Qin Jin

FINVOICE GENERATING

A Dynamic-link library generating Finvoice XML documents

FINVOICE GENERATING

A Dynamic-link library generating Finvoice XML documents

Qin Jin
Bachelor's thesis
Spring 2013
Degree Program in Information Technology
Oulu University of Applied Sciences

PREFACE

This Bachelor’s thesis was done at Oulu University of Applied Sciences, Raahe School of Engineering and Business during the spring in 2013. The work was proposed by Jukka Penttilä of Sunwell Trade Oy.

I would like to thank my thesis instructor Mrs. Lea Hannila for guidance and encouragement during the thesis work. I would also like to show my appreciation for the help and instructions in the technical parts from Dr. Lauri Pirttiaho. My thanks to Mr. Jarmo Karppelin, for assigning Mrs. Hannila as my supervisor, and Mrs. Kaija Posio for the language checking. My gratitude for all the teachers and staff in Raahe for the years I have had the pleasure of spending there.

Last but not least, I appreciate the huge support from my family and friends.

Raahe, May 2013

Qin Jin

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Tekijä: Qin Jin
Opinnäytetyön nimi: Finvoice Generating
Työn ohjaaja: Lea Hannila
Työn valmistumislukukausi ja -vuosi: Kevät 2013 Sivumäärä: 43+22

Tämä opinnäytetyön aiheen tarjosi Jukka Penttilä Sunwell Trade Oy:stä. Työn tavoitteena oli tehdä Dynamic-link library eli ohjelmistokirjasto, jota REX-niminen kassaohjelma tulisi käyttämään Finvoice-laskujen luomiseen. REX-ohjelmisto vaatii muutoksia, jotta se voi käyttää kyseistä DLL:ää. Muutostyö ei ole tämän opinnäytetyön aihe, vaan työssä keskitytään vain DLL:n luomiseen.

Tämän työn tekemistä varten tekijällä piti olla aikaisempaa tietoa ja taitoja käyttää C#-ohjelmointikieltä ja Extensible Markup Languagea, XML. DLL on tehty C#-kielellä käyttämällä Visual Studio 2010-ohjelmointityökaluja. Notepad++:aa käytettiin XML-tiedostojen tarkastuksessa ja editoinnissa. Työssä tehtiin myös testiohjelman C#-kielellä, jossa testaan DLL:n toimivuutta.

Opinnäytetyön tuotos on DLL-tiedosto, joka kykenee luomaan XML tiedostoja, jotka vastaavat Finvoice soveltamisohje 2.0 version määritelmiä. Validi XML-tiedosto voidaan näyttää paperilaskuna Internet Explorer-selainohjelmalla. Finvoice-soveltamisohjeen mahdollisesti muuttuessa tulevaisuudessa ainoa asia, joka pitää päivittää, on Finvoice-skeema, jota käytetään resurssina DLL:ssä.

Asiasanat: Finvoice, sähköinen laskutus, DLL, XML, C#

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology

Author: Qin Jin
Title of Bachelor's thesis: Finvoice Generating
Supervisor: Lea Hannila
Term and year of completion:Spring 2013 Number of pages: 43+22

The topic of this Bachelor's thesis was offered by Jukka Penttilä from Sunwell Trade. The aim was to build a Dynamic-link library that can be used by a billing system called REX to generate valid Finvoice documents. REX required some modifications to use the DLL. But for this thesis work, it was only about implementation of the DLL.

To be able to do the thesis work one needed prior knowledge and skills in C# programming and Extensible Markup Language (XML). The Dynamic-link library was implemented in C# language using Visual Studio 2010 development tools. The Notepad++ was used for displaying and editing XML documents. I also built an application in C# to test the DLL.

The result of my thesis work is a DLL with the functionality of generating XML documents which match the specifications according to the Finvoice Implementation Guidelines, Version 2.0. The valid output, a Finvoice XML file, can be displayed as a regular paper invoice by Internet Explorer. If the Finvoice standards change in the future, the only change that has to be made is to update the Finvoice schema, which is stored as a resource in the DLL.

