote / cac:PaymentMeans / cbc:PaymentMeansCode
Codes	1 **Instrument not defined** Not defined legally enforceable agreement between two or more parties (expressing a contractual right or a right to the payment of money). 2 **Automated clearing house credit** A credit transaction made through the automated clearing house system. 3 **Automated clearing house debit** A debit transaction made through the automated clearing house system. 4 **ACH demand debit reversal** A request to reverse an ACH debit transaction to a demand deposit account. 5 **ACH demand credit reversal** A request to reverse a credit transaction to a demand deposit account. 6 **ACH demand credit** A credit transaction made through the ACH system to a demand deposit account. 7 **ACH demand debit** A debit transaction made through the ACH system to a demand deposit account.

Screenshot of code list page

There are a number of elements in Peppol documents, where the allowed values are restricted to a specific code list. These lists are provided in full in this section of the documentation. These code lists are often maintained by external agencies (such as UN/CEFACT), but some of them are maintained by Peppol itself. In some cases, the main code list is maintained by an external agency, but Peppol only allows a specific subset of the full list.

The code list pages contain the following details:

- The name of the Code list, and if based on a more general list, a reference to the full list.
- Identifier: The identifier of the code list, as it is used in documents when the code list is referred to
- Agency: The agency that maintains the code list
- Version: The version of the code list
- Usage: the elements from the syntax where this code list is used
- Codes: The list of allowed codes, and their description.

