Specification

Project Acronym:	**PEPPOL**
Grant Agreement number:	**224974**
Project Title:	**Pan-European Public Procurement Online**

PEPPOL Post Award eProcurement ICT - Services and Components

Validation Artefacts for Post Award BIS

Revision: 2.0
Status: In use

Author:
Oriol Bausà Peris, Invinet

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	**Public**	**X**
C	**Confidential, only for members of the consortium and the Commission Services**	

Revision History

Revision	Date	Author	Organisation	Description
1.0	20101030	Oriol Bausà Peris	Invinet	First version
2.0	20120415	Bergthor Skulason	NITA	Extend to all post award BISs
2.0	20120420	Oriol Bausà Peris	Invinet	Update artefact lists, review.

Contributors

Organisations

DIFI (Direktoratet for forvaltning og IKT)[1], Norway, www.difi.no
NITA (IT- og Telestyrelsen)[2], Denmark, www.itst.dk
PEPPOL.AT/BRZ (Bundesrechenzentrum)[3], Austria, http://www.brz.gv.at/

Persons

Bergtor Skulason, NITA
Georg Birgisson, Eykur
Oriol Bausà Peris, Invinet (editor)
Philip Helger, BRZ
Susanna Kamptner, BRZ

[1] English: Agency for Public Management and eGovernment
[2] English: National IT- and Telecom Agency
[3] English: Austrian Federal Computing Centre

Table of Contents

1 Introduction

This document is a result of work within PEPPOL project and is published as part of PEPPOL specifications and is a part of a set of specifications for implementing a PEPPOL business process.

This document defines PEPPOL conformance and architecture for validation of documents conformance. The goal is to ensure that implementers do:

- Produce valid and interoperable electronic document instances
- Correctly process electronic documents produced by third parties.

This document is based on the CWA 16073, in particular its 16073-3 Toolbox Requirements Annex B that defines CEN BII Conformance Testing where the concepts of conformance and validation architecture are fully explained.

1.1 Audience

The audience for this document is organizations wishing to be PEPPOL enabled for exchanging invoices and related documents, with their ICT-suppliers. These organizations may be:

- Service providers
- Contracting Authorities
- Economic Operators
- Software Developers

More specifically it is addressed towards the following roles:

- ICT Architects
- ICT Developers
- Business Experts

2 Conformance

2.1 CEN BII Conformance

An XML instance is conformant to BII (CWA 16073) if its structure fits within the structure identified in the **core** or **full** transaction data models published CEN BII workshop[4].

The normative specifications published by BII are the profile specification documents and their data models. The CEN BII did not publish normative technical artefacts for validating the correctness of individual implementations; instead it is up to the implementer to correctly apply the normative specifications in his technical environment.

BII deliverables are syntax neutral and this means that the requirements identified in BII are valid for wide range of electronic invoice documents and can be expressed in different syntaxes such as OASIS UBL, UN/CEFACT CII or ISO 20022. As BII can be bound to multiple syntaxes, different artefacts can be created to apply the BII requirements.

Regarding invoices, currently two syntax mappings of BII deliverables are available, one for UBL 2.0 and the other for Cross Industry Invoice. Nevertheless, as stated above, BII did not create any artefacts that allow ensuring a UBL Invoice instance or a CII Invoice instance is BII Conformant.

Besides the information contents and structure of the data model, BII defined business rules for the invoice document model. Rules such as co-occurrence constraints or calculation principles are expressed as abstract business rules, and they have to be asserted and validated in order to promote interoperability.

As a proof of concept, BII defined a set of core business rules for the invoice document. The validation architecture was also defined[5] and sample tools were built in order to provide a framework to validate BII transactions.

Even if these rules and artefacts have been expressed and are available in CEN BII site, they are not normative BII validation tools.

2.2 PEPPOL Conformance

The standardization work in BII is the basis for the tools and artefacts developed within PEPPOL Post Award, were concepts like core, syntax neutrality and extensions have been reused and artefacts have been created to comply with BII PEPPOL and European legislation.

In terms of electronic invoicing an XML instance is conformant to PEPPOL when:

1. It is a valid instance in terms of its syntactical structure (e.g. UBL or CII)
2. It fulfils the business rules defined within **CEN BII** and the business rules added by **PEPPOL.**

In order to allow the technical validation of an instance, PEPPOL uses the Validation Architecture framework defined in BII as explained below and have based its work on the BII proof of concept tools and artefacts. PEPPOL has expanded the rule sets and has created technical artefacts to technically validate that an instance is conformant with BII and PEPPOL rules.

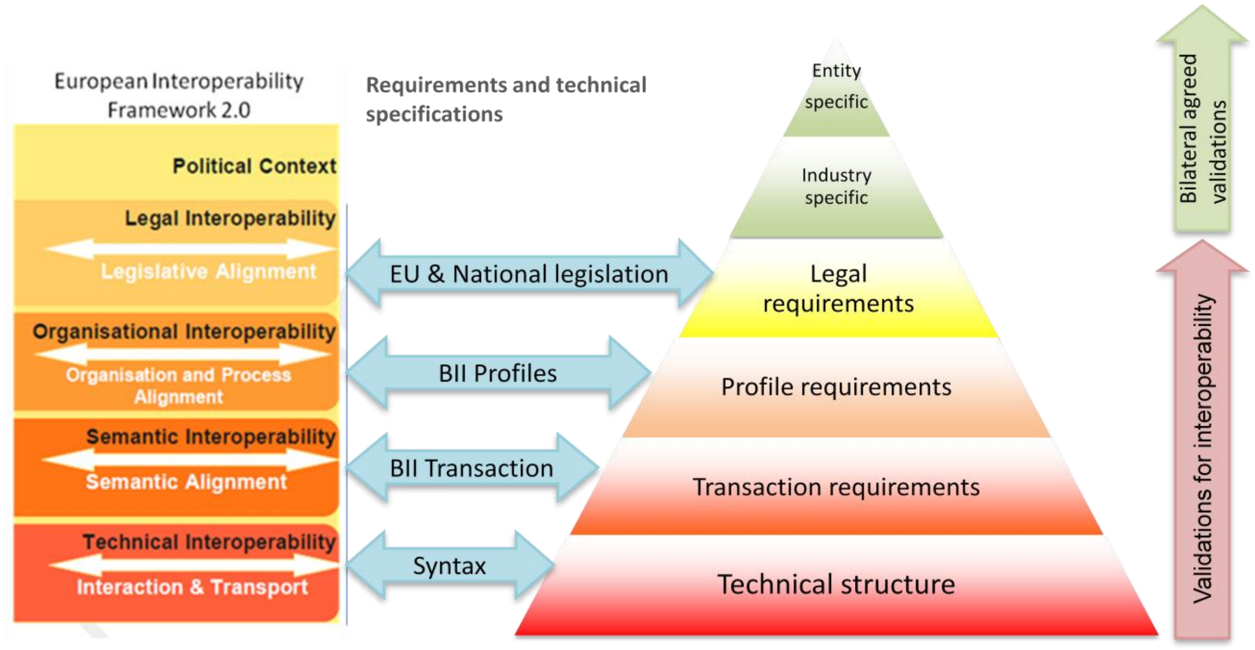
2.3 PEPPOL interoperability

The goal of PEPPOL interoperability is to electronically enable parties to exchange a set of information to be used within a minimal but well defined set of procurement processes without previous bi lateral setup of data, content rules and process specifications. Electronic documents that are conformant to PEPPOL specifications can be exchanged widely within European eInvoicing communities and the issuer can with high degree of certainty expect the document to pass validation by the receiver. The conformance specifications also serve as a basis for bi lateral specification of additional information to be exchanged between trading parties.

[4] http://www.cen.eu/cwa/bii/specs/Profiles/IndexWG1.html
[5] http://www.cen.eu/cwa/bii/specs/Tools/documents/BII3-B-ConformanceTesting_d09.doc

3 Validation architecture

BII has drawn the validation architecture framework as a pyramid with multiple levels of validation. The base of the pyramid has the structural requirements for the document model while the top of the pyramid handles the requirements of specific business entities such as governments, organizations or companies. There is a clear relationship between the pyramid of validation and the different levels of interoperability defined by the European Interoperability Framework 2.0[6]


European Interoperability Framework 2.0
Political Context
Legal Interoperability
Legislative Alignment
Organisational Interoperability
Organisation and Process Alignment
Semantic Interoperability
Semantic Alignment
Technical Interoperability
Interaction & Transport
Requirements and technical specifications
EU & National legislation
BII Profiles
BII Transaction
Syntax
Entity specific
Industry specific
Legal requirements
Profile requirements
Transaction requirements
Technical structure
Bilateral agreed validations
Validations for interoperability

Where the technical validation artefacts provide the technical interoperability level, the semantic interoperability is provided by the CEN BII syntax neutral basic document constraints; the organizational interoperability can be validated using the profile requirements and finally, the legal interoperability is validated using legal requirement validation artefacts.

Different artefacts have been created for the validation of an XML instance in these different layers. The validation of a document instance is applied bottom-up in the pyramid, so the order of validation is:

3.1 BII profile conformance

The first three levels (technical, transaction and profile) are the basis for interoperability and define the compliance to the BII profile specifications. Every document exchanged must be valid according to these three levels of validation, and everyone in the exchange community must be able to process all received elements in a valid instance.

1. Validate that the **technical structure** of the document is correct.
2. Validate the **basic common** rules for the document to ensure the semantics of the instance are valid
3. Validate the **process rules** as the document instance is used within an specific profile

Processing a document instance does not mean being required to process it automatically. The information in a received instance can be partially processed electronically and partially viewed manually e.g. by use of style sheets. The validation levels in the profile conformance are:

[6] http://europa.eu/index_en.htm

3.2 National and EU legal compliance

An invoice or credit note document must be legally compliant according to the legal environment in which the issuing party operates. In PEPPOL these legal requirements originate from two main sources, EU directives and national laws.

The PEPPOL project has collected applicable national laws for invoices and credit notes and aligned them with the EU directives. This has made it possible to state a set of invoice legal requirements that are Pan European and apply to all invoices and credit notes exchanged within the PEPPOL project.