Keywords: Finvoice, e-invoicing, DLL, XML, C#

Abbreviation Table

Electronic Data Interchange	EDI
Extensible Markup Language	XML
European Union Value Added Tax	EU VAT
Finnish Financial Services	FFI
Point of Sales	POS
Portable Document Format	PDF
Dynamic-link Library	DLL
Application Programming Interface	API
XML definition language	XSD
Extensible Stylesheet Language	XSL
Unified Modelling Language	UML
Microsoft Developer Network	MSDN

Table of contents

1 INTRODUCTION

The idea of e-invoicing is not new. The first electronic invoices were sent over 30 years ago using electronic data interchange (EDI). (GXS, date of retrieval 7.4.2013a). In the past, electronic invoice was scanned as a paper-based invoice and issued via emails. Even though you remove paper invoices from the process it is not fully automated. Nowadays, e-invoice is an electronic bill, which passes the information between suppliers and buyers in a machine-readable language, which is typically Extensible Markup Language (XML). It is processed by e-invoicing systems which save time and cost, and reduce data errors when transferring invoices over different media. Electronic invoicing allows businesses to be green, productive and service-oriented (FFI 2010, date of retrieval 11.4.2013).

According to the latest European Union Value Added Tax (EU VAT) directive all invoices must contain a minimum set of data. This directive is then interpreted by individual member states who can apply their own criteria. (GXS, date of retrieval 7.4.2013b.)

Finvoice is an e-invoicing standard in Finland. It is managed by the Federation of Finnish Financial Services (FFI). It is the most commonly used e-invoicing format in Finland. Finvoice documents can be delivered securely and safely from senders to recipients via banks, as Figure 1 shows below:

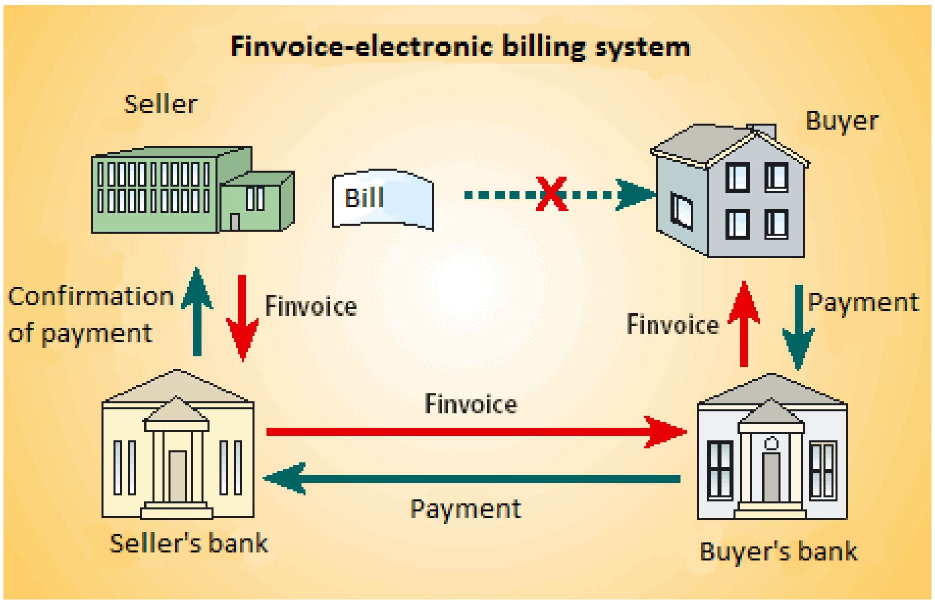


FIGURE 1. Finvoice e-billing system (Huhtanen 2003, 13)

In the Finvoice billing system, the seller and buyer agree together and with their respective banks about using the electronic billing system. The seller sends the electronic bill to its bank, which forwards it to the client's bank. The client's bank will deliver the bill for the customer to be paid.