4 Peppol BIS Example invoice

```xml
<?xml version="1.0" encoding="utf-8"?>
<Invoice
  xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateCo
mponents-2"
  xmlns:cbc="urn:oasis:names:specification:ubl:schema:xsd:CommonBasicCompon
ents-2"
  xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2">
  <cbc:UBLVersionID>2.1</cbc:UBLVersionID>
  <cbc:CustomizationID>urn:cen.eu:en16931:2017#compliant#urn:fdc:peppol.eu:
2017:poacc:billing:3.0</cbc:CustomizationID>
  <cbc:ProfileID>urn:fdc:peppol.eu:2017:poacc:billing:01:1.0</cbc:ProfileID
>
  <cbc:ID>INVOICE12345</cbc:ID>
  <cbc:IssueDate>2025-01-09</cbc:IssueDate>
  <cbc:DueDate>2025-02-08</cbc:DueDate>
  <cbc:InvoiceTypeCode>380</cbc:InvoiceTypeCode>
  <cbc:DocumentCurrencyCode>EUR</cbc:DocumentCurrencyCode>
  <cac:OrderReference>
    <cbc:ID>47806</cbc:ID>
  </cac:OrderReference>
  <cac:AccountingSupplierParty>
    <cac:Party>
      <cbc:EndpointID schemeID="0106">12345678</cbc:EndpointID>
      <cac:PostalAddress>
        <cbc:StreetName>Peppolstreet 1</cbc:StreetName>
        <cbc:CityName>Businesstown</cbc:CityName>
        <cbc:PostalZone>1111 ZZ</cbc:PostalZone>
        <cac:Country>
          <cbc:IdentificationCode>NL</cbc:IdentificationCode>
        </cac:Country>
      </cac:PostalAddress>
      <cac:PartyTaxScheme>
        <cbc:CompanyID>NL1111.11.111.B.01</cbc:CompanyID>
        <cac:TaxScheme>
          <cbc:ID>VAT</cbc:ID>
        </cac:TaxScheme>
      </cac:PartyTaxScheme>
      <cac:PartyLegalEntity>
        <cbc:RegistrationName>SimplerInvoicing</cbc:RegistrationName>
        <cbc:CompanyID schemeID="0106">12345678</cbc:CompanyID>
      </cac:PartyLegalEntity>
    </cac:Party>
  </cac:AccountingSupplierParty>
  <cac:AccountingCustomerParty>
    <cac:Party>
      <cbc:EndpointID schemeID="9944">NL1234567890B01</cbc:EndpointID>
      <cac:PostalAddress>
        <cbc:StreetName>Teststreet 123</cbc:StreetName>
        <cbc:CityName>Testcity</cbc:CityName>
        <cbc:PostalZone>1111 AA</cbc:PostalZone>
        <cac:Country>
          <cbc:IdentificationCode>NL</cbc:IdentificationCode>
        </cac:Country>
      </cac:PostalAddress>
      <cac:PartyLegalEntity>
        <cbc:RegistrationName>Buyers Inc.</cbc:RegistrationName>
        <cbc:CompanyID schemeID="0106">11111111</cbc:CompanyID>
      </cac:PartyLegalEntity>
    </cac:Party>
  </cac:AccountingCustomerParty>
  <cac:PaymentMeans>
    <cbc:PaymentMeansCode>30</cbc:PaymentMeansCode>
    <cbc:PaymentID>Deb. 10202 / Fact. 12115118</cbc:PaymentID>
    <cac:PayeeFinancialAccount>
      <cbc:ID>NL11 BANK 1111111111</cbc:ID>
    </cac:PayeeFinancialAccount>
  </cac:PaymentMeans>
  <cac:TaxTotal>
    <cbc:TaxAmount currencyID="EUR">42.00</cbc:TaxAmount>
    <cac:TaxSubtotal>
      <cbc:TaxableAmount currencyID="EUR">200.00</cbc:TaxableAmount>
      <cbc:TaxAmount currencyID="EUR">42.00</cbc:TaxAmount>
      <cac:TaxCategory>
        <cbc:ID>S</cbc:ID>
        <cbc:Percent>21</cbc:Percent>
        <cac:TaxScheme>
          <cbc:ID>VAT</cbc:ID>
        </cac:TaxScheme>
      </cac:TaxCategory>
    </cac:TaxSubtotal>
  </cac:TaxTotal>
  <cac:LegalMonetaryTotal>
    <cbc:LineExtensionAmount
currencyID="EUR">200.00</cbc:LineExtensionAmount>
    <cbc:TaxExclusiveAmount
currencyID="EUR">200.00</cbc:TaxExclusiveAmount>
    <cbc:TaxInclusiveAmount
currencyID="EUR">242.00</cbc:TaxInclusiveAmount>
    <cbc:PayableAmount currencyID="EUR">242.00</cbc:PayableAmount>
  </cac:LegalMonetaryTotal>
  <cac:InvoiceLine>
    <cbc:ID>1</cbc:ID>
    <cbc:InvoicedQuantity unitCode="C62">1</cbc:InvoicedQuantity>
    <cbc:LineExtensionAmount
currencyID="EUR">200.00</cbc:LineExtensionAmount>
    <cac:Item>
      <cbc:Name>Office Supplies</cbc:Name>
      <cac:ClassifiedTaxCategory>
        <cbc:ID>S</cbc:ID>
        <cbc:Percent>21</cbc:Percent>
        <cac:TaxScheme>
          <cbc:ID>VAT</cbc:ID>
        </cac:TaxScheme>
      </cac:ClassifiedTaxCategory>
    </cac:Item>
    <cac:Price>
      <cbc:PriceAmount currencyID="EUR">200.00</cbc:PriceAmount>
    </cac:Price>
  </cac:InvoiceLine>
</Invoice>
```

5 Validating invoices

On of the requirements in the Peppol agreements is that it is not allowed to send documents on the network that are not compliant to the validation rules of the document type in question. This means that, for a Service Provider, or a software package, to be compliant to the Peppol standards, it is wise to validate every document before sending it to the network.

Peppol does not mandate *how* you validate your document, as long as whatever you send is compliant to both the structure of the syntax and all validation rules in the specification.

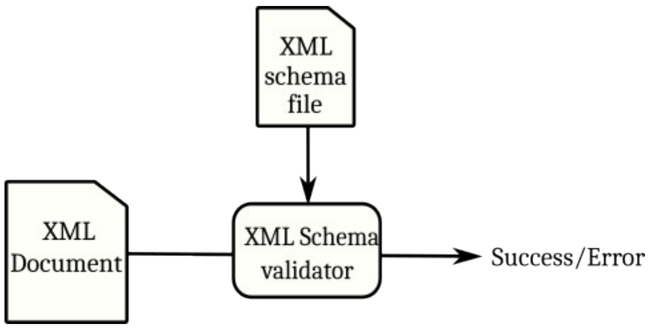
That said, Peppol does provide validation artefacts that everyone can use to validate Peppol BIS documents, for the rules from EN-16931 as well as the Peppol rules. These validation artefacts are published in the form of Schematron files.