A national legal disposition may however require additional rules to domestic documents that are not required to be fulfilled by foreign companies and are applied within the context of nationality of the parties.

1. Validate Pan European legal requirements as specified in the EU directives.
2. Validate national legal requirements if the xml instance is in the proper context. For instance, Norwegian legal requirements will only apply to Norwegian companies performing domestic invoice exchange.

3.3 Extended business rules

Even if there is a common interoperability framework where there is a common layout and rules for the documents exchanged within a community, there might be particular requirements at bilateral level that have to be defined and fulfilled to allow for electronic communication between specific parties. Such requirements must be defined using bilateral agreements. These rules are expressed on the upper levels of the pyramid.

Specific industries can define rules applicable only to its sector. The same applies to single companies: A single entity can define rules to allow document instances automatically being imported into their IT system.

This completes the validation pyramid with the three upper levels, where the first one comprehends the rule sets to comply with legal rules in different countries while the remaining levels require a bilateral agreement to be used between two trading partners.

1. Validate industry-specific requirements. This kind of requirements applies to industries and should be agreed between parties using bilateral agreements. Specifications for this level are not part of PEPPOL.
2. Validate entity-specific requirements. This kind of requirements applies to each bilateral exchange between companies and should be agreed between parties by using bilateral agreements. Specifications for this level are not part of PEPPOL.

The purpose of PEPPOL is to create at least one validation artefact for each validation layer, so users get technical material to ensure their implementations are conforming to the standards.

4 Validation artefacts

According to the BII specifications, individual electronic transactions can be found in different Profiles.

For validation purposes, there are different types of artefacts:

- XML Schema (XSD). Most XML syntaxes use XSD to define the structure of the document models.
- Schematron files. BII and PEPPOL have defined their business rules using Schematron validation rules.
- XSLT files. Schematron files can easily be transformed into XSLT files, so it is easier to run validations.

All technical artefacts except the ones in the technical layer are expressed in two different sets of artefacts, a set of Schematron files that can be used with a Schematron engine to validate the instance and an XSLT file that can be used with an XSLT engine. Both solutions are equivalent in terms of functionality; the XSLT is created from the Schematron files, so they can be interchanged.

4.1 Validation artefacts for UBL 2.0

4.1.1 Technical level

Document data models in PEPPOL are UBL 2.0 documents. Document models must follow UBL structure, so the first validation artefact to use when validating an XML instance is the matching UBL 2.0 XML Schema that will fail if the instance is not a valid UBL 2.0 instance. For the profiles being implemented, the following UBL XSD schemas shall be used:

- UBL-Invoice-2.0.xsd
- UBL-CreditNote-2.0.xsd
- UBL-Catalogue-2.0.xsd
- UBL-Order-2.0.xsd
- UBL-OrderResponseSimple-2.0.xsd

4.1.2 Transaction level

Transaction level validation focuses on the common rules agreed for the document transaction. The transaction level is tightly coupled with CEN BII transaction work and its rules. Besides technical constraints due to the syntax standard, BII mandates constraints on the transaction data model.

As identified above, there are no normative BII artefacts to validate BII constraints; however, there is a need to technically validate an XML instance to be able to assert that a particular instance is valid according to BII document core data model. That’s why PEPPOL has developed Schematron artefacts that allow for technically validating BII conformance of invoice document instances.

The transaction level validation is carried out in two steps. The first step checks if the instance fulfils the rules of a core document, in which case it checks if the instance contains elements that are not in the core. The second step checks if the content of the document instance follows the rules that are specified in the BII profile specifications.

Checking core data model

Every XML instance conformant with BII must declare its customization identifier. In particular, UBL has a data element in the root of the data model called CustomizationID where the sender must identify the customization used to create the electronic document. Each transaction data model in BII has a unique transaction URN that is placed in the CustomizationID data element.

Based on this CustomizationID, there are different artefacts available to validate conformance to BII core.

- BIICORE-UBL-T01-V01 for the order transaction (BIITrdm001[7])
- BIICORE-UBL-T10-V01 for the invoice transaction (BIITrdm010)
- BIICORE-UBL-T14-V01 for the credit note transaction (BIITrdm014)
- BIICORE-UBL-T15-V01 for the corrective invoice transaction (BIITrdm015)

PEPPOL's naming for the BII Core artefacts follows this syntax:

BIICORE-*<syntax>-<transactionID>-V<version_number>, example: BIICORE-UBL-T10-V01*

Every rule identified in these artefacts has the following type of identifier:

BIICORE-*<transactionID>-R<3_digits_rule_number>, example: BIICORE-T10-R001*

All these validation artefacts test core data models and have similar results:

1. If CustomizationID is not correct, the artefact raises an error. For instance, trying to run BIICORE-T10-UBL-V01 on a credit note, which is transaction data model 014, will raise an error.
2. If there are empty elements, it raises a warning, as this is something usually not allowed in international standards.
3. It raises a warning if there is an element in the XML instance that is not defined in the BII core data model transaction.
4. It raises an error or a warning if the cardinality of elements is not according to the BII core data model transaction.

Checking BII rules

BII has defined business rules that extend the structural set of rules for the core defined above. The main purpose of these rules is to explain how to use the data model to improve interoperability.

- BIIRULES-UBL-T01-V02, BII business rules for order transaction (BIITrdm001)
- BIIRULES-UBL-T02-V01, BII business rules for order acceptance transaction (BIITrdm002)
- BIIRULES-UBL-T03-V01, BII business rules for order rejection transaction (BIITrdm003)
- BIIRULES-UBL-T10-V02, BII business rules for invoice transaction (BIITrdm010)
- BIIRULES-UBL-T14-V01, BII business rules for credit note transaction (BIITrdm014)
- BIIRULES-UBL-T15-V01, BII business rules for corrective invoice transaction (BIITrdm015)
- BIIRULES-UBL-T19-V01, BII business rules for catalogue transaction (BIITrdm019)
-

In this case, the naming for the BII rules artefacts follows this syntax:

BIIRULES-*<syntax>-<transactionID>-V<version_number>, example: BIIRULES-UBL-T10-V01*

And the identifiers for the rules inside the artefacts are:

BIIRULE-*<transactionID>-R<3_digits_rule_number>, example: BIIRULE-T10-R001*

The different types of rules applied by BIIRULES artefacts are:

- Calculation model rules
- Co-occurrence constraints

[7] BII identifier for the transaction data model.

- Usage of structures such as addresses, tax structures, etc.
- Code list constraints
- Presence of mandatory elements
- Facets on the contents, numeric element should be within a certain range (<, <=, =, =>, >; or any Boolean combination of this) or must be within certain length constrains (minimum or maximum length).

4.1.3 Profile level

Profile layer contains the rules that apply to a transaction relative to the process in which it is used as defined in the BII profile.

PEPPOL has created validation artefacts for each transaction type (e.g. transaction 10 which is invoice) and contains rules that apply in the context of the profile in which a transaction instance is being used.

The actual artefacts created in PEPPOL are:

- BIIPROFILES-UBL-T01-V01, BII Profile rules for transaction 01 when used in Profile 3 and 6.
- BIIPROFILES-UBL-T10-V01, BII Profile rules for transaction 10 when used in Profile 4, 5 and 6.
- BIIPROFILES-UBL-T14-V01, BII Profile rules for transaction 14 when used in Profile 5 and 6.
- BIIPROFILES-UBL-T15-V01, BII Profile rules for transaction 15 when used in Profile 5 and 6.

In this case, the naming for BII profile rules artefacts follows this syntax:

```
BIIPROFILES-<syntax>-<transactionID>-V<version_number>, example: BIIPROFILE-UBL-T10-V01
```

And the identifiers for the rules inside the artefacts are:

```
BII<profile_ID>-<transactionID>-R<3_digits_rule_number>, example: BII04-T10-R001
```

These artefacts contain rules such as:

- Presence of certain data elements that are required for the process in the profile being used.
- Identifier of the profile.

4.1.4 Legal level

The legal layer contains the rules that are required by law. PEPPOL has focused on the European legal provisions for invoice and credit notes documents. Every country participating in PEPPOL has also identified their own legal framework and defined the legal rules that affect domestic trade, rules beyond cross border trade.

European legal framework of compliance there are two levels of legal rules to be considered: **Firstly**, there are cross border rules that are specified in the EU directives and affect every country within the European Union.

These rules are similar to the ones identified in BII for constraints and cover legislation concerning cross border trade in Europe. The main difference between PEPPOL and BII approach is that BII specifications ALLOW for compliance to EU directives whereas the PEPPOL application REQUIRES such compliance.

The actual artefacts created by PEPPOL are the following:

- EUGEN-UBL-T01-V02, PEPPOL common legal business rules for transaction 01
- EUGEN-UBL-T10-V01, PEPPOL common legal business rules for transaction 10
- EUGEN-UBL-T14-V01, PEPPOL common legal business rules for transaction 14
- EUGEN-UBL-T15-V01, PEPPOL common legal business rules for transaction 15

- EUGEN-UBL-T19-V01, PEPPOL common legal business rules for transaction 19
-

In this case, the artefacts must have the following syntax:

EUGEN-*<syntax>-<transactionID>-V<version_number>, example: EUGEN-UBL-T10-V01*

And the identifiers for the rules inside the artefacts are:

EUGEN-*<transactionID>-R<3_digits_rule_number>, example: EUGEN-T10-R001*

These rules will be submitted by PEPPOL as candidate to be adopted in CEN BII 2 workshop as feedback for standardisation.

National legal compliance
Secondly, every country has identified their own rule set based on their national legislation. These rules affect the companies of the country when creating invoices or credit notes.