The Point of Sales (POS) application called REX from Finnish Integrated Retail Systems Oy was invented in 1999. It has functions for invoicing the customers, but it can only produce paper invoices or Portable Document Format (PDF) invoices. Therefore, removing paper and automating the invoicing process will reduce costs and save time significantly. REX with an ability of e-invoicing automation will improve its business efficiency and enable the integration with other business systems.

Jukka Penttilä, an independent Retail System consultant, from Sunwell Trade, offered a topic to make a Dynamic-link library (DLL) which REX can call with the application programming interface (API) generating the Finvoice documents. In order to use the class library, REX requires some modifications and integration with the DLL. But for this thesis work, it is only about implementation of the DLL.

2 THE WORK ENVIRONMENT

This chapter goes through the programming languages and tools used during the work.

2.1 Extensible Markup Language (XML)

XML is a markup language. It provides a way to describe structured data (MSDN, date of retrieval 11.5.2013b). XML is the most common language for data transmissions due to the format in which the data is stored. It can be read by different incompatible applications. It is also readable for human beings. All the data is stored as XML elements that identify the data and carry the actual data.

XML messages consist of the following types of entities: elements, aggregates and structures. An element is a simple entity including only one string of data. An aggregate is a structural entity consisting of elements. A structure is a more complicated entity that includes either aggregates or both aggregates and elements. (FFI 2012, 60)

The outcome of the DLL for this thesis work is an XML file in the Finvoice standard which can contain hundreds of elements agreed with organizations and their applications. To guarantee that the output, an XML file, contains the valid structure, data content and relationships between them, it must be validated by the Finvoice schema.

The Finvoice schema, published by FFI, is an XML definition language (XSD). It is an XML-based file and it is used to define and validate the content and structure of XML documents. It defines elements and child elements (order, amount), attributes that can appear in an XML document. Organizations wanting to exchange data should build their applications so that they are able to produce and consume Finvoice-formatted XML files by using the Finvoice schema.

Notepad++ was chosen to display and edit XML documents for this thesis work. It is a text editor and source code editor that supports several languages for the Windows environment.

Comparing the notepad bundled with Window Operating System and any other web browsers to it, notepad++ is more convenient. It supports syntax highlighting and code folding. Users can also configure the font and syntax highlighting for each element. Therefore, it makes displaying the complex Finvoice documents a lot easier.

2.2 C#

C# (pronounced "C sharp") is a programming language that is designed for building a variety of applications that run on the .NET Framework. C# is simple, powerful, type-safe, and object-oriented. (MSDN, date of retrieval 19.2.2013) The .NET Framework provides a runtime environment and libraries for C# and some other languages. The .NET Framework class library provides a comprehensive and integrated collection of classes, interfaces and value types, especially for building XML involved applications. Therefore, there is no need to create structured text documents, like XML, by a text editor and process them as strings by manipulation functions (Ian, Matthew & Jesse Liberty 2010, 452).

C# is also introduced as Visual C# in the Microsoft Visual Studio. It supports Visual C# with a full-featured code editor, compiler, project templates, designers, code wizards, a powerful and easy-to-use debugger, and other tools (MSDN, date of retrieval 19.2.2013). Visual Studio fully supports XML. It provides tools and features to make it easier to work with XML, EXtensible Stylesheet Language (XSL), and XML schemas. Microsoft Visual Studio 2010 was used for implementation of this thesis work.

2.3 Unified Modeling Language (UML)

Unified Modelling Language (UML) is a standardized general-purpose modelling language in the field of object-oriented software engineering. The Unified Modelling Language includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems. (Wikipedia, date of retrieval 25.3.2013)

Several kinds of UML diagrams were used to specify, visualize and construct the development of this project, including the structure and design. Microsoft Visio was used to draw UML diagrams.