This section is a summary of an article we have written before, which you can find here: [IONITE-VALIDATION]

5.1 XML Schema

The XML Schema that is used for Peppol BIS document is that of UBL 2.1 itself; there are no official XML Schema definitions for just the elements used in Peppol BIS (or EN-16931, for that matter).

XML Schema validation is generally pretty straightforward: Many XML libraries offer this functionality, and it is often simply a matter of loading the XML Schema (`.xsd`) file and providing the document to validate, upon which the validator returns an error or success response.



XML Schema validation

Note that the UBL 2.1 XML Schemas are distributed as a file tree: there are separate files for all the main UBL document types (such as Invoice, CreditNote, and Order), and a number of shared files, which are included by the main document files. You'll need to fetch the xsd files for the main document types you want to validate from the `maindoc/` directory, and get *all* the files from the `common/` directory, as these are included by the main document xsd files.

5.2 Schematron

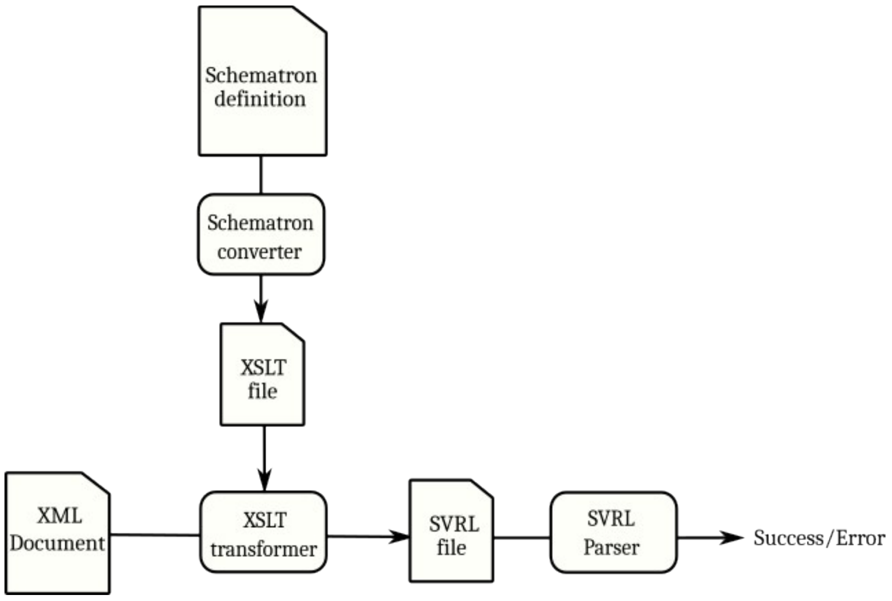
For many validation rules in business documents, XML Schemas do not quite suffice: often, these rules require some calculation or information about other values, and go beyond the mere structure of the document. For instance, a rule that the invoice total must equal the sum of the invoice lines (plus charges minus allowances), cannot be expressed in an XML Schema.

Therefore, many official validation files use a different standard: Schematron.

> Schematron is a rule-based validation language for making assertions about the presence or absence of patterns in XML trees. It is a structural schema language expressed in XML using a small number of elements and XPath languages. In many implementations, the Schematron XML is processed into XSLT code for deployment anywhere that XSLT can be used.
>
> Schematron is capable of expressing constraints in ways that other XML schema languages like XML Schema and DTD cannot. For example, it can require that the content of an element be controlled by one of its siblings. Or it can request or require that the root element, regardless of what element that is, must have specific attributes. Schematron can also specify required relationships between multiple XML files. Constraints and content rules may be associated with "plain-English" (or any language) validation error messages, allowing translation of numeric Schematron error codes into meaningful user error messages.

Schematron validation is slightly more complicated than XML Schema validation: Schematron definitions are generally not used directly (though some implementations do support this), instead, they are transformed into XSLT (eXtensible Stylesheet Language) files, which can be used by any XSLT transformer to transform a given document into an SVRL (Schematron Validation Report Language) document.

