



European eInvoicing Standard in Italy

|  |  |
| --- | --- |
| **Project number** | INEA/CEF/ICT/A2017/1560867 2017-IT-IA-0150 |
| **Project acronym** | EeISI |
| **Project title** | European eInvoicing Standard in Italy |
| **Starting date** | 1 May 2018 |
| **Ending date** | 30 June 2019 |
| **Programme** | Connecting Europe Facility (CEF)  CEF-TC-2017-3: eInvoicing |

**Deliverable D4.1**

**eInvoice Mapper service, software release and documentation**

|  |  |
| --- | --- |
| **Related WP** | WP4 - Implementation |
| **Deliverable number** | 4.1 |
| **Due date** | 31/01/2019 |
| **Revision date** | 31/01/2019 |
| **Actual date** | 31/01/2019 |

Deliverable Info

|  |  |
| --- | --- |
| **Responsible organisation** | Infocert |
| **Editor** | Infocert |
| **Contributors** | Infocert |
| **Reviewers** | Infocert |
|  |  |
| **Abstract** | **This document describes the Mapper service, software release and documentation. It aims to provide information related the mapper implementation approach. The collected information may be used for integration purposes and for understanding the mapper functionalities usage. The document refers to the mapper release in accordance with project timeline. Further modification and improvements of the mapper will not be documented in this project deliverable** |
| **Keywords** | Software, Release |
|  |  |
| **Acknowledgement** | This work was partially supported by the European Commission (EC) through the Connecting Europe Facility (CEF) programme under project EeISI.(grant agreement no. INEA/CEF/ICT/A2017/1560867 2017-IT-IA-0150) |
| **Disclaimer** | The sole responsibility of this publication lies with the author(s). The European Union is not responsible for any use that may be made of the information contained therein. |
| **Confidentiality** | The information in this document is confidential and restricted only to the members of the EeISI consortium (including the Commission Services). |
|  |  |
| **Note** | *-* |
|  |  |

This Page Intentionally Left Blank

**Version Control**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Organisation** | **Description** |
| 0.1 | 21/01/2019 | InfoCert | InfoCert | First draft |
| 1.0 | 31/01/2019 | InfoCert | InfoCert | Final release |
| 1.1 | 31/01/2019 | InfoCert | InfoCert | Minor corrections |
| 1.1.1 | 31/03/2020 | Roberto Reale | AGID | Technical review and quality assessment |

# Table of contents

[Deliverable Info 2](#_Toc4057432)

[Table of contents 2](#_Toc4057433)

[List of tables 3](#_Toc4057434)

[List of figures 3](#_Toc4057435)

[1 Introduction 4](#_Toc4057436)

[1.1 Purpose 4](#_Toc4057437)

[1.2 Scope 5](#_Toc4057438)

[1.3 Overview 5](#_Toc4057439)

[1.4 Definitions, Acronyms and Abbreviations 5](#_Toc4057440)

[1.1. References 6](#_Toc4057441)

[2. Mapper workflow description 7](#_Toc4057442)

[3. Technical Mapper Overview 11](#_Toc4057443)

[3.1. Basic Information 11](#_Toc4057444)

[3.2. Code Repositories 11](#_Toc4057445)

[3.3. Versioning 11](#_Toc4057446)

[3.4. Maven Artifact Repository 11](#_Toc4057447)

[4. Build EeISI 12](#_Toc4057448)

[4.1. Get The Code 12](#_Toc4057449)

[4.2. Compile The Code 12](#_Toc4057450)

[4.3. Exploring the Maven Artifacts 12](#_Toc4057451)

[4.3.1. Entry point artifacts 13](#_Toc4057452)

[4.3.2. Foundation or basic artifacts 14](#_Toc4057453)

[4.3.3. Converter artifacts 14](#_Toc4057454)

[4.3.4. Technical artifacts 15](#_Toc4057455)

[5. Integrate EeISI 15](#_Toc4057456)

[5.1. Prepare The Validation Repository 15](#_Toc4057457)

[5.2. Prepare A Maven Project For An EeISI Client 15](#_Toc4057458)

[5.3. Write a EeISI Client 16](#_Toc4057459)

[5.4. Configure EeISI Logging 17](#_Toc4057460)

[5.5. Configure EeISS properties 17](#_Toc4057461)

[5.6. Launch the Program 22](#_Toc4057462)

[6. Main Libraries of EeISI mapper 22](#_Toc4057463)

[7. EeISI mapper interface usage details 23](#_Toc4057464)

[7.1. API interface usage 23](#_Toc4057465)

[7.2. CLI (command line interface) usage 25](#_Toc4057466)

[7.3. Errors configuration 26](#_Toc4057467)

[7.3.1. Introduction 26](#_Toc4057468)

[7.3.2. Error Logical Structure 26](#_Toc4057469)

[7.3.3. Error Codes 27](#_Toc4057470)

[7.3.4. Error Parameters 27](#_Toc4057471)

[7.3.5. Examples 28](#_Toc4057472)

[CIUS Validation Error 28](#_Toc4057473)

[FatturaPA Element Conversion Error 29](#_Toc4057474)

[7.3.6. Error codes details 29](#_Toc4057475)

# List of tables

[Table 1 Error codes list 31](#_Toc4057476)

# List of figures

[Figure 1 Mapper main steps 7](#_Toc4057477)

[Figure 2 CEN2XMLPA 8](#_Toc4057478)

[Figure 3 XMLPA2CEN 9](#_Toc4057479)

[Figure 4 XMLPA2PeppolBIS3 10](#_Toc4057480)

[Figure 5 UBL/CII2PeppolBIS3 10](#_Toc4057481)

[Figure 6 Maven EeISI mapper structure 13](#_Toc4057482)

# Introduction

## Purpose

The purpose of this document is to explain the basic technical info to work with the EeISI mapper.

This document is jointly written by the Solution Architect and Solution Manager, each working on his view on the same matter which is functionally and technically driven.

In the end, all the parties have to agree on the content.

The intended audiences for this document are:

* Solution Architect
* Technical People Integrating EeISI
* Stakeholders of the project

## Scope

Define, explain and list all the info that allows a 3d party to work with the EeISI software.

## Overview

This document introduces several technical concepts in the next chapter. It is organized as a tutorial that would guide you through the whole process of integrating EeISI in your own solution.