3 DEFINITION

REX system includes two applications: Kassa and Konttori. Kassa is used for selling the items in the store by cashiers. It can generate bills to either existing clients or new ones. It can store customer information into a database and fetch it, too. The information of customers, suppliers, products, etc. is managed by Konttori which is similar to an accounting application. But REX system is isolated, not connected to any bank system, which means it cannot send bills to clients' bank account or receive bills from suppliers' bank account via Internet. To enable the functionality of e-billing or e-invoicing for REX system, the DLL I have created has to be integrated to the program.

A DLL is an executable file that acts as a shared library of functions. (MSDN, date of retrieval 26.3.2013a). Hence, by referencing a DLL that has functionality for generating valid Finvoice documents, REX can extend its ability without big changes in the program. Those two components can be modularized into one program by using DLL, as shown in Figure 2 below:

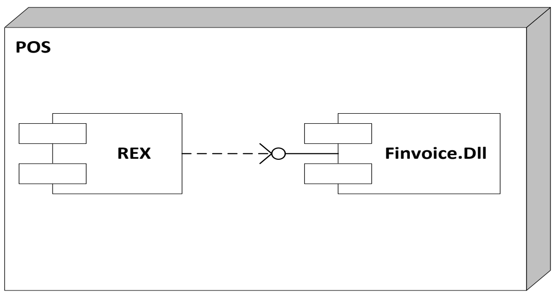


FIGURE 2. Components diagram of REX system in the future

The thesis is all about making the Finvoice.dl. The requirement of the DLL is that REX can call it with an interface and create a valid Finvoice document. Only when the program is requested to generate an e-invoice by users, the Finvoice.dll will be loaded into the main program at run time.

Because they are separated modules, the loading time of the program is shorter. In addition, when there is a new standard of Finvoice document, the program can be updated easily by just changing the recourse and code in Finvoice.dll.

4 IMPLEMENTATION

Iimplementing a DLL basically contains three steps: creating a DLL project, adding classes into it and adding it to the reference. This chapter explains the details how the Finvoice.dll was built.

4.1 Finvoice technical description

Technically, a Finvoice document is an XML file used as an electronic invoice. XML enables the invoice to be represented both in a form understood by the application and, using a browser, in a form corresponding to a paper invoice. The browser representation of an invoice may be printed as a hard copy and processed in the traditional way. (FFI 2012, 3)

The structure of Finvoice XML message is shown in appendix 1. The entity inside a bold rectangle means that it is mandatory. It must occur at least once. The entity with a dotted line rectangle means that it is not mandatory. It may occur once, or several times. Only when the XML contains all the mandatory data in a correct format, the file will be validated.

The Finvoice XML file is specified by an XML schema defined by the FFI. It describes all allowable content, valid type, occurrence, default value, etc. The entire XML file generated by Finvoice.dll must be validated against this schema, which is saved as a source file in the Finvoice.dll. The schema could also refer to a uniform resource locator (URI) dynamically so that the schema can update itself. Then REX system should require a permission to access the Internet.

4.2 Programming Logic

Since the Finvoice XML files may contain a large amount of entities and complex structures, the allocation of the input data into the right node is the biggest difficulty of this thesis work. Depending on the format of the input data provided by REX, there are a few assumptions of the solution.

One of my assumptions was that the input was a file stream that contains all the data which has to be written into a Finvoice document. Then the DLL should first map all the data with identifiers which are the same as the names of the elements defined in the Finvoice schema. Each data pair, an identifier and actual data, will be stored in a container. And then make a template of the Finvoice document that has nodes of all the elements but without content. So to allocate all the data, you just need to match their identifiers with the names of elements in the template. But due to the complex relationship between the parent elements, child elements and sibling elements, it is possible to implement, but too complicated.

Then I assumed that the input was just a pair of strings. The pair includes a string for path and a string for value. The elements will be added one at a time into an empty Finvoice document. Therefore in this way the program can avoid dealing with the complex relationships. And there will not be any elements without contents in the document.

Therefore, as the following figure shows, Finvoice.dll provides an interface to interact with REX via exported methods: CreateFinvoiceDocument, PutElementAttribute, ValidateXml and saveFinvoiceDocument.