Simply said, this is a new XML document that contains a list of warnings and errors, about the XML document. If the document adheres to all rules defined in the Schematron file, these lists are empty. By checking for the presence or absence of errors in the SVRL result file, you can check whether a given XML document is valid or not.



XML Schema validation

Full validation toolkits tend to provide something that can be used in a similar way to XML Schema validation, e.g. you load the schematron or XSLT file, provide the document to validate, and it will return a success or error response.

You can also use a standard XML toolkit which provides XSLT functionality, in which case your program must interpret the resulting XML file yourself. Do note that the schematron validation files for both EN-16931 and Peppol BIS require that the XSLT transformator supports XSLT 2.

5.3 Steps to validate a Peppol BIS document

When validating documents yourself, using the official validation files, it is important to keep in mind that the Schematron definitions act on the presumption that XML Schema validation has already been performed; while they have a number of existence/nonexistence checks themselves, many rules assume that the overall structure of the document is already known to be valid.

Another thing to keep in mind is that the EN-16931 rules and the Peppol BIS rules are distributed separately, in the case of Peppol BIS. You will need to check both.

For validating Peppol BIS documents, this means that there are 3 individual steps to perform:

1. Validate the structure itself, using the general UBL 2.1 XML Schema definition [UBL-2.1-INVOICE]
2. Validate the EN-16931 rules, using the schematron file, or the XSLT file derived from it [SCHEMATRON-VALIDATION-EN16931]
3. Validate the Peppol BIS rules, using the schematron file, or the XSLT file derived from it [SCHEMATRON-VALIDATION-PEPPOLBIS]

5.4 Steps to validate other documents

As mentioned, there are many other document types allowed on the Peppol network as well. Some of these are based on UBL or CII as well, while others use a completely different structure. Similarly, for some of these, there are schematron files, while for others, there are not.

Peppol does not publish the specification, nor any available validation files for document types that are not issues by Peppol. You will have to find the requirements, and any official validation files, at the organization that maintains the document type standard.

5.5 Online validators

There are several tools online that can help you validate documents. These are not intended to provide production-level service, but can be of use when developing a specific document type and you want to find out whether your output is correct.

- The NPA Peppol Test Tool [NPA-PEPPOL-TESTTOOL]
- Peppol Practical - Document Validation [PEPPOL-PRACTICAL-VALIDATOR]:

There are a number of companies that do provide production-level validation, but these generally require a contract. Search for ‘Validate Peppol Documents’ and you will get several relevant results.

5.6 Validation libraries

If you want to integrate document validation in your own solution, but do not want to build it completely, the following libraries might help:

- ion-docval [ION-DOCVAL]
- PHIVE [PHIVE-VALIDATOR]

Both libraries are open source.

6 How to get started on an implementation

If you have an existing system that manages invoice data, and wish to create (valid) Peppol BIS invoices, the most comprehensive approach would be to thoroughly read all the relevant specifications, starting with UBL and the European Norm, then map your internal data structures on the existing syntax mapping, and check that mapping against the requirements, check the implementation against all rules, and identify and fill in any implementation gaps.

That might, however, be more of a long-term approach. In order to get some faster results, a more gradual approach may be more practical.

First of all, find some examples of Peppol BIS documents. Ideally, some examples of valid and invalid documents, although given the rules, it should be relatively easy to create an invalid document based on a valid one. Peppol provides a ZIP file with some examples [PEPPOL-EXAMPLE-FILES].

Then, make sure you have a way to validate any file that your software creates, as described in section 5 . When testing your validator, make sure to test it correctly identifies non-compliant documents.

At this point, it is wise to make sure you at least have access to the relevant standards and skim through them, so that when specific questions regarding the requirements do arise, you'll know where to find them.

Then you can start on creating documents yourself; given the examples, see which data from your software system is required for the most minimal Peppol BIS document, and run the output against the validator. Depending on the existing serialization options your software presumably already supports, this could be a full-fledged XML factory, but there's nothing wrong with template-based system either.