Legal rules for a country have a pre-requisite: a creator of a document must always comply with the legal requirements of his country; the receiver requires that documents received from a sender from the same country comply with the same legal requirements. A receiver is not allowed to require senders in other countries to comply to his national legislation, as defined in EU directives. The national legal rules are therefore applied based on context but not only on the transaction content, i.e. if a company is in a particular country it applies its own country rules for the given transaction type. In PEPPOL the following artefacts have been created so far (and are subject to extension):

• ATNAT-UBL-T10-V01,	Austrian business rules for invoice transaction (BiiCoreTrdm010)
• NONAT-UBL-T10-V01,	Norwegian business rules for invoice transaction (BiiCoreTrdm010)
• NONAT-UBL-T14-V01,	Norwegian business rules for credit note transaction (BiiCoreTrdm014)
• NONAT-UBL-T15-V01,	Norwegian business rules for corrective invoice transaction (BiiCoreTrdm015)
• ITNAT-UBL-T10-V03,	Italian business rules for invoice transaction (BiiCoreTrdm010)
• DKNAT-UBL-T10-V01,	Danish business rules for invoice transaction (BiiCoreTrdm010)

Legal national artefacts must have the following syntax:

<country_code[8]>NAT-*<syntax>-<transactionID>-V<version_number>, example: ATNAT-UBL-T10-V01*

And the identifiers for the rules inside the artefacts are:

*<country_code>*NAT-*<transactionID>-R<3_digits_rule_number>, example: ATNAT-R001*

4.1.5 Bilateral layer rule set

The two upper layers of the validation pyramid are mainly focused on rules that have narrower scope, like bilaterally agreed rules. A business entity, such as governments, organizations or companies, receiving invoices can within an industry or bilaterally agree with some or all of its suppliers that they meet certain additional requirements to support its own processing. Whether an industry or entity creates warning rules or fatal error rules is up to them, nevertheless, it has to be kept in mind that the stricter they are the higher are

[8] The country code should be represented as the upper-case ISO 3166-2 code for the respective country.

the barriers for interoperability they raise. In order to be BII conformant, entities that apply specific additional rules must also technically accept core level documents. An entity may apply more than one set of additional rules and apply them depending on supplier type or context. Government offices have defined a set of constraints to the document models that are not required by law, so they do not fit into xxNAT rule sets, but are required by the Governments to allow for easy integration of received invoices into their IT systems.

With Austrian and Norway Government requirements the following artefacts have been created:

• ATGOV-UBL-T10-V01,	Austrian government business rules for invoice transaction (BiiCoreTrdm010)
• NOGOV-UBL-T10-V01,	Norwegian government business rules for invoice transaction (BiiCoreTrdm010)
• NOGOV-UBL-T14-V01,	Norwegian government business rules for credit note transaction (BiiCoreTrdm014)
• NOGOV-UBL-T15-V01,	Norwegian government business rules for corrective invoice transaction (BiiCoreTrdm015)

Also in this case, syntax for the naming of validation artefacts is suggested:

<Rule Set identifier>-<syntax>-<transactionID>-V<version_number>, example: ATGOV-UBL-T10-V01

And the identifiers for the rules inside the artefacts are:

<Rule Set identifier>-<transactionID>-R<3_digits_rule_number>, example: ATGOV-R001

The Rule Set identifier in the above naming syntax can be of any structure.

4.1.6 Summary of artefacts

Table below summarizes the artefacts developed to test instances.

Step	Artefact	Outcome	Suggested action
Technical Layer	UBL-Order-2.0.xsd UBL-OrderResponseSimple-2.0.xsd UBL-Invoice-2.0.xsd UBL-CreditNote-2.0.xsd UBL-Catalogue-2.0.xsd	Valid/invalid	Accept/reject document
Basic Layer	BIICORE-UBL-T01-V01 BIICORE-UBL-T10-V01 BIICORE-UBL-T14-V01 BIICORE-UBL-T15-V01	Report with errors/warnings	If error, reject the document If warning depends on the receiver
	BIIRULES-UBL-T01-V02 BIIRULES-UBL-T02-V01 BIIRULES-UBL-T03-V01 BIIRULES-UBL-T10-V02 BIIRULES-UBL-T14-V01 BIIRULES-UBL-T15-V01 BIIRULES-UBL-T19-V01	Report with errors/warnings	If error, reject the document If warning depends on the receiver

Profile Layer	BIIPROFILES-UBL-T01-V01 BIIPROFILES-UBL-T10-V01 BIIPROFILES-UBL-T14-V01 BIIPROFILES-UBL-T15-V01	Report with errors/warnings	If error, reject the document If warning depends on the receiver
Legal Layer	EUGEN-UBL-T01-V02 EUGEN-UBL-T10-V01 EUGEN-UBL-T14-V01 EUGEN-UBL-T15-V01 EUGEN-UBL-T19-V01	Report with errors/warnings	If error, reject the document If warning depends on the receiver
Industry and Entity Layer	ATNAT-UBL-T10-V01 NONAT-UBL-T10-V01 NONAT-UBL-T14-V01 NONAT-UBL-T15-V01 ITNAT-UBL-T10-V03 DKNAT-UBL-T10-V01 ATGOV-UBL-T10-V01 NOGOV-UBL-T10-V01 NOGOV-UBL-T14-V01 NOGOV-UBL-T15-V01	Report with errors/warnings	If error, reject the document If warning depends on the receiver

4.2 Validation artefacts for UN/CEFACT CII

Since there are no pilots within PEPPOL implementing BII results using UN/CEFACT CII syntax, there have not been developed any validation tools for that syntax.

5 Validation tools

5.1 Tools

5.1.1 Schematron files

Technical tools produced in PEPPOL to validate XML Invoice and Credit Note instances are based on Schematron[9]. The Schematron validation artefacts are packaged using the following structure:

```
/
/abstract
/SYNTAX1
/SYNTAX2
/codelist
```

5.1.1.1 Main folder:

Root folder for the validation artefacts has the main Schematron file which imports the abstract, code list and syntax binding artefacts. It has to be invoked when trying to validate an XML instance. The name of the main Schematron file placed in the root validation directory follows the pattern:

```
<RULESET>-<SYNTAX>-<TRANSACTION>.SCH
```

Where:

RULESET - is the identifier for the rule set. PEPPOL has defined 4 main rule sets in the interoperability levels of the validation pyramid: BIICORE, BIIRULES, BIIPROFILES or EUGEN. Additionally, some other rule sets have been identified for countries such as Austria and Norway. They are ATNAT, NONAT, ATGOV and NOGOV.

SYNTAX - is the identifier for the XML syntax to which the business rules are bound.

TRANSACTION - is the identifier for the transaction (Txx) as defined in CEN BII.

5.1.1.2 Abstract folder:

Abstract folder has the abstract Schematron artefacts. The abstract folder is not syntax specific, so the filenames for the Schematron files placed in that directory follow the pattern:

```
<RULESET>-<TRANSACTION>.sch
```

Schematron fragments in the abstract folder group all the assertions per each context in a single pattern.

5.1.1.3 Codelist folder:

If the rule set has code list rules, there is a code list folder that contains all code lists and associations from code lists to actual values in the XML instance in a single Schematron fragment.

The code list Schematron fragment is generated from Genericode[10] files and the associations defined in Context Value Association (CVA) files.

5.1.1.4 Syntax specific folder:

For every syntax, there is a folder with the actual mapping from the abstract rules to real XPATH expressions for that specific syntax.

PEPPOL has defined the binding to UBL so there is only a folder for UBL in the artefacts provided by PEPPOL.

[9] www.schematron.com
[10] OASIS Code List TC

5.1.2 XSLT files

There are several ways of implementing Schematron validators[11]. PEPPOL has used the validator implementation defined by Schematron to create XSLT files that parse and validate XML instances, producing an SVRL output.

SVRL stands for Schematron Validation Report Language and is a reporting language defined within ISO Schematron that produces an XML instance that identifies errors or warnings.

This Schematron implementation should be viewed as a sample implementation, since it can be implemented in many other ways.

XSLT validation files have the same pattern names as the original Schematron files but with the file extension ".xslt".

5.2 How to use validation artefacts

Validation artefacts can be used in different testing environments. They can be used from within XML processing tools or in different environments supporting XSLT or Schematron engines such as web browsers.

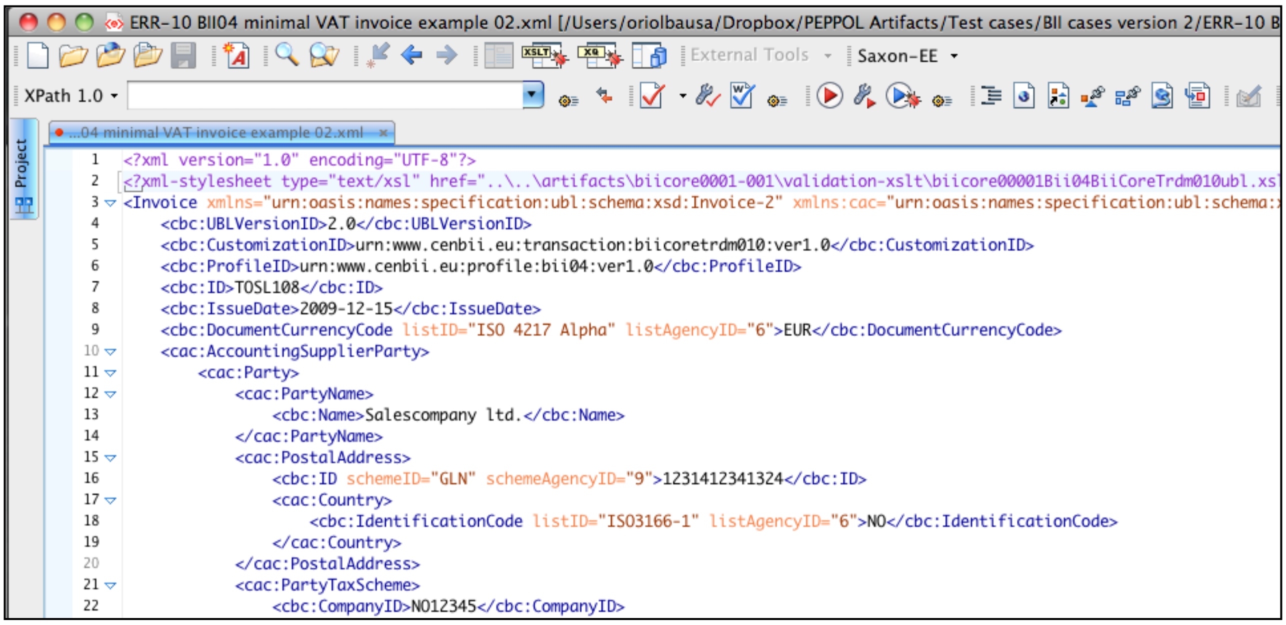
5.2.1 Using XML processing tools

XML processing tools such as XML Altova® or Oxygen® among others can be used to test XML instances. This kind of tools can process both Schematron and XSLT files.