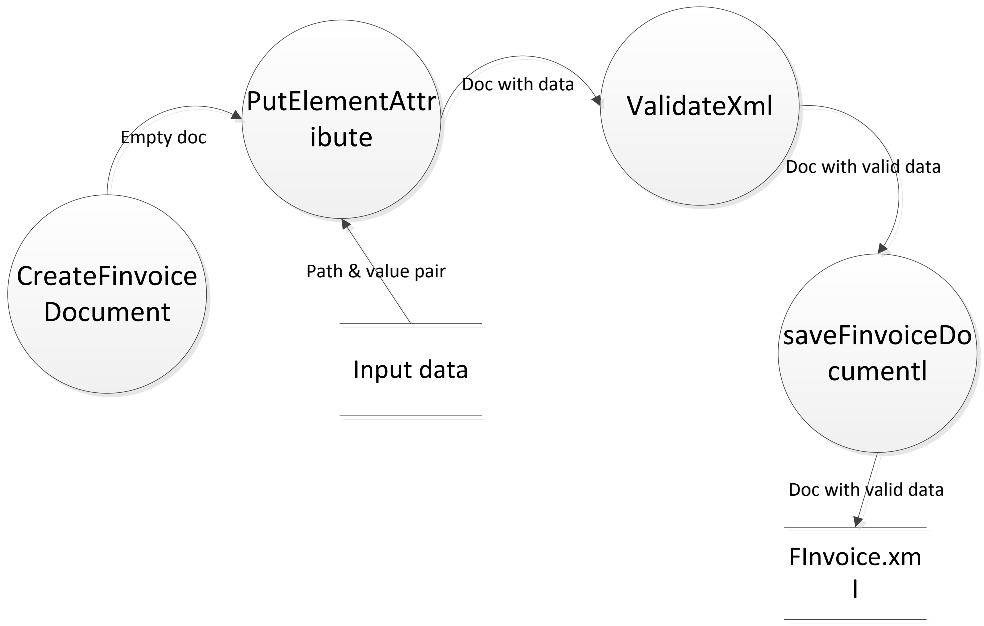


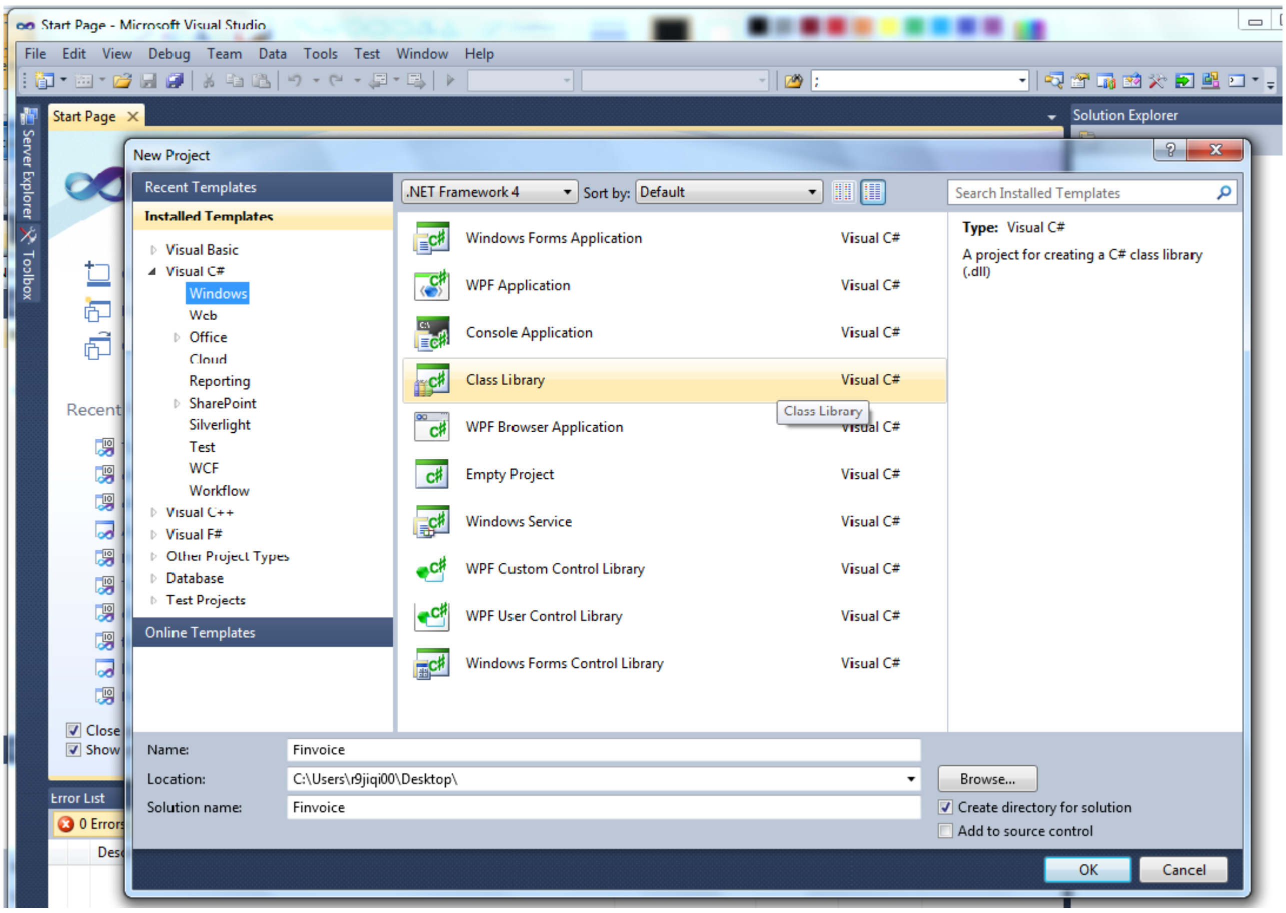
FIGURE 3. Data flow diagram of Finvoice.dll

Each process will be triggered by function calls from REX. The methods CreateFinvoiceDocument and ValidateXml should only be called once per a Finvoice document. Depending on the amount of input pairs, the PutElementAttributes will be called as many times as there are element pairs. In the end, if there are no errors, the result is a valid Finvoice XML file formatted nicely and then saved onto the hard drive.

4.3 FinvoiceDocument class

Before starting to program, a project must be created. Here is a walkthrough of creating a class library (.dll) using Visual Studio 2010:

1. Start Visual Studio 2010
2. From the File menu, select New and then Project. Or from Start Page, select New Project
3. From the New Project pane, choose Visual C# and select Windows under it, then select Class Library
4. Choose a name and location for the Class Library, the click on OK button, see FIGURE 4. Then a DLL is created.