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **B2B** | Business to Business |
| **B2C** | Business to Consumer/Citizen |
| **B2G** | Business to Government |
| **BII** | Business Interoperability Interfaces |
| **C2G** | Citizen to Government |
| **CCTS** | Core Component Technical Specification |
| **CEF** | Connecting Europe Facility |
| **CEM** | Certified Electronic Mail – Legal Mail (PEC Posta Elettronica Certificata in Italy) |
| **CEN** | European Committee for Standardisation |
| **CII** | Cross Industry electronic Invoice |
| **CIUS** | Core Invoice Usage Specification |
| **DSI** | Digital Service Infrastructures |
| **EDIFACT** | Electronic Data Interchange For Administration, Commerce and Transport |
| **EMSFEI** | European Multi-Stakeholder Forum on eInvoicing |
| **e-SENS** | Electronic Simple European Networked Services |
| **FatturaPA** | Public administration electronic invoice framework (FatturaPubblica Amministrazione) |
| **G2G** | Government to Government |
| **INEA** | Innovation and Networks Executive Agency |
| **OASIS** | Organization for the Advancement of Structured Information Standards |
| **PEPPOL** | Pan-European Public Procurement Online |
| **PEPPOL-BIS** | Pan-European Public Procurement Online Business Interoperability Specifications |
| **SDI** | Electronic exchange system in Italy (Sistema Di Interscambio) |
| **UBL** | Universal Business Language |
| **UN/CEFACT** | United Nations Centre for Trade Facilitation and Electronic Business |
| **UNTDID** | UN Trade Data Interchange Directory |
| **URI** | Uniform Resource Identifier |
| **URL** | Uniform Resource Location |
| **URN** | Uniform Resource Name |
| **XML** | Extensible Mark-up Language |

* 1. References

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application.

* EN 16931-1:2017 Electronic invoicing - Part 1: Semantic data model of the core elements of an electronic invoice
* CEN/TS 16931-2:2017 Electronic invoicing - Part 2: List of syntaxes that comply with EN 16931-1
* CEN/TS 16931-3-1:2017 Electronic invoicing - Part 3 - 1: Syntax bindings of the core elements of an electronic invoice - Syntax binding methodology
* CEN/TS 16931-3-2:2017 Electronic invoicing - Part 3 - 2: Syntax bindings of the core elements of an electronic invoice - Binding to ISO/IEC 19845 (UBL 2.1)
* CEN/TS 16931-3-3:2017 Electronic invoicing - Part 3 - 3: Syntax bindings of the core elements of an electronic invoice - Binding to UN/CEFACT XML
* CEN/TS 16931-3-4:2017 Electronic invoicing - Part 3 - 4: Syntax bindings of the core elements of an electronic invoice - Binding to ISO/IEC 9735 (UN/EDIFACT)
* ISO 3166 1, Codes for the representation of names of countries and their subdivisions — Part 1: Country codes
* ISO 4217, Codes for the representation of currencies
* ISO 639 2, Codes for the representation of names of languages
* ISO 8601, Data elements and interchange formats — Information interchange — Representation of dates and times
* ISO 15000-5, Electronic Business Extensible Markup Language (ebXML) — Part 5: Core Components Specification (CCS)
* ISO 6523, Information technology — Structure for the identification of organizations and organization parts
* ISO/IEC 19845, Information technology -- Universal business language version 2.1 (UBL v2.1)

Moreover the following Italian documentation is referenced in this deliverable:

* Schema del file xml FatturaPA versione 1.2 - xsd
* Specifiche tecniche del formato della FatturaPA versione 1.2.1- pdf
* Rappresentazione tabellare del tracciato FatturaPA versione 1.2.1- pdf
* Rappresentazione tabellare del tracciato FatturaPA versione 1.2.1- excel
* Foglio di stile per la visualizzazione della FatturaPA versione 1.2.1 - xslt
* generica Foglio di stile per la visualizzazione della Fattura Ordinaria versione 1.2.1 - xslt
* Elenco modifiche al tracciato FatturaPA - pdf
* Suggerimenti per la compilazione della FatturaPA versione 1.5

1. Mapper workflow description

The EeISI mapper allows to convert and validate electronic invoices according to CEN TS 16931-2 defined syntaxes (namely UBL and CII) to FatturaPA format and viceversia. In addition, the mapper is enriched with new entry points (intermediate CEN meta language) and a mapping to Peppol BIS 3 profile which has been published in 2018 spring.

The mapper performs the main following steps

1. **Validation**: source invoice and target invoice are validated through a set of defined constrains. This step discards the invoice if it does not adhere to the desired constraints (errors are raised)
2. **Mapping**: the information contained in source invoice syntax is converted in target invoice syntax according to the core invoice semantic model defined in EN 16931-1:2017



Figure 1 Mapper main steps

The validation phase includes different validation levels:

1. **Syntactic** validation, using schema XSD
2. **Semantic** validation, using Schematron (sch artifacts)
3. **Customized** Validation (for example a national or community CIUS)

The mapping/conversion phase includes different typologies of mapping for each business element or business group:

1. One to one: there is a perfect match between source element and target element
2. Many to one: source elements are concatenated into target element
3. One to many: source element is split into target elements
4. Customized: the mapping requires a customized process because a specific behaviour needs to be applied

The European norm EN 16931-1:2017 defines a core semantic model to represent eInvoices in a standard way. The mapper is implemented according to the semantic model and does not include, at the moment, extensions to this model. Only restrictions (CIUS core invoice usage specifications) are applied.

The mapper can be defined as “CEN meta model centric” which means that all the invoices expressed in a source syntax are mapped to the CEN semantic model and then converted into invoices expressed in a target syntax. The source invoice can only be converted if it is compliant to the CEN core semantic model. The compliance is verified through the CEN schematron artefacts.

An additional conversion functionality is foreseen using the *-- force* parameter which perform the conversion even though the validation didn’t succeed.

The mapper input and output syntaxes/formats are:

* 1. UBL invoice / UBL credit not
  2. CII
  3. Peppol BIS3 billing
  4. Fattura PA
  5. XML CEN (intermediate semantic meta model)

Please note that the Peppol BIS 3 billing is expressed in UBL syntax (invoice and credit note) and not in CII syntax. In particular it is defined as a CIUS. For this reason, if the invoice source is Peppol BIS 3, as it is CEN compliant, it is considered treated as a generic invoice CEN compliant. If Peppol BIS 3 is the target format the mapper performs a customized conversion to comply with the Peppol CIUS validation artifacts.

The following diagrams represent the different validation and mapping applied by the mapper depending on the source invoice syntax and the target invoice syntax.

Please note that the intermediate invoice-cen.xml is a java object representing a CEN invoice. In eigor-cli this is represented through an xml file.



Figure 2 CEN2XMLPA



Figure 3 XMLPA2CEN



Figure 4 XMLPA2PeppolBIS3



Figure 5 UBL/CII2PeppolBIS3

1. Technical Mapper Overview
   1. Basic Information

The EeISI mapper is implemented in the Java 8 language.

The codebase is structured as a Maven project. Building the project hence requires a proper and working Maven installation at version >= 3.5.0. A working Git installation is also needed not only to checkout the project but also to build it since some info about the revision on which the code is based are included in the built artifacts.