5.2.1.1 Test an instance using Schematron in Oxygen

Oxygen[12] is a commercial XML IDE that has integrated Schematron support.

1. Open the XML instance

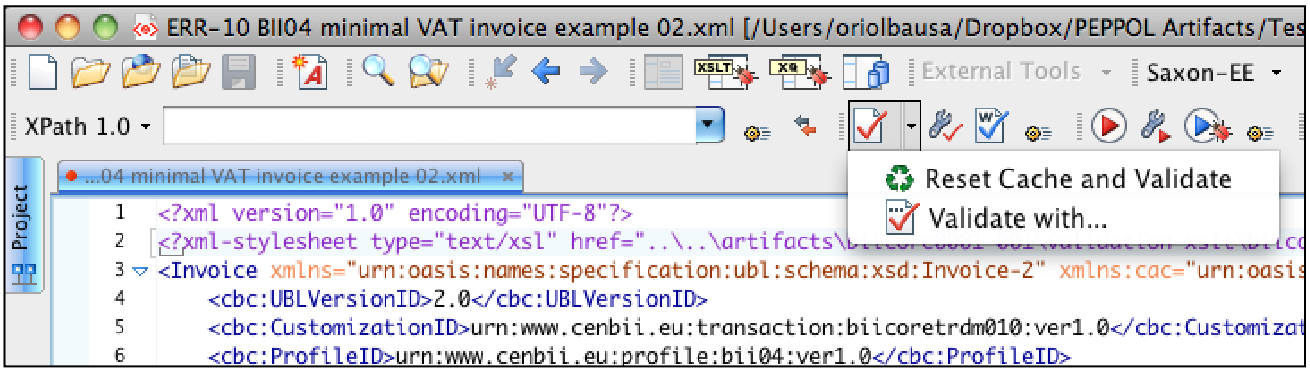


ERR-10 BII04 minimal VAT invoice example 02.xml [/Users/oriolbausa/Dropbox/PEPPOL Artifacts/Test cases/BII cases version 2/ERR-10 B

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="..\..\artifacts\biicore0001-001\validation-xslt\biicore00001Bii04BiiCoreTrdm010ubl.xs
<Invoice xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2" xmlns:cac="urn:oasis:names:specification:ubl:schema:
    <cbc:UBLVersionID>2.0</cbc:UBLVersionID>
    <cbc:CustomizationID>urn:www.cenbii.eu:transaction:biicoretrdm010:ver1.0</cbc:CustomizationID>
    <cbc:ProfileID>urn:www.cenbii.eu:profile:bii04:ver1.0</cbc:ProfileID>
    <cbc:ID>TOSL108</cbc:ID>
    <cbc:IssueDate>2009-12-15</cbc:IssueDate>
    <cbc:DocumentCurrencyCode listID="ISO 4217 Alpha" listAgencyID="6">EUR</cbc:DocumentCurrencyCode>
    <cac:AccountingSupplierParty>
        <cac:Party>
            <cac:PartyName>
                <cbc:Name>Salescompany ltd.</cbc:Name>
            </cac:PartyName>
            <cac:PostalAddress>
                <cbc:ID schemeID="GLN" schemeAgencyID="9">1231412341324</cbc:ID>
                <cac:Country>
                    <cbc:IdentificationCode listID="ISO3166-1" listAgencyID="6">NO</cbc:IdentificationCode>
                </cac:Country>
            </cac:PostalAddress>
            <cac:PartyTaxScheme>
                <cbc:CompanyID>NO12345</cbc:CompanyID>
```

[11] http://www.schematron.com/implementation.html
[12] http://www.oxygenxml.com/

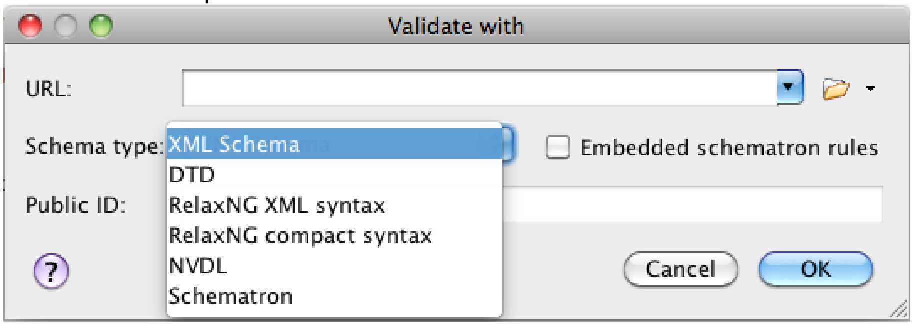
2. Select Validate with…



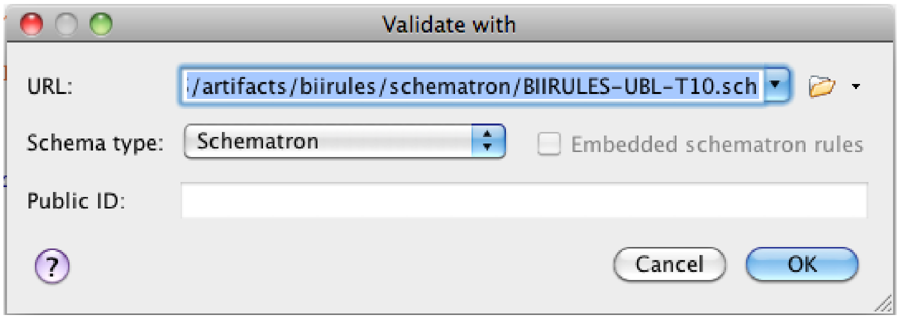
ERR-10 BII04 minimal VAT invoice example 02.xml [/Users/oriolbausa/Dropbox/PEPPOL Artifacts/Tes

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="..\..\artifacts\
<Invoice xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2" xmlns:cac="urn:oasis
    <cbc:UBLVersionID>2.0</cbc:UBLVersionID>
    <cbc:CustomizationID>urn:www.cenbii.eu:transaction:biicoretrdm010:ver1.0</cbc:Customizat
    <cbc:ProfileID>urn:www.cenbii.eu:profile:bii04:ver1.0</cbc:ProfileID>
```

3. Select Schematron in the dropdown box



4. Select the Schematron file you want to use

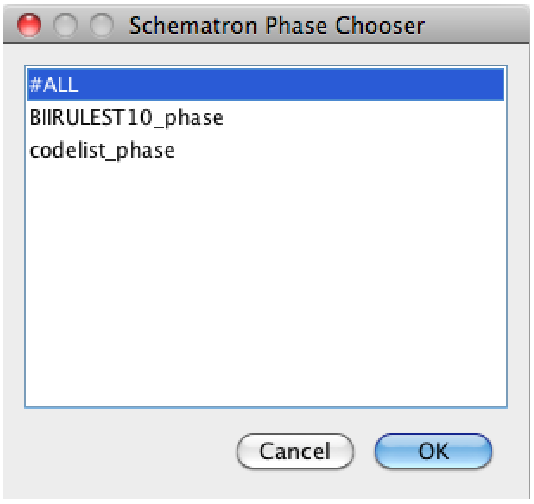


Validate with

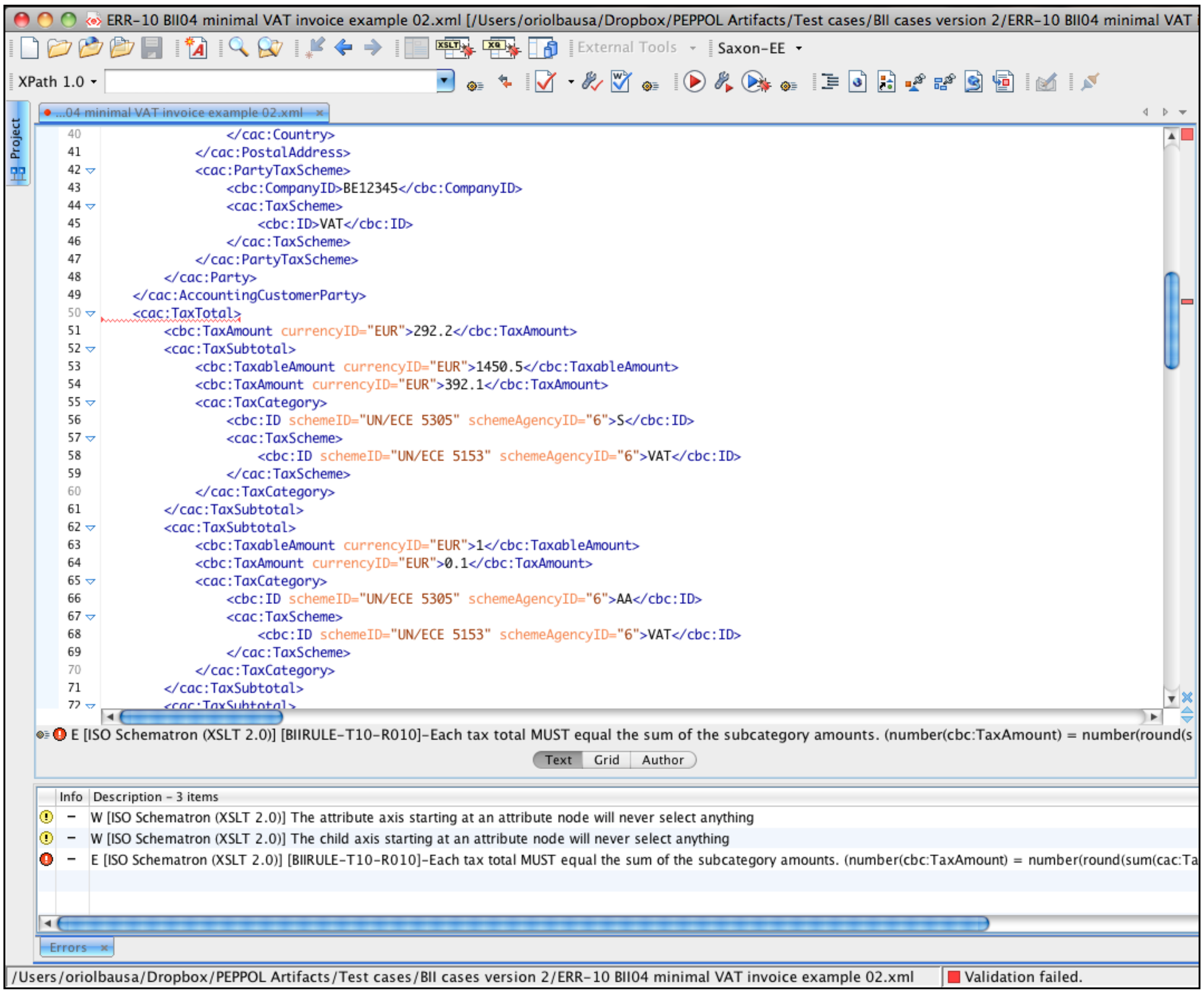
URL: /artifacts/biirules/schematron/BIIRULES-UBL-T10.sch