.NET Framework 4 Sort by: Default

Windows Forms Application	Visual C#
WPF Application	Visual C#
Console Application	Visual C#
Class Library	Visual C#
WPF Browser Application	Visual C#
Empty Project	Visual C#
Windows Service	Visual C#
WPF Custom Control Library	Visual C#
WPF User Control Library	Visual C#
Windows Forms Control Library	Visual C#

Type: Visual C#

A project for creating a C# class library (.dll)

Name: Finvoice

Location: C:\Users\r9jiqi00\Desktop\

Solution name: Finvoice

☑ Create directory for solution

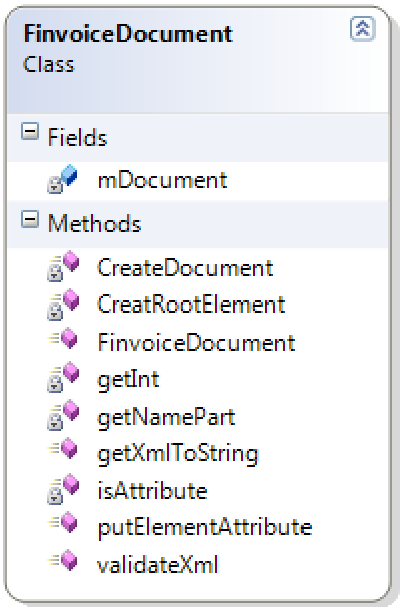
☐ Add to source control

FIGURE 4. Screenshot of Visual Studio 2010

The name given to the Class Library, Finvoice, will be the name of a namespace. By declaring the Finvoice namespace, it is easier to organize the scope of code integrated with a large program. It is declared by the keyword: namespace, as in the following example:

```
namespace Finvoice
{
    public class Class1{ }
}
```