* 1. Code Repositories

The codebase is currently stored in two Git repositories that are aligned at regular intervals.

The authoritative repository lives on Infocert premises to enable a safe development and quality check. Access to it is available only from Infocert’s premises or through a dedicated VPN. On top of that you need to be explicitly authorized to be able to access this repository. The EeISI GIT URL in this repo is the following:

<https://gitlab.mgmt.infocert.it/linvoice/invoice.eeisi.git>

The branches on the repository are mainly managed according the Git Flow best practice, even though some exceptions exist because of technical needs.

The main branches on the repo are:

/master: that is always aligned with the latest tag released.

/develop: that is the head of the development branch, where new features and bugfixes get released before a stable release.

Every release is also tagged. The tag name corresponds to the release version.

The second aligned repository is public according to EeISI project open source approach, available at:

<https://github.com/2017-IT-IA-0150/EeISI.git>

Typically when a release is done on the authoritative Git, /develop and /master branches are pushed to this one too, along with all the new tags.

The public github repository is under EUPL licence.

* 1. Versioning

Versioning is done according to Semantic Versioning ( <https://semver.org/> ). The version available in the /develop branch is always a SNAPSHOT version that usually has a minor upgraded respect the last release. Proper versioning of the released version is typically defined at release time.

* 1. Maven Artifact Repository

The artifacts produced by the project are stored in a private Maven Artifact repository available only on Infocert premises or through a VPN. There’s currently no plan to store these artifacts on a public repository so users should probably compile the project on any machine that has to include EeISI or provide a Maven repository by themselves.

1. Build EeISI

This paragraph contains the information necessary to deploy and integrate EeISI mapper into specific environments. At the moment the project foresees the main integration in production environment at the Italian Revenue Agency where the SDI (Sistema di Interscambio) resides. An additional integration is deployed into Infocamere premises in order to offer a service to SMES to map XMLPA invoices to CEN compliant target invoices

* 1. Get The Code

Obtain a copy of the EeISI source code from one of the aforementioned Git repositories.

$ git clone *repository\_name*

* 1. Compile The Code

Open a shell, go to the source root folder and compile the project.

$ cd <PROJECT-HOME>

$ mvn clean install -Prelease

After a while, Maven will compile and install the EeISI libraries in your local Maven repository.

* 1. Exploring the Maven Artifacts

This is a brief overview of the Maven modules, their goals and a description of the noteworthy generated artifacts.

├───converter-cen-cii

│ └───mappings

├───converter-cen-fattpa

│ └───mappings

├───converter-cen-ubl

│ └───mappings

├───converter-cen-ublcn

│ └───mappings

├───converter-cen-xmlcen

│ └───mappings

├───converter-cii-cen

│ └───mappings

├───converter-commons

│ ├───cii

│ │ ├───cius

│ │ │ ├───schematron

│ │ │ └───schematron-xslt

│ │ ├───schematron

│ │ │ ├───abstract

│ │ │ ├───CII

│ │ │ └───codelist

│ │ ├───schematron-xslt

│ │ └───xsd

│ │ ├───coupled

│ │ │ ├───codelist

│ │ │ │ └───standard

│ │ │ ├───data

│ │ │ │ └───standard

│ │ │ └───identifierlist

│ │ │ └───standard

│ │ └───uncoupled

│ │ └───data

│ │ └───standard

│ ├───fattpa

│ │ ├───xsd

│ │ └───xsdstatic

│ │ └───imported

│ ├───ubl

│ │ ├───cius

│ │ │ ├───schematron

│ │ │ └───schematron-xslt

│ │ ├───schematron

│ │ │ ├───abstract

│ │ │ ├───codelist

│ │ │ └───UBL

│ │ ├───schematron-xslt

│ │ ├───xsd

│ │ └───xsdstatic

│ │ └───imported

│ ├───ublcn

│ │ ├───xsd

│ │ └───xsdstatic

│ │ └───imported

│ └───xmlcen

│ ├───xsd

│ └───xsdstatic

├───converter-fattpa-cen

│ └───mappings

├───converter-ubl-cen

│ └───mappings

├───converter-ublcn-cen

│ └───mappings

└───converter-xmlcen-cen

└───mappings

Figure 6 Maven EeISI mapper structure

* + 1. Entry point artifacts
* eigor-api
  + Sourcecode of the EeISI Java API. This artifact is the entry point for everyone that needs to integrate EeISI in their systems.
* eigor-configurations
  + A resourceless Maven module which goal is to build files in various formats: jar, zip and tar.gz that contains all XSD and schematrons used by the mapper to validate input and output invoices. These files are needed in the integration phase of the EeISI library in 3rd party solutions, mainly because some formats as CII and UBL provides XSD split across different files. Some libraries used in EeISI are not able to load such XSD from jar in classpath, so it is mandatory to set up a local folder where the content of these artifacts is uncompressed. This artifact carries also a tweaked version of those XSD that avoid to look up imported XSD in Internet. This can beneficial in some scenarios where the application integrating EeISI cannot access the public Internet.
* eigor-cli
  + A self contained command line interface that uses EeISI to convert invoices. It is not needed to convert invoices with the eigor-api but it could be priceless to explore EeISI functionalities and to reproduce conversion scenarios.
    1. Foundation or basic artifacts
* cen-core
  + Contains the resources needed to dynamically generate Java classes representing the BT and BG elements of the CEN norm.
    1. Converter artifacts

This module contains all the converter modules based on the CEN centric approach (the CEN core semantic model is used as common structure for all the conversions)