Schema type: Schematron

5. Select the Phase you want to run (usually there is only one phase)



6. Get the results



ERR-10 BII04 minimal VAT invoice example 02.xml [/Users/oriolbausa/Dropbox/PEPPOL Artifacts/Test cases/BII cases version 2/ERR-10 BII04 minimal VAT

```
                </cac:Country>
            </cac:PostalAddress>
            <cac:PartyTaxScheme>
                <cbc:CompanyID>BE12345</cbc:CompanyID>
                <cac:TaxScheme>
                    <cbc:ID>VAT</cbc:ID>
                </cac:TaxScheme>
            </cac:PartyTaxScheme>
        </cac:Party>
    </cac:AccountingCustomerParty>
    <cac:TaxTotal>
        <cbc:TaxAmount currencyID="EUR">292.2</cbc:TaxAmount>
        <cac:TaxSubtotal>
            <cbc:TaxableAmount currencyID="EUR">1450.5</cbc:TaxableAmount>
            <cbc:TaxAmount currencyID="EUR">392.1</cbc:TaxAmount>
            <cac:TaxCategory>
                <cbc:ID schemeID="UN/ECE 5305" schemeAgencyID="6">S</cbc:ID>
                <cac:TaxScheme>
                    <cbc:ID schemeID="UN/ECE 5153" schemeAgencyID="6">VAT</cbc:ID>
                </cac:TaxScheme>
            </cac:TaxCategory>
        </cac:TaxSubtotal>
        <cac:TaxSubtotal>
            <cbc:TaxableAmount currencyID="EUR">1</cbc:TaxableAmount>
            <cbc:TaxAmount currencyID="EUR">0.1</cbc:TaxAmount>
            <cac:TaxCategory>
                <cbc:ID schemeID="UN/ECE 5305" schemeAgencyID="6">AA</cbc:ID>
                <cac:TaxScheme>
                    <cbc:ID schemeID="UN/ECE 5153" schemeAgencyID="6">VAT</cbc:ID>
                </cac:TaxScheme>
            </cac:TaxCategory>
        </cac:TaxSubtotal>
        <cac:TaxSubtotal>
```

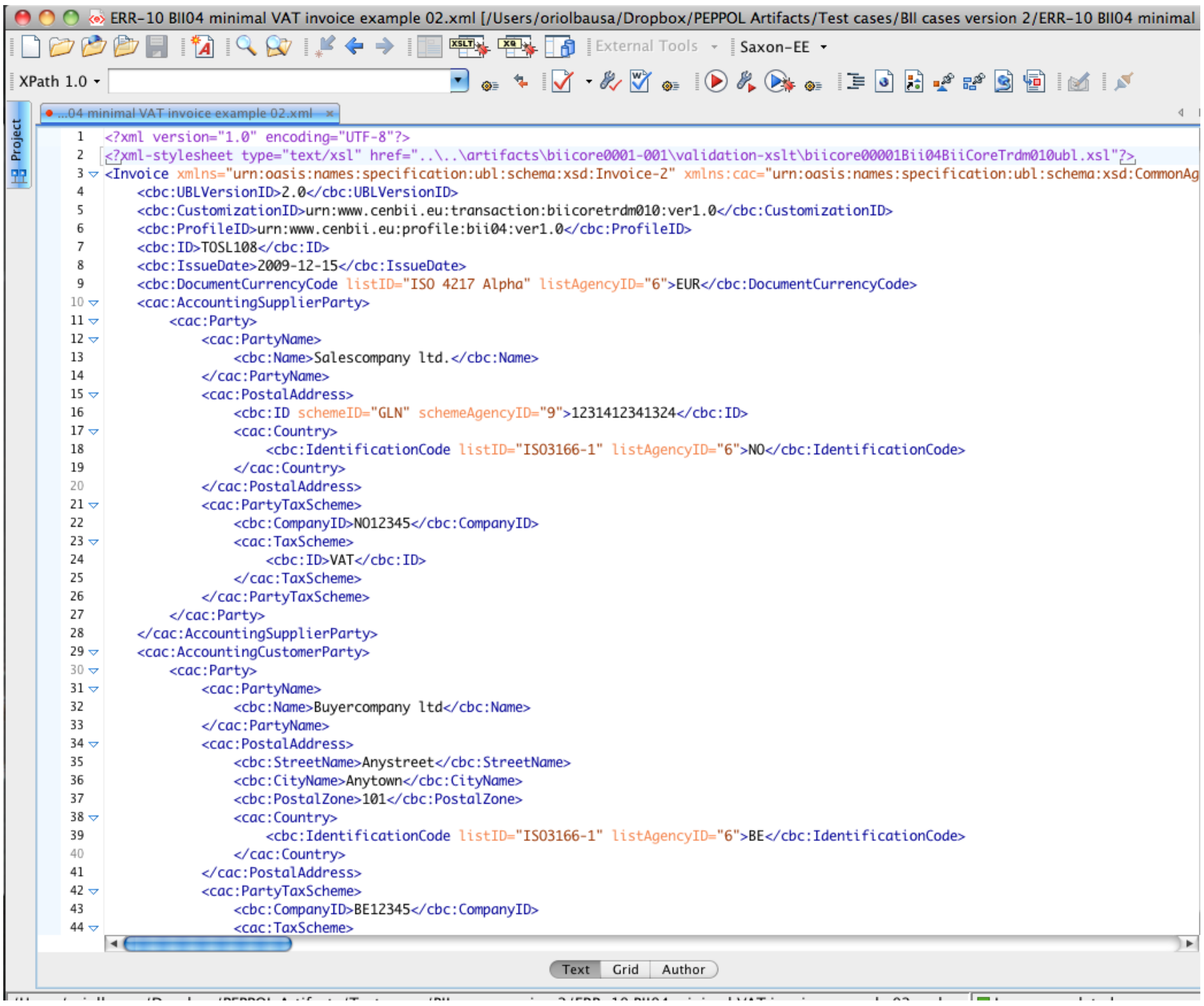
```
E [ISO Schematron (XSLT 2.0)] [BIIRULE-T10-R010]-Each tax total MUST equal the sum of the subcategory amounts. (number(cbc:TaxAmount) = number(round(s
```

```
Info  Description - 3 items
 -    W [ISO Schematron (XSLT 2.0)] The attribute axis starting at an attribute node will never select anything
 -    W [ISO Schematron (XSLT 2.0)] The child axis starting at an attribute node will never select anything
 -    E [ISO Schematron (XSLT 2.0)] [BIIRULE-T10-R010]-Each tax total MUST equal the sum of the subcategory amounts. (number(cbc:TaxAmount) = number(round(sum(cac:Ta
```

/Users/oriolbausa/Dropbox/PEPPOL Artifacts/Test cases/BII cases version 2/ERR-10 BII04 minimal VAT invoice example 02.xml   Validation failed.

5.2.1.2 Test an instance using XSLT in Oxygen

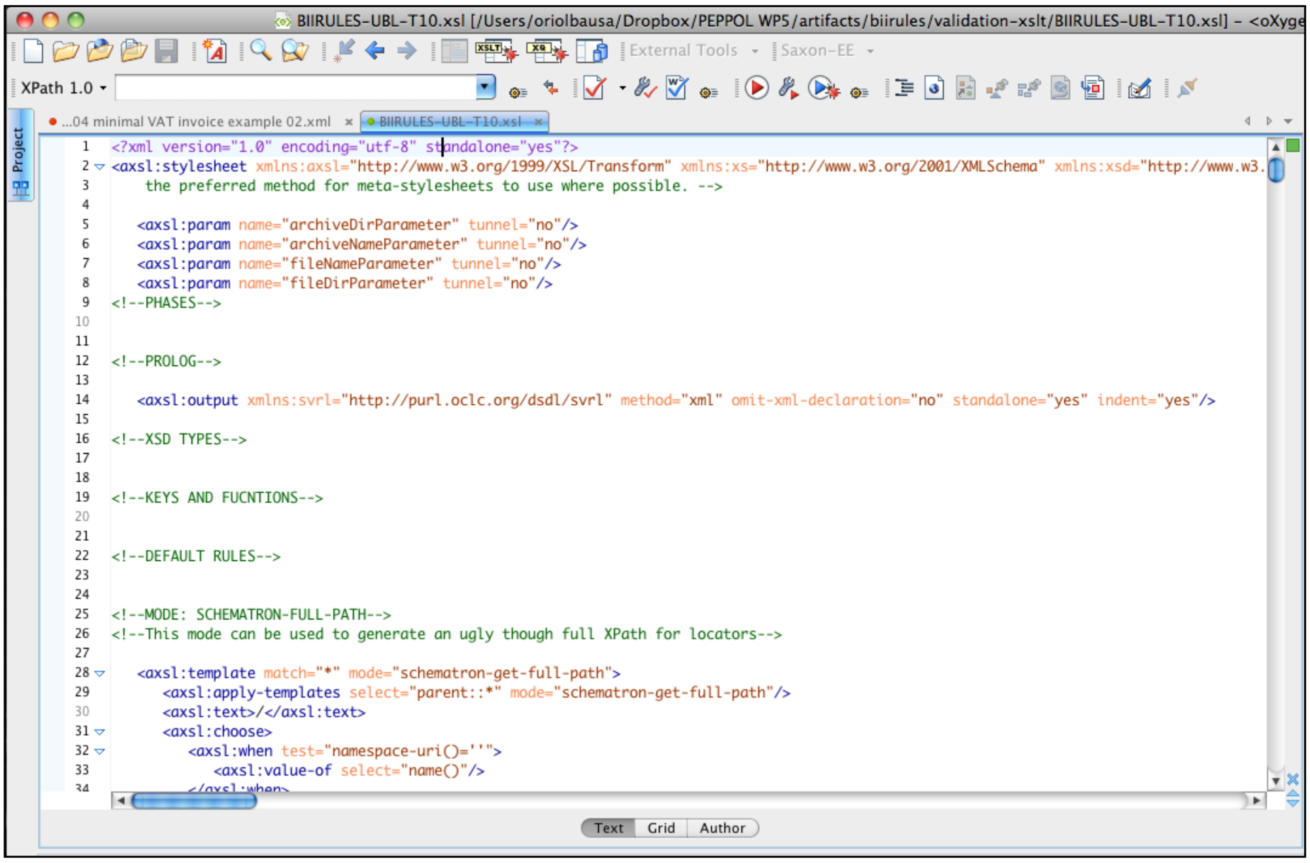
1. Open the XML instance



ERR-10 BII04 minimal VAT invoice example 02.xml [/Users/oriolbausa/Dropbox/PEPPOL Artifacts/Test cases/BII cases version 2/ERR-10 BII04 minimal