Validate your output, and the most important variants;

- are all required elements always present?
- might codelist-based elements receive values outside of the code list? Think of currencies, countries, ISO6523 ICD schemes [ISO-6523], etc.

Ideally, there should be automated testing of the full ranges of options, but this may not always be practical. It is, however, wise to put validation in the main process of creation, to make sure the system fails early when invalid output would be produced.

At this point your system should be able to produce valid basic Peppol BIS invoices. Now you can start adding additional features and data elements your system supports, by adding them as optional values to the factory or template(s). With each addition, make sure the range of possible vlaues is tested for validity.

When in doubt, refer to the syntax and rules documentation as mentioned in sections 3.3 and 3.4 , or the main documentation of Peppol BIS and EN-16931 itself [PEPPOL-BIS-3-BILLING] [EN-16931].

7 References

[UBL]: Universal Business Language https://groups.oasis-open.org/communities/tc-community-home2?CommunityKey=556949c8-dac8-40e6-bb16-018dc7ce54d6

[CII]: Cross-Industry Invoice https://unece.org/trade/uncefact/e-invoice

[UBL2.1]: Universal Business Language version 2.1 http://docs.oasis-open.org/ubl/os-UBL-2.1/UBL-2.1.html

[ION-SMP-ROLE]: The Role of an SMP https://ion-smp.net/documentation/about/role

[EN-16931]: European Norm EN-16931 https://ec.europa.eu/digital-building-blocks/sites/display/DIGITAL/Obtaining+a+copy+of+the+European+standard+on+eInvoicing

[EN-16931-CIUSes]: Full list of CIUSes and Extensions of EN-16931 https://ec.europa.eu/digital-building-blocks/sites/display/EINVCOMMUNITY/Registry+of+CIUS+%28Core+Invoice+Usage+Specifications%29+and+Extensions

[PEPPOL]: Peppol http://peppol.eu

[PEPPOL-BIS-3-BILLING]: Peppol BIS 3 Billing - Documentation https://docs.peppol.eu/poacc/billing/3.0

[PEPPOL-DOCUMENT-TYPES]: Peppol Code Lists - Document Types https://docs.peppol.eu/edelivery/codelists/v9.0/Peppol%20Code%20Lists%20-%20Document%20types%20v9.0.html

[UBL-2.1-INVOICE]: UBL 2.1 Invoice schema files https://docs.oasis-open.org/ubl/UBL-2.1.html#T-INVOICE

[DATYPIC-UBL2.1-INVOICE] Description of UBL 2.1 Invoice Elements http://www.datypic.com/sc/ubl21/e-ns39_Invoice.html

[UBL2.1-DATA-TYPES]: UBL (2.1) Data Types http://docs.oasis-open.org/ubl/os-UBL-2.1/UBL-2.1.html#A-DATA-TYPE-QUALIFICATIONS-IN-UBL

[IONITE-VALIDATION]: Validating Peppol Documents https://ionite.net/news-articles/2023-08-17_validating_peppol_documents

[SCHEMATRON-VALIDATION-EN16931]: Schematron file for EN-16931 https://docs.peppol.eu/poacc/billing/3.0/files/CEN-EN16931-UBL.sch

[SCHEMATRON-VALIDATION-PEPPOLBIS]: Schematron file for Peppol BIS Billing https://docs.peppol.eu/poacc/billing/3.0/files/PEPPOL-EN16931-UBL.sch

[NPA-PEPPOL-TESTTOOL]: NPA Peppol Test Tool - Validator https://test.peppolautoriteit.nl/validate

[PEPPOL-PRACTICAL-VALIDATOR]: Peppol Practical - Document Validation https://peppol.helger.com/public/locale-en_US/menuitem-validation

[ION-DOCVAL]: ion-docval validation toolkit https://ion-docval.ionite.net

[PHIVE-VALIDATOR]: PHIVE - Integrative Validation Engine https://github.com/phax/phive

[PEPPOL-EXAMPLE-FILES]: https://docs.peppol.eu/poacc/billing/3.0/files/BIS-Billing3-Examples.zip

[ISO-6523]: ISO 6523 ICD Scheme list http://iso6523.info