* converter-cen-cii
  + Sourcecode of the converter that transform invoices defined with the CEN model in CII XML files.
* converter-cen-fattpa
  + Sourcecode of the converter that transform invoices defined with the CEN model in FattPA files.
* converter-cen-peppol
  + Sourcecode of the converter that transform invoices defined with the CEN model in Peppol files.
* converter-cen-peppolcn
  + Sourcecode of the converter that transform credit notes defined with the CEN model in Peppol files.
* converter-cen-ubl
  + Sourcecode of the converter that transform invoices defined with the CEN model in UBL XML files.
* converter-cen-ublcn
  + Sourcecode of the converter that transform credit notes defined with the CEN model in UBL files.
* converter-cen-xmlcen
  + Sourcecode of the converter that transform credit notes defined with the CEN model in XMLCEN iles.
* converter-cii-cen
  + Sourcecode of the converter that transform CII files containing invoices in the corresponding CEN model.
* converter-commons
  + Resources or classes that are in common and shared across two or more converters, be they CEN-xxx or xxx-CEN converters.
* converter-csvcen-cen
  + Sourcecode of the converter that transform purposely built CSV files representing a CEN invoice into the corresponding CEN model. This converter is mainly used for debug purposes because the supported CSV format is totally custom and not tied to any defined standard. It proves invaluable in debug though because it allows to feed the mapper with the same CEN invoice that was obtained during a troubleshooting session.
* converter-fattpa-cen
  + Sourcecode of the converter that transform FattPA files containing invoices in the corresponding CEN model.
* converter-ubl-cen
  + Sourcecode of the converter that transform UBL files containing invoices in the corresponding CEN model.
* converter-ublcn-cen
  + Sourcecode of the converter that transform UBL credit notes in the corresponding CEN model.
* converter-xmlcen-cen
  + Sourcecode of the converter that transform either XMLCEN credit notes and invoices in the corresponding CEN model.
    1. Technical artifacts
* eigor-core
  + Contains the basic models and abstractions that define how the various converter can work together, and some services used by all of them.
* eigor-parent
  + Maven parent project defining common properties, managed dependencies, base plugin definitions and distribution repositories used in all other modules.
* eigor-spring-core
  + A porting of Spring Core library in EeISI. This artifact mainly offer the possibility to use several Spring Framework resource related abstractions in the EeISI configuration files without the need to have EeISI depend on the whole Spring Framework. For further info, please check out https://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/resources.html
* eigor-test
  + Contains resources and classes leveraged during automatic tests.
* eigor-test-schemas
  + Contains test schematrons and XSD used to unit test the related validators.

1. Integrate EeISI
   1. Prepare The Validation Repository

EeISI uses several artifacts to validate the invoices: it handles XSD schemas, and schematron files. So, before start using EeISI it is necessary to verify if all used artifacts are up to date.

A regularly check is performed each time a new version of EeISI is released to verify if schema and schematron have been update since the previous mapper version.

Unzip the *eigor-configurations/target/eigor-configurations-x.y.z.zip* or the equivalent *eigor-configurations/target/eigor-configurations-x.y.z.tar.gz* file in the local integration environment.

*Please note that this is valid up to mapper version 4.2.0 but in next releases this feature will be enhanced and this step becomes optional. It is s only needed if the integrator wants to change some defaults, like SCHs and configurable mappings, otherwise sensible defaults are used.*

* 1. Prepare A Maven Project For An EeISI Client

You can now use EeISI the same way you would do with any other Maven project. Create a new Maven project with this pom.xml:

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>test</groupId>

<artifactId>test</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>it.infocert.eigor</groupId>

<artifactId>eigor-api</artifactId>

<version>4.0.1-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

</dependencies>

</project>

eigor-api is the dependency that will give you the entry point to the EeISI API. logback-classic is a SLF4J logging implementation that EeISI can use to log.

* 1. Write a EeISI Client

In the src/main create a Test.java file with a main method. The first step is to create an EigorApiBuilder that provides instances of the EigorApi that can be used to convert between invoices.

import com.infocert.eigor.api.EigorApi;

import com.infocert.eigor.api.EigorApiBuilder;

EigorApi api = new EigorApiBuilder()

.build();

We can now use the api to convert, for example, an Italian FatturaPA invoice in an UBL invoice. The usage is pretty straightforward, just call the convert() method passing

* the name of the format of the invoice that should be converted.
* the name of the format the source invoice should be converted into.
* an InputStream providing the content of the invoice.

import it.infocert.eigor.api.ConversionResult;

import it.infocert.eigor.api.IConversionIssue;

File sourceInvoice = new File("C:\\issue-245-fattpa.xml");

ConversionResult<byte[]> convert = api.convert(

"fatturapa",

"ubl",

new FileInputStream(sourceInvoice)

);

Finally, you can just access the ConversionResult object to have a list of issues and the final converted invoice:

List<IConversionIssue> issues = convert.getIssues();

for (IConversionIssue issue : issues) {

System.out.println(issue);

}

System.out.println( new String( convert.getResult() ) );

* 1. Configure EeISI Logging

Just write in the src/main/resources a valid Logback configuration file in order to set up the logging as you wish. If you're in doubt you can use this template:

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

<encoder>

<pattern>%d{mm:ss} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

<root level="error">

<appender-ref ref="STDOUT" />

</root>

<logger name="it.infocert" level="warn" />

</configuration>

* 1. Configure EeISI properties

Before launching the application, you should provide an eigor.properties file that define some key information.

Just create an empty eigor.properties file in the src/main/resources. This is enough to have EeISI to read that file.

*Please note that this is valid up to mapper version 4.2.0 but in next releases this feature will be enhanced and this step becomes optional. It is s only needed if the integrator wants to change some defaults, otherwise sensible defaults are used.*

Write the following content in the eigor.properties file:

eigor.workdir=${prop.java.io.tmpdir}eigor

eigor.validation-home=C:/tmp/eeisi/converterdata

eigor.converter.cen-cii.mapping.one-to-one=classpath:converterdata/converter-cen-cii/mappings/one\_to\_one.properties

eigor.converter.cen-cii.mapping.many-to-one=classpath:converterdata/converter-cen-cii/mappings/many\_to\_one.properties

eigor.converter.cen-cii.mapping.one-to-many=classpath:converterdata/converter-cen-cii/mappings/one\_to\_many.properties

eigor.converter.cen-cii.mapping.custom=classpath:converterdata/converter-cen-cii/mappings/custom.conf

eigor.converter.cen-cii.xsd=file://${eigor.validation-home}/converter-commons/cii/xsd/uncoupled/data/standard/CrossIndustryInvoice\_100pD16B.xsd

eigor.converter.cen-cii.schematron=file://${eigor.validation-home}/converter-commons/cii/schematron-xslt/EN16931-CII-validation.xslt

eigor.converter.cen-cii.schematron.auto-update-xslt=false

eigor.converter.cen-cii.cius=file://${eigor.validation-home}/converter-commons/cii/cius/schematron-xslt/EN16931-CIUS-IT-CIIValidation.xslt

eigor.converter.cen-cii.cius.auto-update-xslt=false

eigor.converter.cen-cii.guideline-context=urn:cen.eu:en16931:2017

eigor.converter.cii-cen.mapping.one-to-one=classpath:converterdata/converter-cii-cen/mappings/one\_to\_one.properties

eigor.converter.cii-cen.mapping.many-to-one=classpath:converterdata/converter-cii-cen/mappings/many\_to\_one.properties

eigor.converter.cii-cen.mapping.one-to-many=classpath:converterdata/converter-cii-cen/mappings/one\_to\_many.properties

eigor.converter.cii-cen.mapping.custom=classpath:converterdata/converter-cii-cen/mappings/custom.conf

eigor.converter.cii-cen.xsd=file://${eigor.validation-home}/converter-commons/cii/xsd/uncoupled/data/standard/CrossIndustryInvoice\_100pD16B.xsd