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="..\..\artifacts\biicore0001-001\validation-xslt\biicore00001Bii04BiiCoreTrdm010ubl.xsl"?>
<Invoice xmlns="urn:oasis:names:specification:ubl:schema:xsd:Invoice-2" xmlns:cac="urn:oasis:names:specification:ubl:schema:xsd:CommonAg
    <cbc:UBLVersionID>2.0</cbc:UBLVersionID>
    <cbc:CustomizationID>urn:www.cenbii.eu:transaction:biicoretrdm010:ver1.0</cbc:CustomizationID>
    <cbc:ProfileID>urn:www.cenbii.eu:profile:bii04:ver1.0</cbc:ProfileID>
    <cbc:ID>TOSL108</cbc:ID>
    <cbc:IssueDate>2009-12-15</cbc:IssueDate>
    <cbc:DocumentCurrencyCode listID="ISO 4217 Alpha" listAgencyID="6">EUR</cbc:DocumentCurrencyCode>
    <cac:AccountingSupplierParty>
        <cac:Party>
            <cac:PartyName>
                <cbc:Name>Salescompany ltd.</cbc:Name>
            </cac:PartyName>
            <cac:PostalAddress>
                <cbc:ID schemeID="GLN" schemeAgencyID="9">1231412341324</cbc:ID>
                <cac:Country>
                    <cbc:IdentificationCode listID="ISO3166-1" listAgencyID="6">NO</cbc:IdentificationCode>
                </cac:Country>
            </cac:PostalAddress>
            <cac:PartyTaxScheme>
                <cbc:CompanyID>NO12345</cbc:CompanyID>
                <cac:TaxScheme>
                    <cbc:ID>VAT</cbc:ID>
                </cac:TaxScheme>
            </cac:PartyTaxScheme>
        </cac:Party>
    </cac:AccountingSupplierParty>
    <cac:AccountingCustomerParty>
        <cac:Party>
            <cac:PartyName>
                <cbc:Name>Buyercompany ltd</cbc:Name>
            </cac:PartyName>
            <cac:PostalAddress>
                <cbc:StreetName>Anystreet</cbc:StreetName>
                <cbc:CityName>Anytown</cbc:CityName>
                <cbc:PostalZone>101</cbc:PostalZone>
                <cac:Country>
                    <cbc:IdentificationCode listID="ISO3166-1" listAgencyID="6">BE</cbc:IdentificationCode>
                </cac:Country>
            </cac:PostalAddress>
            <cac:PartyTaxScheme>
                <cbc:CompanyID>BE12345</cbc:CompanyID>
                <cac:TaxScheme>
```

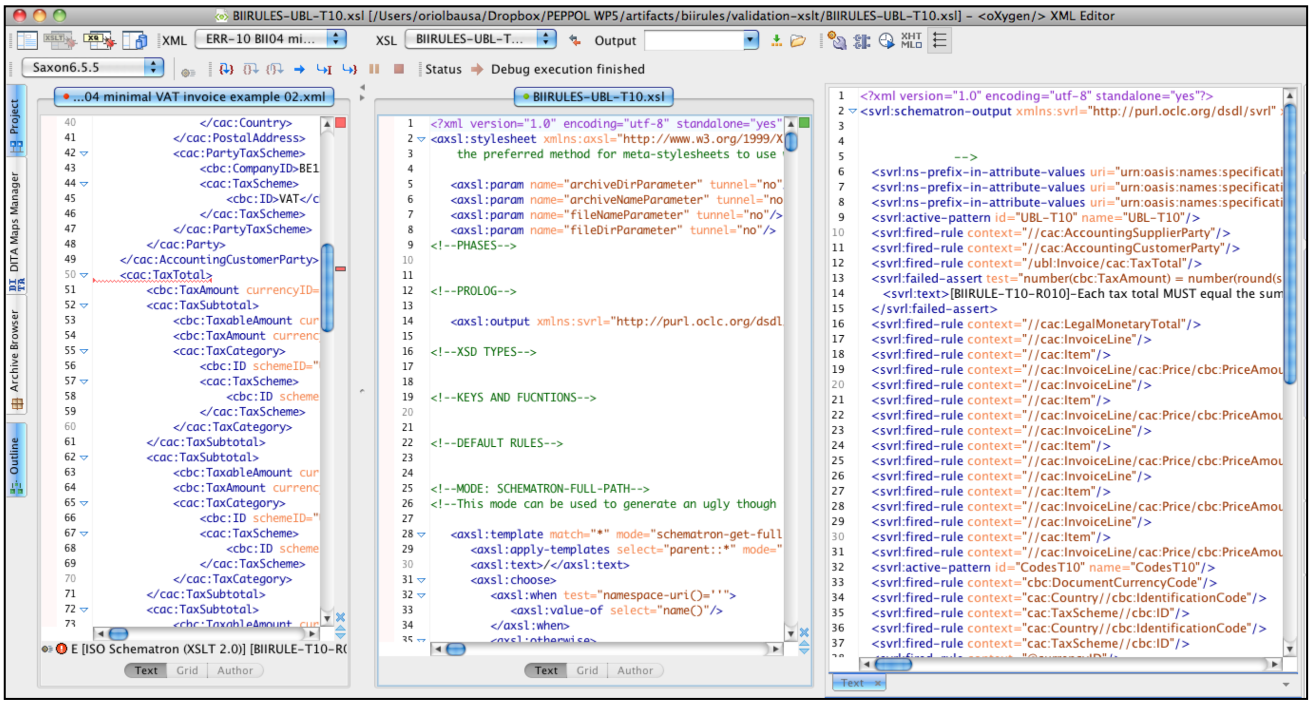
2. Open the XSLT validation file


BIIRULES-UBL-T10.xsl [/Users/oriolbausa/Dropbox/PEPPOL WP5/artifacts/biirules/validation-xslt/BIIRULES-UBL-T10.xsl] - <oXyge

...04 minimal VAT invoice example 02.xml | BIIRULES-UBL-T10.xsl

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<axsl:stylesheet xmlns:axsl="http://www.w3.org/1999/XSL/Transform" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsd="http://www.w3.
   the preferred method for meta-stylesheets to use where possible. -->

   <axsl:param name="archiveDirParameter" tunnel="no"/>
   <axsl:param name="archiveNameParameter" tunnel="no"/>
   <axsl:param name="fileNameParameter" tunnel="no"/>
   <axsl:param name="fileDirParameter" tunnel="no"/>
<!--PHASES-->


<!--PROLOG-->

   <axsl:output xmlns:svrl="http://purl.oclc.org/dsdl/svrl" method="xml" omit-xml-declaration="no" standalone="yes" indent="yes"/>

<!--XSD TYPES-->


<!--KEYS AND FUCNTIONS-->


<!--DEFAULT RULES-->


<!--MODE: SCHEMATRON-FULL-PATH-->
<!--This mode can be used to generate an ugly though full XPath for locators-->

   <axsl:template match="*" mode="schematron-get-full-path">
      <axsl:apply-templates select="parent::*" mode="schematron-get-full-path"/>
      <axsl:text>/</axsl:text>
      <axsl:choose>
         <axsl:when test="namespace-uri()=''">
            <axsl:value-of select="name()"/>
         </axsl:when>
```

3. Go to XSLT mode pressing the XSLT button
4. Run the XSLT and get the SVRL instance


BIIRULES-UBL-T10.xsl [/Users/oriolbausa/Dropbox/PEPPOL WP5/artifacts/biirules/validation-xslt/BIIRULES-UBL-T10.xsl] - <oXygen/> XML Editor

XML: ERR-10 BII04 mi... XSL: BIIRULES-UBL-T... Output:

Saxon6.5.5 Status ➔ Debug execution finished

...04 minimal VAT invoice example 02.xml

```
               </cac:Country>
            </cac:PostalAddress>
            <cac:PartyTaxScheme>
               <cbc:CompanyID>BE1
               <cac:TaxScheme>
                  <cbc:ID>VAT</c
               </cac:TaxScheme>
            </cac:PartyTaxScheme>
         </cac:Party>
      </cac:AccountingCustomerParty>
      <cac:TaxTotal>
         <cbc:TaxAmount currencyID=
         <cac:TaxSubtotal>
            <cbc:TaxableAmount cur
            <cbc:TaxAmount currenc
            <cac:TaxCategory>
               <cbc:ID schemeID="
               <cac:TaxScheme>
                  <cbc:ID scheme
               </cac:TaxScheme>
            </cac:TaxCategory>
         </cac:TaxSubtotal>
         <cac:TaxSubtotal>
            <cbc:TaxableAmount cur
            <cbc:TaxAmount currenc
            <cac:TaxCategory>
               <cbc:ID schemeID="
               <cac:TaxScheme>
                  <cbc:ID scheme
               </cac:TaxScheme>
            </cac:TaxCategory>
         </cac:TaxSubtotal>
         <cac:TaxSubtotal>
```