eigor.converter.cii-cen.schematron=file://${eigor.validation-home}/converter-commons/cii/schematron-xslt/EN16931-CII-validation.xslt

eigor.converter.cii-cen.schematron.auto-update-xslt=false

eigor.converter.cii-cen.cius=file://${eigor.validation-home}/converter-commons/cii/cius/schematron-xslt/EN16931-CIUS-IT-CIIValidation.xslt

eigor.converter.cii-cen.cius.auto-update-xslt=false

eigor.converter.ubl-cen.xsd=file://${eigor.validation-home}/converter-commons/ubl/xsd/UBL-Invoice-2.1.xsd

eigor.converter.ubl-cen.cius=file://${eigor.validation-home}/converter-commons/ubl/cius/schematron-xslt/EN16931-CIUS-IT-UBLValidation.xslt

eigor.converter.ubl-cen.cius.auto-update-xslt=false

eigor.converter.ubl-cen.schematron=file://${eigor.validation-home}/converter-commons/ubl/schematron-xslt/EN16931-UBL-validation.xslt

eigor.converter.ubl-cen.schematron.auto-update-xslt=false

eigor.converter.ubl-cen.mapping.one-to-one=classpath:converterdata/converter-ubl-cen/mappings/one\_to\_one.properties

eigor.converter.ubl-cen.mapping.many-to-one=classpath:converterdata/converter-ubl-cen/mappings/many\_to\_one.properties

eigor.converter.ubl-cen.mapping.one-to-many=classpath:converterdata/converter-ubl-cen/mappings/one\_to\_many.properties

eigor.converter.ubl-cen.mapping.custom=classpath:converterdata/converter-ubl-cen/mappings/custom.conf

eigor.converter.ublcn-cen.xsd=file://${eigor.validation-home}/converter-commons/ublcn/xsdstatic/UBL-CreditNote-2.1.xsd

eigor.converter.ublcn-cen.cius=file://${eigor.validation-home}/converter-commons/ubl/cius/schematron-xslt/EN16931-CIUS-IT-UBLValidation.xslt

eigor.converter.ublcn-cen.cius.auto-update-xslt=false

eigor.converter.ublcn-cen.schematron=file://${eigor.validation-home}/converter-commons/ubl/schematron-xslt/EN16931-UBL-validation.xslt

eigor.converter.ublcn-cen.schematron.auto-update-xslt=false

eigor.converter.ublcn-cen.mapping.one-to-one=classpath:converterdata/converter-ublcn-cen/mappings/one\_to\_one.properties

eigor.converter.ublcn-cen.mapping.many-to-one=classpath:converterdata/converter-ublcn-cen/mappings/many\_to\_one.properties

eigor.converter.ublcn-cen.mapping.one-to-many=classpath:converterdata/converter-ublcn-cen/mappings/one\_to\_many.properties

eigor.converter.ublcn-cen.mapping.custom=classpath:converterdata/converter-ublcn-cen/mappings/custom.conf

eigor.converter.cen-fatturapa.mapping.one-to-one=classpath:converterdata/converter-cen-fattpa/mappings/one\_to\_one.properties

eigor.converter.cen-fatturapa.mapping.many-to-one=classpath:converterdata/converter-cen-fattpa/mappings/many\_to\_one.properties

eigor.converter.cen-fatturapa.mapping.one-to-many=classpath:converterdata/converter-cen-fattpa/mappings/one\_to\_many.properties

eigor.converter.cen-fatturapa.mapping.custom=classpath:converterdata/converter-cen-fattpa/mappings/custom.conf

eigor.converter.cen-fatturapa.xsd=file://${eigor.validation-home}/converter-commons/fattpa/xsd/Schema\_del\_file\_xml\_FatturaPA\_versione\_1.2.xsd

eigor.converter.cen-fatturapa.field-lengths=file://${eigor.validation-home}/converter-cen-fattpa/mappings/field\_lengths.properties

eigor.converter.fatturapa-cen.mapping.one-to-one=classpath:converterdata/converter-fattpa-cen/mappings/one\_to\_one.properties

eigor.converter.fatturapa-cen.mapping.many-to-one=classpath:converterdata/converter-fattpa-cen/mappings/many\_to\_one.properties

eigor.converter.fatturapa-cen.mapping.one-to-many=classpath:converterdata/converter-fattpa-cen/mappings/one\_to\_many.properties

eigor.converter.fatturapa-cen.mapping.custom=classpath:converterdata/converter-fattpa-cen/mappings/custom.conf

eigor.converter.fatturapa-cen.xsd=file://${eigor.validation-home}/converter-commons/fattpa/xsdstatic/Schema\_del\_file\_xml\_FatturaPA\_versione\_1.2.xsd

eigor.converter.cen-ubl.mapping.one-to-one=classpath:converterdata/converter-cen-ubl/mappings/one\_to\_one.properties

eigor.converter.cen-ubl.mapping.many-to-one=classpath:converterdata/converter-cen-ubl/mappings/many\_to\_one.properties

eigor.converter.cen-ubl.mapping.one-to-many=classpath:converterdata/converter-cen-ubl/mappings/one\_to\_many.properties

eigor.converter.cen-ubl.mapping.custom=classpath:converterdata/converter-cen-ubl/mappings/custom.conf

eigor.converter.cen-ubl.xsd=file://${eigor.validation-home}/converter-commons/ubl/xsd/UBL-Invoice-2.1.xsd

eigor.converter.cen-ubl.schematron=file://${eigor.validation-home}/converter-commons/ubl/schematron-xslt/EN16931-UBL-validation.xslt

eigor.converter.cen-ubl.schematron.auto-update-xslt=false

eigor.converter.cen-ubl.customization-id=urn:cen.eu:en16931:2017

eigor.converter.cen-ublcn.mapping.one-to-one=classpath:converterdata/converter-cen-ublcn/mappings/one\_to\_one.properties

eigor.converter.cen-ublcn.mapping.many-to-one=classpath:converterdata/converter-cen-ublcn/mappings/many\_to\_one.properties

eigor.converter.cen-ublcn.mapping.one-to-many=classpath:converterdata/converter-cen-ublcn/mappings/one\_to\_many.properties

eigor.converter.cen-ublcn.mapping.custom=classpath:converterdata/converter-cen-ublcn/mappings/custom.conf

eigor.converter.cen-ublcn.xsd=file://${eigor.validation-home}/converter-commons/ublcn/xsdstatic/UBL-CreditNote-2.1.xsd

eigor.converter.cen-ublcn.schematron=file://${eigor.validation-home}/converter-commons/ubl/schematron-xslt/EN16931-UBL-validation.xslt

eigor.converter.cen-ublcn.schematron.auto-update-xslt=false

eigor.converter.xmlcen-cen.mapping.one-to-one=classpath:converterdata/converter-xmlcen-cen/mappings/one\_to\_one.properties

eigor.converter.xmlcen-cen.mapping.many-to-one=classpath:converterdata/converter-xmlcen-cen/mappings/many\_to\_one.properties

eigor.converter.xmlcen-cen.mapping.one-to-many=classpath:converterdata/converter-xmlcen-cen/mappings/one\_to\_many.properties

eigor.converter.xmlcen-cen.mapping.custom=classpath:converterdata/converter-xmlcen-cen/mappings/custom.conf

eigor.converter.xmlcen-cen.xsd=file://${eigor.validation-home}/converter-commons/xmlcen/xsdstatic/semanticCEN0.0.3.xsd

You mainly use this file to set up some key info.

* eigor.workdir is a mandatory property, it should refer to a folder in your filesystem where EeISI can work some temporary files used during the conversions.
* eigor.validation-home is not mandatory, if you look to the config closely, you'll note that it is just used as a placeholder in the definitions of other properties. Be sure that eigor.validation-home refers to the home of the validation repository you created before.

If you have knowledge of the Spring Framework you'll note that several properties refers to resources in your filesystem using the same syntax of Spring's ResourceLoader to seamlessly access resources seamlessly regardless of their location. If you need you can have further info in the official Spring Framework documentation.

* 1. Launch the Program

Launch the program you just deployed.

The resulting console should be similar to the following:

17:19:10.741 [main] DEBUG it.infocert.eigor.api.configuration.DefaultEigorConfigurationLoader - Skipping loading Eigor configuration from classpath resource '/eigor-test.properties' that does not exist.

17:19:10.752 [main] DEBUG it.infocert.eigor.api.configuration.DefaultEigorConfigurationLoader - Successfully loaded Eigor configuration from classpath resource '/eigor.properties'

17:19:10.911 [main] INFO com.infocert.eigor.api.EigorApiBuilder - Eigor

maven-version: 4.0.1-SNAPSHOT

git-branch: develop

git-revision: df700f909204e336760af138c22f21555c8b9db7

git-timestamp: 2019-01-29T13:40:29+0100

UBL\_OUT.SCH\_VALIDATION.INVALID - Schematron failed assert '[BR-E-01]-An Invoice that contains an Invoice line […] n XML element at '/\*:Invoice[namespace-uri()='urn:oasis:names:specification:ubl:schema:xsd:Invoice-2'][1]'. Fatal: true

<cbc:CustomizationID>urn:cen.eu:en16931:2017</cbc:CustomizationID>

<cbc:ID>123</cbc:ID>

...

</cac:InvoiceLine>

</Invoice>

Where you can identify:

1. the log, written out according to the Logback configuration file.
2. the issues discovered by EeISI during conversion, in this case problems related to the validity of the produced invoice related to the UBL schematron.
3. the output invoice as it was produced by EeISI.
4. Main Libraries of EeISI mapper

This is an overview of the main dependencies of EeISI.

* Logging
  + ch.qos.logback:logback-classic:jar:1.2.3
  + ch.qos.logback:logback-core:jar:1.2.3
  + log4j:log4j:jar:1.2.17
* Bug prevention
  + com.google.code.findbugs:annotations:jar:2.0.3
  + org.jetbrains:annotations:jar:13.0
  + com.google.code.findbugs:jsr305:jar:3.0.2
* Utility
  + org.apache.commons:commons-compress:jar:1.9
  + org.apache.commons:commons-compress:jar:1.9
  + org.apache.commons:commons-csv:jar:1.4
  + com.google.guava:guava:jar:20.0
  + commons-io:commons-io:jar:2.5
  + joda-time:joda-time:jar:2.9.9
  + org.reflections:reflections:jar:0.9.11
  + commons-logging:commons-logging:jar:1.1.3
  + net.minidev:accessors-smart:jar:1.1
* Schematorn Support
  + com.helger:ph-collection:jar:9.1.5
  + com.helger:ph-commons:jar:9.1.5
  + com.helger:ph-jaxb:jar:9.1.5
  + com.helger:ph-schematron:jar:5.0.6
  + com.helger:ph-xml:jar:9.1.5
* XML / XSD / Xpath Support
  + com.jayway.jsonpath:json-path:jar:2.2.0
  + javax.xml.bind:jaxb-api:jar:2.2.12-b140109.1041
  + jaxen:jaxen:jar:1.1.6
  + net.sf.saxon:Saxon-HE:jar:9.8.0-14
  + org.glassfish.jaxb:jaxb-core:jar:2.2.11
  + org.glassfish.jaxb:jaxb-runtime:jar:2.2.11
  + org.jdom:jdom2:jar:2.0.6
  + org.jvnet.staxex:stax-ex:jar:1.7.7
  + org.glassfish.jaxb:txw2:jar:2.2.11
* Automated Test / Unit Testing
  + junit:junit:jar:4.12
  + org.hamcrest:hamcrest-core:jar:1.3
  + org.hamcrest:hamcrest-core:jar:1.3
  + org.hamcrest:hamcrest-library:jar:1.3
  + org.powermock:powermock-api-mockito-common:jar:1.6.6
  + org.powermock:powermock-api-mockito:jar:1.6.6
  + org.powermock:powermock-api-support:jar:1.6.6
  + org.powermock:powermock-core:jar:1.6.6
  + org.powermock:powermock-module-junit4-common:jar:1.6.6
  + org.powermock:powermock-module-junit4:jar:1.6.6
  + org.powermock:powermock-reflect:jar:1.6.6
  + org.mockito:mockito-core:jar:1.10.19
  + org.assertj:assertj-core:jar:2.6.0
  + org.xmlunit:xmlunit-core:jar:2.2.1
  + org.xmlunit:xmlunit-matchers:jar:2.2.1

1. EeISI mapper interface usage details
   1. API interface usage

The EeISI framework provides a clean and simple API which allows applications to quickly integrate it in their workflow and start converting invoices.

**Conversion**

When a full conversion between two different formats is needed, it is necessary to call the conversion() method.

public ConversionResult<byte[]> convert(final String sourceFormat, final String targetFormat, final InputStream invoice)

This method converts the provided invoice assuming it is expresses in the given source format. The result contains the converted invoice, if the conversion process was able to complete the conversion and any error or warning that may be occurred during the process.

* **sourceFormat** is the format of the invoice to be converted and can be selected among: **ubl, ublcn, cii, fatturapa, xmlcen**
* **targetFormat** is the format the invoice will be converted to and can be selected among**: ubl, ublcn, cii, fatturapa, xmlcen, peppolbis, peppolcn**
* invoice is the invoice to convert.

If it is required to have a more detailed feedback on the conversion, it is possible to register some callbacks that are invoked when, during the conversion, noteworthy events occur.