E [ISO Schematron (XSLT 2.0)] [BIIRULE-T10-R(

BIIRULES-UBL-T10.xsl

```
<?xml version="1.0" encoding="utf-8" standalone="yes"
<axsl:stylesheet xmlns:axsl="http://www.w3.org/1999/X
   the preferred method for meta-stylesheets to use

   <axsl:param name="archiveDirParameter" tunnel="no"
   <axsl:param name="archiveNameParameter" tunnel="no
   <axsl:param name="fileNameParameter" tunnel="no"/>
   <axsl:param name="fileDirParameter" tunnel="no"/>
<!--PHASES-->


<!--PROLOG-->

   <axsl:output xmlns:svrl="http://purl.oclc.org/dsdl

<!--XSD TYPES-->


<!--KEYS AND FUCNTIONS-->


<!--DEFAULT RULES-->


<!--MODE: SCHEMATRON-FULL-PATH-->
<!--This mode can be used to generate an ugly though

   <axsl:template match="*" mode="schematron-get-full
      <axsl:apply-templates select="parent::*" mode="
      <axsl:text>/</axsl:text>
      <axsl:choose>
         <axsl:when test="namespace-uri()=''">
            <axsl:value-of select="name()"/>
         </axsl:when>
```

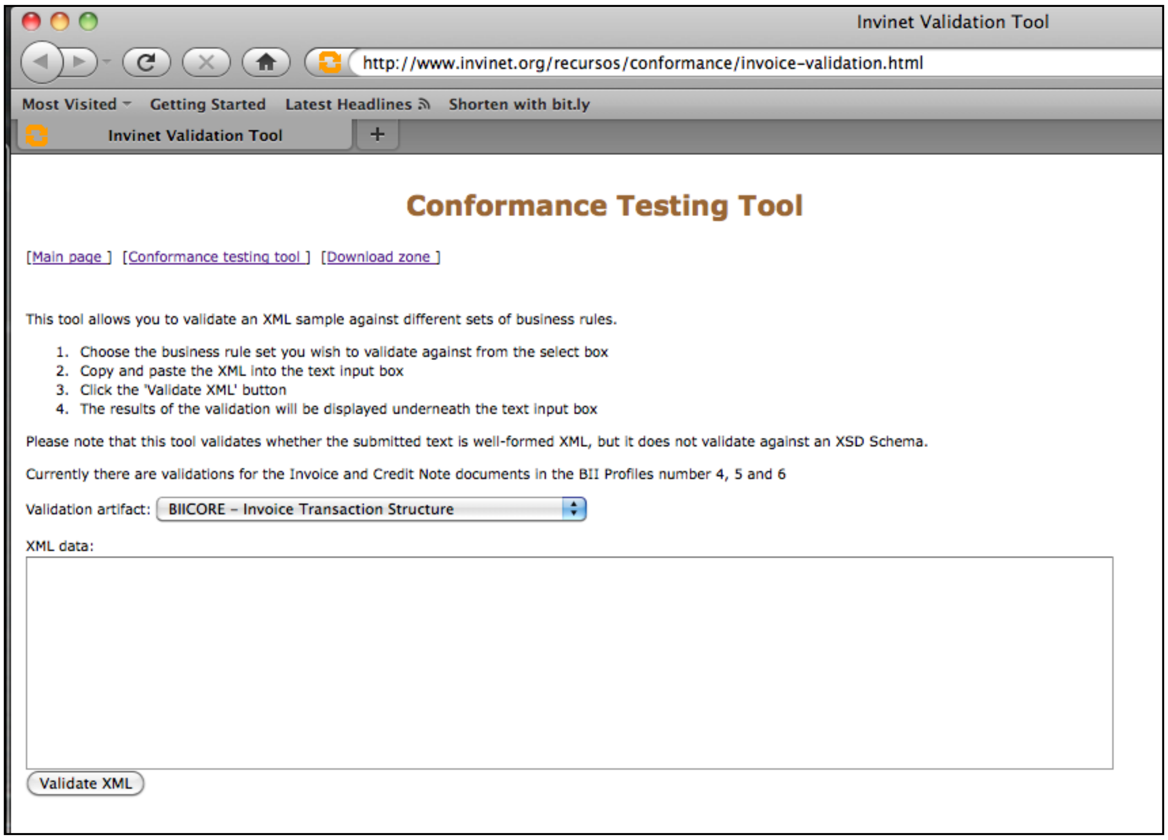
```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<svrl:schematron-output xmlns:svrl="http://purl.oclc.org/dsdl/svrl"


                  -->
  <svrl:ns-prefix-in-attribute-values uri="urn:oasis:names:specificati
  <svrl:ns-prefix-in-attribute-values uri="urn:oasis:names:specificati
  <svrl:ns-prefix-in-attribute-values uri="urn:oasis:names:specificati
  <svrl:active-pattern id="UBL-T10" name="UBL-T10"/>
  <svrl:fired-rule context="//cac:AccountingSupplierParty"/>
  <svrl:fired-rule context="//cac:AccountingCustomerParty"/>
  <svrl:fired-rule context="/ubl:Invoice/cac:TaxTotal"/>
  <svrl:failed-assert test="number(cbc:TaxAmount) = number(round(s
    <svrl:text>[BIIRULE-T10-R010]-Each tax total MUST equal the sum
  </svrl:failed-assert>
  <svrl:fired-rule context="//cac:LegalMonetaryTotal"/>
  <svrl:fired-rule context="//cac:InvoiceLine"/>
  <svrl:fired-rule context="//cac:Item"/>
  <svrl:fired-rule context="//cac:InvoiceLine/cac:Price/cbc:PriceAmou
  <svrl:fired-rule context="//cac:InvoiceLine"/>
  <svrl:fired-rule context="//cac:Item"/>
  <svrl:fired-rule context="//cac:InvoiceLine/cac:Price/cbc:PriceAmou
  <svrl:fired-rule context="//cac:InvoiceLine"/>
  <svrl:fired-rule context="//cac:Item"/>
  <svrl:fired-rule context="//cac:InvoiceLine/cac:Price/cbc:PriceAmou
  <svrl:fired-rule context="//cac:InvoiceLine"/>
  <svrl:fired-rule context="//cac:Item"/>
  <svrl:fired-rule context="//cac:InvoiceLine/cac:Price/cbc:PriceAmou
  <svrl:fired-rule context="//cac:InvoiceLine"/>
  <svrl:fired-rule context="//cac:Item"/>
  <svrl:fired-rule context="//cac:InvoiceLine/cac:Price/cbc:PriceAmou
  <svrl:active-pattern id="CodesT10" name="CodesT10"/>
  <svrl:fired-rule context="cbc:DocumentCurrencyCode"/>
  <svrl:fired-rule context="cac:Country//cbc:IdentificationCode"/>
  <svrl:fired-rule context="cac:TaxScheme//cbc:ID"/>
  <svrl:fired-rule context="cac:Country//cbc:IdentificationCode"/>
  <svrl:fired-rule context="cac:TaxScheme//cbc:ID"/>
```

5.2.2 Using a web browser

Web Browsers include an XSLT engine. A web page can be set up to validate document instances using the built-in XSLT engine.

5.2.2.1 Test an instance using a web validation tool

1. Open web validation tool page



Conformance Testing Tool

[Main page] [Conformance testing tool] [Download zone]

This tool allows you to validate an XML sample against different sets of business rules.

1. Choose the business rule set you wish to validate against from the select box
2. Copy and paste the XML into the text input box
3. Click the 'Validate XML' button
4. The results of the validation will be displayed underneath the text input box

Please note that this tool validates whether the submitted text is well-formed XML, but it does not validate against an XSD Schema.

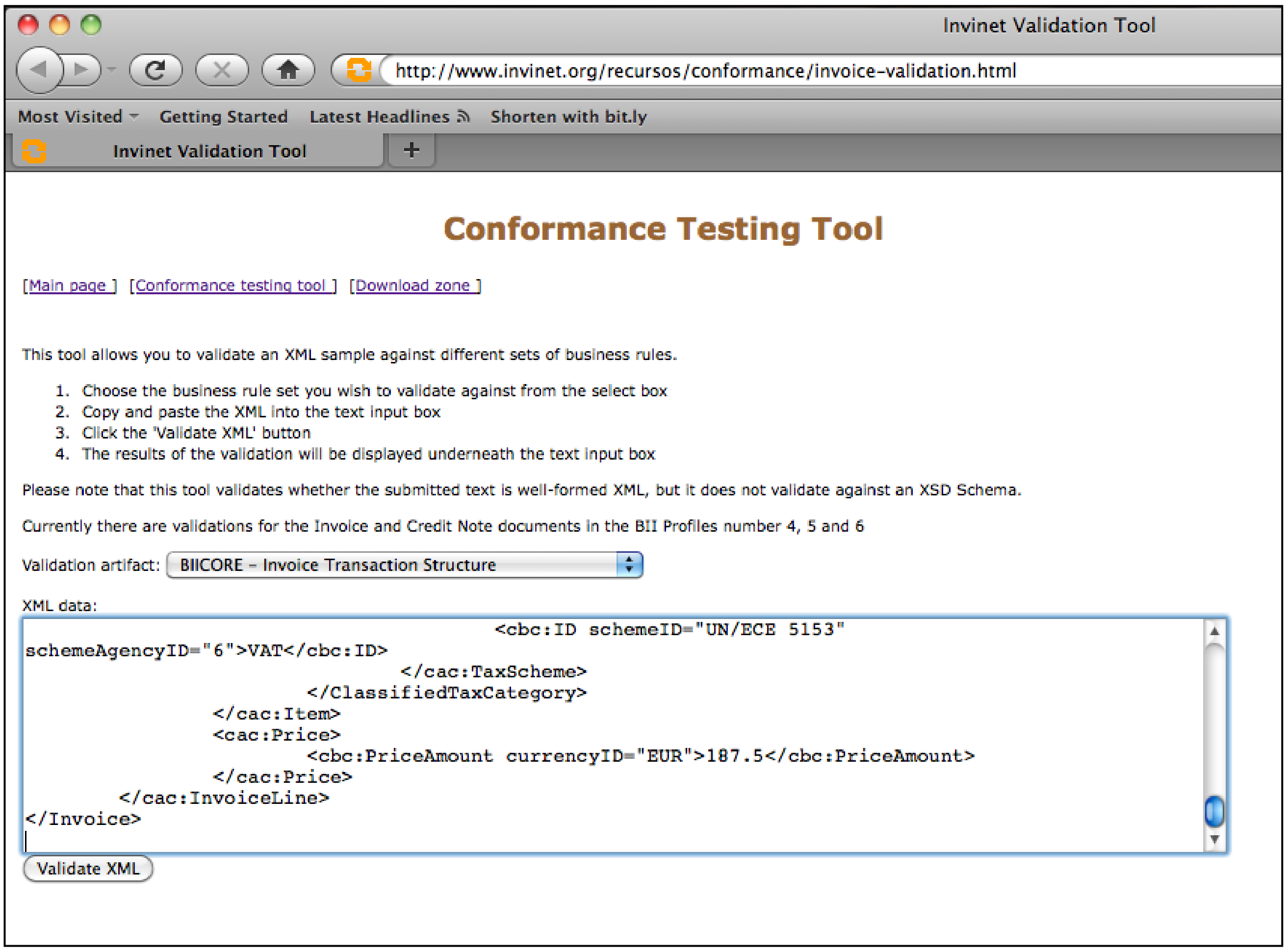
Currently there are validations for the Invoice and Credit Note documents in the BII Profiles number 4, 5 and 6