**Validation**

EeISI gives the option to just validate a document. A document is considered valid when it would be converted without issues if fed to the conversion method.

EeISI has different validation methods.

public ConversionResult<Void> validate(final String sourceFormat, final InputStream invoice)

public ConversionResult<Void> validateSyntax(final String sourceFormat, final InputStream invoice)

public ConversionResult<Void> validateSemantic(final String sourceFormat, final InputStream invoice)

The first one fully validate a given document. A document that is valid according to this method can be transformed in the specified target format without issues. This method usually applies an XSD schema to guarantee the document is a valid document, a schematron to check whether it complies with the CEN model and finally, if needed, a CIUS schematron to check whether it matches the Italian constraints.

The other two methods work exactly like the validate method, already existing and implemented for the eIGOR project. The input parameters are therefore the format of the document to be validated (String) and the document itself (InputStream). The ConversionResult object is provided as output, providing information about the result of the conversion.

ValidateSynatx validates just the syntax of the provided document, usually applying the XSD, while validateSemantic applies also the schematron to check the compliance with the CEN model.

**Custom Validation**

Two methods are provided to execute custom validation checks based on schematron and XSDs.

customSchSchematronValidation(File schemaFile, InputStream xmlToValidate)

This method accepts a file descriptor referencing a schematron file and applies it to the given XML.

customXsdValidation(File schemaFile, InputStream xmlToValidate)

This method accepts a file descriptor referencing an XSD file and applies it to the given XML.

* 1. CLI (command line interface) usage

EeISI mapper has a cli module which is a self contained command line interface that uses EeISI to convert invoices. It is not needed to convert invoices with the eigor-api but it could be priceless to explore EeISI functionalities and to reproduce conversion scenarios

NAME

eigor[.bat|.sh] - transform an invoice in another format

SYNOPSIS

eigor[.bat|.sh] --input <original\_invoice> --source <original\_invoice\_format> --output <output\_folder> --target <transformed\_invoice\_format>

or with *“-- force"* if the conversion shall proceed even though errors are present

eigor[.bat|.sh] --input <original\_invoice> --source <original\_invoice\_format> --output <output\_folder> --target <transformed\_invoice\_format> --force

or with*” -- intermediate-validation”* if it is necessary to validate the intermediate XMLCEN semantic meta-language

eigor[.bat|.sh] --input <original\_invoice> --source <original\_invoice\_format> --output <output\_folder> --target <transformed\_invoice\_format> --intermediate-validation

DESCRIPTION

--input

Mandatory, path to the file of the invoice to transform.

--source

Mandatory, format of the invoice specified with '--input', as 'fatturapa', 'ubl', ...

--output

Mandatory, path of the folder where the converted invoice will be stored along with other files.

--target

Mandatory, format of the transformed invoice, support the same formats of '--source'.

--force

Optional, continue conversion process even if errors are encountered.

If not specified, the conversion process stops at the first problem detected.

--intermediate-validation

Optional, if specified, the intermediate cen xml is validated against its

xsd and schematrons.

EXAMPLES

./eigor.sh --input 012.xml --source fatturapa --output ~/transformed --target ubl --force

Transform the 'fatturapa' input file '012.xml' into a ubl invoice and stores the result in '~/transformed', without stopping because of errors.

eigor --input examples\ubl\UBL-Invoice-2.1-Example.xml --source ubl --output reports --target fatturapa

Transform the 'UBL' input file 'UBL-Invoice-2.1-Example.xml' into a 'Fattura PA' invoice and stores the result in 'reports', stopping when the first error occurs.

This example should work straight out of the box.

eigor.bat --source fatturapa --output .\reports --target ubl --force --input examples\fattpa\A10-Licenses.xml

* 1. Errors configuration
     1. Introduction

EeISI will produce a series of error codes to identify and understand error messages.

In order to provide the end user with the ability to customize the error messages, the EeISI error data structure will contain parameters with information about the error.

* + 1. Error Logical Structure
* sourceMessage:

The actual error description as detected at code level. Useful to analyze bugs.

Example: “XSD validation error at 4:32. cvc-complex-type.2.4.a: invalid content was found starting with element “cac:AccountingSupplierParty”. One of "{"urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2":Pro fileID, "urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents- 2":Profil eExecutionID, "urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2":ID}"., org.xml.sax.SAXParseException; lineNumber: 4; columnNumber: 32; cvc-complex- type.2.4.a: contenuto non valido che inizia con l'elemento "cac:AccountingSupplierParty". È previsto un elemento "{"urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2":Pro fileID, "urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents- 2":Profil eExecutionID, "urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2":ID}" expected.”

* severity:

A flag / label defining the severity of the bug.

Example: warning | error

* errorCode:

An alphanumeric string uniquely identifying the type of error. Please, read the following section about error codes for details about how an error code is built.

Examples

UBL\_IN.SCH\_VALIDATION.INVALID

* parameters

A set of values in a map like structure that better define the error. The list of parameters depends on the type of error.

* + 1. Error Codes

Error codes will be made up by three sections.

* **location:** A code depending on the module where the error occurred. (e.g. *converter-fattpa-cen*)
* **action:** What caused the error. It indicates what action was going on in the conversion when the error was thrown. (e.g. *XSDValidation*)
* **error:** The specific error that was detected.

Examples:

UBL\_IN.SCH\_VALIDATION.INVALID

An error code representing a problem occurred in the conversion of an incoming UBL invoice (UBLIN), while checking for schematron compliance that resulted in an invalid document.

FATTPA\_OUT.XSD\_VALIDATION.INVALID

An error code representing a problem occurred in the verification of a produced invoice in FATTPA format (FATTPAOUT), while checking for XSD compliance that resulted in an invalid document.

CSVCEN\_IN.GENERIC.INVALID

A generic error occurred while converting the incoming invoice to the CEN model.

* + 1. Error Parameters

Each error message carries some parameters that provide key metadata relative to the error that occurred.

For example, an XSD validation error has the following parameters:

* Each error will carry a set of parameters that better define the error. Every errorCode could have a different set of parameters.

Examples:

lineNumber: 4

invalidElement: “cac:AccountingSupplierParty”

expectedElements:

“urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents- 2":Profil eID, "urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents- 2":Profil eExecutionID, "urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents- 2":ID}"., org.xml.sax.SAXParseException; lineNumber: 4; columnNumber: 32; cvc- complex-type.2.4.a: contenuto non valido che inizia con l'elemento "cac:AccountingSupplierParty". È previsto un elemento "{"urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents- 2":Pro fileID,

"urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2":Profil eExecutionID, "urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2":ID

Using those parameters it’s possible for an end user to translate and/or adapt the error message to

better suit the needs.

For example, by translating it in another language:

“Erreur de validation XSD: ligne ${lineNumber}. Contenu non valide trouvé en commençant par

l’élément ${invalidElement}, un de [${expectedElements}] prevu.”

Or outputting it in another format, like XML:



<error>

<message>

${message}

</message>

<level>error</level>

<error-code>${errorCode}</error-code>

<params>

<line-number>${lineNumber}</line-number> <invalid- element>${invalidElement}</invalid-element>

<expected-elements>

${expectedElements}

</expected-elements>

</params>

</error>

Each error message can carry a different set of parameters. Refer to the error code reference for the complete list.

* + 1. Examples

CIUS Validation Error

* **errorCode:** UBL\_IN.SCH\_VALIDATION.MISSING\_VALUE
* **severity:** warning
* **sourceMessage:** Schematron failed assert '[BR-02]-An Invoice shall have an Invoice number (BT-1).' on XML element at '/\*:Invoice[namespace- uri()='urn:oasis:names:specification:ubl:schema:xsd:Invoice-2'][1]'.,java.lang

.Exception: Schematron failed assert '[BR-02]-An Invoice shall have an Invoice number (BT-1).' on XML element at '/\*:Invoice[namespace- uri()='urn:oasis:names:specification:ubl:schema:xsd:Invoice-2'][1]'.

#### parameters:

* **ruleNumber:** BR-02
* **cenElement:** BT-1
* **xmlElement:** /\*:Invoice[namespace- uri()='urn:oasis:names:specification:ubl:schema:xsd:Invoice-2'][1]

FatturaPA Element Conversion Error

* **errorCode:** FATTPA\_OUT.GENERIC.ILLEGAL\_VALUE
* **errorLevel:** error
* **message:** No CedentePrestatore was found in current FatturaElettronicaHeader

#### params:

* **missingElement:** CedentePrestatore
* **parentElement:** FatturaElettronicaHeader
  + 1. Error codes details

|  |  |  |  |
| --- | --- | --- | --- |
| **Error Codes** | **Number of parameters** | **Parameters** |  |
| UBL-IN.XSD-VALIDATION.INVALID | 1 | sourceMsg |  |
| UBL-IN.SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| UBL-IN.CIUS-SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| UBL-IN.CONFIGURED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| UBL-IN.CONFIGURED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| UBL-IN.CONFIGURED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| UBL-IN.HARDCODED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| UBL-IN.HARDCODED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| UBL-IN.HARDCODED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| UBL-IN.GENERIC.INVALID | 1 | sourceMsg |  |
| UBL-OUT.SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| UBL-OUT.CIUS-SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| UBL-OUT.CONFIGURED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| UBL-OUT.CONFIGURED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| UBL-OUT.CONFIGURED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| UBL-OUT.HARDCODED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| UBL-OUT.HARDCODED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| UBL-OUT.HARDCODED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| UBL-OUT.GENERIC.INVALID | 1 | sourceMsg |  |
| FATTPA-IN.XSD-VALIDATION.INVALID | 1 | sourceMsg |  |
| FATTPA-IN.CONFIGURED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| FATTPA-IN.CONFIGURED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| FATTPA-IN.CONFIGURED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| FATTPA-IN.HARDCODED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| FATTPA-IN.HARDCODED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| FATTPA-IN.HARDCODED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| FATTPA-IN.GENERIC.INVALID | 1 | sourceMsg |  |
| FATTPA-OUT.CIUS-SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| FATTPA-OUT.CONFIGURED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| FATTPA-OUT.CONFIGURED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| FATTPA-OUT.CONFIGURED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| FATTPA-OUT.HARDCODED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| FATTPA-OUT.HARDCODED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| FATTPA-OUT.HARDCODED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| FATTPA-OUT.GENERIC.INVALID | 1 | sourceMsg |  |
| CII-IN.XSD-VALIDATION.INVALID | 1 | sourceMsg |  |
| CII-IN.SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| CII-IN.CIUS-SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| CII-IN.CONFIGURED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| CII-IN.CONFIGURED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| CII-IN.CONFIGURED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| CII-IN.HARDCODED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| CII-IN.HARDCODED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| CII-IN.HARDCODED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| CII-IN.GENERIC.INVALID | 1 | sourceMsg |  |
| CII-OUT.XSD-VALIDATION.INVALID | 1 | sourceMsg |  |
| CII-OUT.SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| CII-OUT.CIUS-SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| CII-OUT.CONFIGURED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| CII-OUT.CONFIGURED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| CII-OUT.CONFIGURED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| CII-OUT.HARDCODED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| CII-OUT.HARDCODED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| CII-OUT.HARDCODED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| CII-OUT.GENERIC.INVALID | 1 | sourceMsg |  |
| PEPPOL-OUT.SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| PEPPOL-OUT.CIUS-SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| PEPPOL-OUT.CONFIGURED-MAP.ILLEGAL-VALUE | 1 | sourceMsg |  |
| PEPPOL-OUT.CONFIGURED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| PEPPOL-OUT.CONFIGURED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| PEPPOL-OUT.HARDCODED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| PEPPOL-OUT.HARDCODED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| PEPPOL-OUT.HARDCODED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| XMLCEN-OUT.GENERIC.INVALID | 2 | sourceMsg | offendingItem |
| XMLCEN-IN.XSD-VALIDATION.INVALID |  |  |  |
| XMLCEN-IN.SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| XMLCEN-IN.CIUS-SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| XMLCEN-IN.CONFIGURED-MAP.ILLEGAL-VALUE | 1 | sourceMsg |  |
| XMLCEN-IN.CONFIGURED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| XMLCEN-IN.CONFIGURED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| XMLCEN-IN.HARDCODED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| XMLCEN-IN.HARDCODED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| XMLCEN-IN.HARDCODED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| XMLCEN-IN.GENERIC.INVALID | 2 | sourceMsg | offendingItem |
| XMLCEN-OUT.SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| XMLCEN-OUT.CIUS-SCH-VALIDATION.INVALID | 1 | sourceMsg |  |
| XMLCEN-OUT.CONFIGURED-MAP.ILLEGAL-VALUE | 1 | sourceMsg |  |
| XMLCEN-OUT.CONFIGURED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| XMLCEN-OUT.CONFIGURED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| XMLCEN-OUT.HARDCODED-MAP.ILLEGAL-VALUE | 2 | sourceMsg | offendingItem |
| XMLCEN-OUT.HARDCODED-MAP.INVALID | 2 | sourceMsg | offendingItem |
| XMLCEN-OUT.HARDCODED-MAP.MISSING-VALUE | 2 | sourceMsg | offendingItem |
| XMLCEN-OUT.GENERIC.INVALID | 2 | sourceMsg | offendingItem |

Table 1 Error codes list