Validation artifact: BIICORE – Invoice Transaction Structure

XML data:

Validate XML

2. Copy & paste an XML invoice instance in the text area



Conformance Testing Tool

[Main page] [Conformance testing tool] [Download zone]

This tool allows you to validate an XML sample against different sets of business rules.

1. Choose the business rule set you wish to validate against from the select box
2. Copy and paste the XML into the text input box
3. Click the 'Validate XML' button
4. The results of the validation will be displayed underneath the text input box

Please note that this tool validates whether the submitted text is well-formed XML, but it does not validate against an XSD Schema.

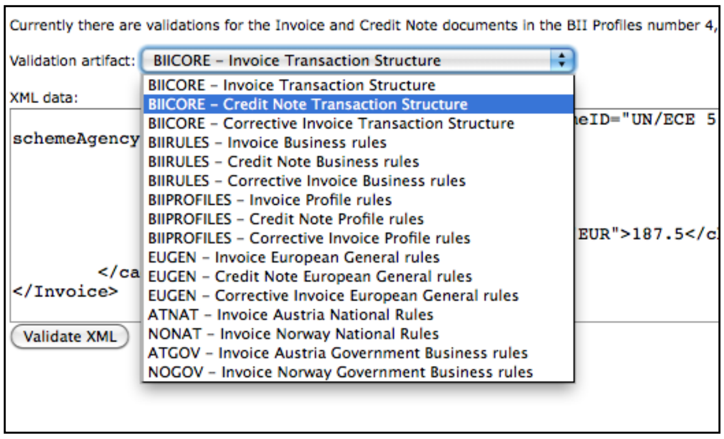
Currently there are validations for the Invoice and Credit Note documents in the BII Profiles number 4, 5 and 6

Validation artifact: BIICORE – Invoice Transaction Structure

XML data:

```
                                        <cbc:ID schemeID="UN/ECE 5153"
schemeAgencyID="6">VAT</cbc:ID>
                                </cac:TaxScheme>
                        </ClassifiedTaxCategory>
                </cac:Item>
                <cac:Price>
                        <cbc:PriceAmount currencyID="EUR">187.5</cbc:PriceAmount>
                </cac:Price>
        </cac:InvoiceLine>
</Invoice>
```

3. Select the validation artefact to run



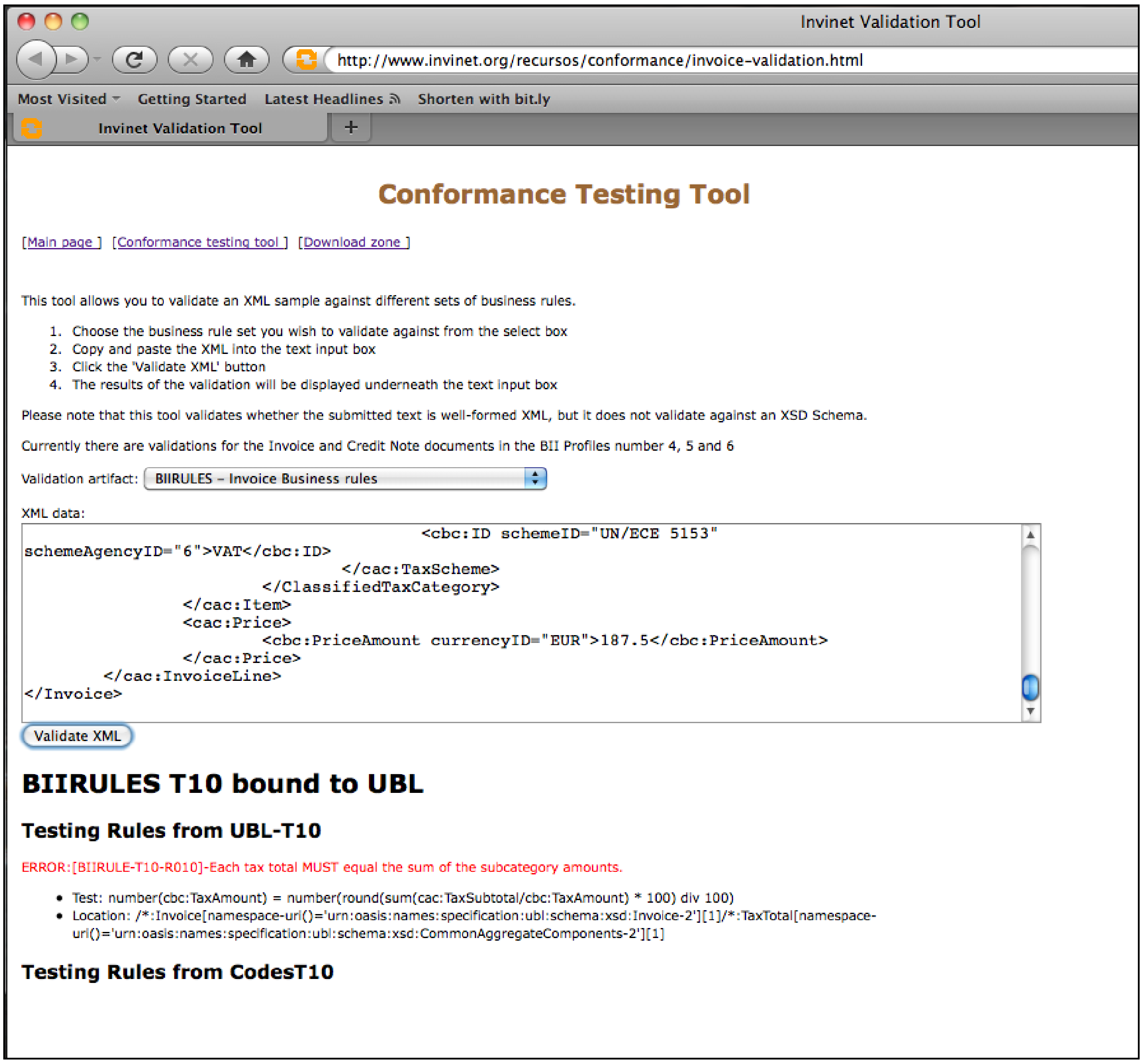
Currently there are validations for the Invoice and Credit Note documents in the BII Profiles number 4,

Validation artifact: BIICORE – Invoice Transaction Structure

XML data:

- BIICORE – Invoice Transaction Structure
- BIICORE – Credit Note Transaction Structure
- BIICORE – Corrective Invoice Transaction Structure
- BIIRULES – Invoice Business rules
- BIIRULES – Credit Note Business rules
- BIIRULES – Corrective Invoice Business rules
- BIIPROFILES – Invoice Profile rules
- BIIPROFILES – Credit Note Profile rules
- BIIPROFILES – Corrective Invoice Profile rules
- EUGEN – Invoice European General rules
- EUGEN – Credit Note European General rules
- EUGEN – Corrective Invoice European General rules
- ATNAT – Invoice Austria National Rules
- NONAT – Invoice Norway National Rules
- ATGOV – Invoice Austria Government Business rules
- NOGOV – Invoice Norway Government Business rules

4. Press validate button



Conformance Testing Tool

[Main page] [Conformance testing tool] [Download zone]

This tool allows you to validate an XML sample against different sets of business rules.

1. Choose the business rule set you wish to validate against from the select box
2. Copy and paste the XML into the text input box
3. Click the 'Validate XML' button
4. The results of the validation will be displayed underneath the text input box

Please note that this tool validates whether the submitted text is well-formed XML, but it does not validate against an XSD Schema.

Currently there are validations for the Invoice and Credit Note documents in the BII Profiles number 4, 5 and 6

Validation artifact: BIIRULES – Invoice Business rules

XML data:

```
                                        <cbc:ID schemeID="UN/ECE 5153"
schemeAgencyID="6">VAT</cbc:ID>
                                <cac:TaxScheme>
                        </ClassifiedTaxCategory>
                </cac:Item>
                <cac:Price>
                        <cbc:PriceAmount currencyID="EUR">187.5</cbc:PriceAmount>
                </cac:Price>
        </cac:InvoiceLine>
</Invoice>
```

Validate XML

BIIRULES T10 bound to UBL

Testing Rules from UBL-T10

ERROR:[BIIRULE-T10-R010]-Each tax total MUST equal the sum of the subcategory amounts.

- Test: number(cbc:TaxAmount) = number(round(sum(cac:TaxSubtotal/cbc:TaxAmount) * 100) div 100)
- Location: /*:Invoice[namespace-uri()='urn:oasis:names:specification:ubl:schema:xsd:Invoice-2'][1]/*:TaxTotal[namespace-uri()='urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2'][1]

Testing Rules from CodesT10