After a DLL file is created, there is a public class, Class1 created by default. It was renamed with a more sensible name in my DLL, FinvoiceDocument. It has a private field called mDocument and a few methods, as FIGURE 5 shows below:


FinvoiceDocument
Class

Fields

mDocument

Methods

CreateDocument
CreatRootElement
FinvoiceDocument
getInt
getNamePart
getXmlToString
isAttribute
putElementAttribute
validateXml

FIGURE 5. Class diagram of FinvoiceDocument generated by VS 2010

4.4 FinvoiceDocument Field

The FinvoiceDocument class needs a data field to fulfill the Finvoice XML features and store them into it. Hence, this field should be an object of a type which can represent XML documents. In .NET Framework, there are two suitable classes: XmlDocument Class and XDocument Class. They both can represent XML documents but they are different in some ways:

- Version. XDocument class came with .NET Framework 3.5 and XmlDocument was before that. To be able to install and use .NET Framework 3.5 or even newer versions of it, the development environment must fulfill some requirements, such as a processor, RAM, operating system, etc. Therefore, when using .NET 3.0 or an older version of it, XmlDocument is the only option.
- Inheritance. XDocument is in the namespace System.Xml.Linq which contains the classes for LINQ to XML. LINQ to XML is an in-memory XML programming interface that enables you to modify XML documents efficiently and easily (MSDN, date of retrieval 26.3.2013). The inheritance hierarchy is as follows:

  System.Object

    System.Xml.Linq.XObject

System.Xml.Linq.XNode
  System.Xml.Linq.XContainer
    System.Xml.Linq.XDocument

XmlDocument is in namespace System.Xml which provides standards-based support for processing XML (MSDN, date of retrieval 11.3.2013). This class implements the W3C Document Object Model (DOM) Level 1 Core and the Core DOM Level 2. The DOM is an in-memory (cache) tree representation of an XML document and enables the navigation and editing of this document. (MSDN, date of retrieval 9.4.2013b). The inheritance Hierarchy is shown below:

System.Object
  System.Xml.XmlNode
    System.Xml.XmlDocument

- Processing XML documents with LINQ to XML is in general easier than with DOM API. Here is an example of how to create the same XML document by XDocument and XmlDocument:

  Here is a simple XML file:

```
<root attribute="value">
     <child1>Child1</child1>
     <child2>Child2</child2>
</root>
```

To create this XML file by XmlDocument:

```
 XmlDocument doc = new XmlDocument();
 XmlElement root = doc.CreateElement("root");
 root.SetAttribute("attribute", "value");
 XmlElement child1 = doc.CreateElement("child1");
 child1.InnerText = "child1";
 XmlElement child2 = doc.CreateElement("child2");
 child2.InnerText = "child2";
 root.AppendChild(child1);
 root.AppendChild(child1);
 doc.AppendChild(root);
```

To create this XML file by XDocument:

```
XDocument doc = new XDocument(
                new XElement("root",
                    new XAttribute("attribute", "value"),
                    new XElement("child1", "child1"),
                    new XElement("child2", "child2")));
```

In conclusion, the field which represents the FinvoiceDocument is designed to be a type of XDocument object.

4.5 Create FinvoiceDocument

This process is used to generate an XML file only with the root element. It generates an instance of FinvoiceDocument class that has an attribute of XDocument, which represents an XML document. The class constructor calls two internal functions: CreateDocument and CreateRootElemnt. CreateDocument declares the XML version, encoding type and initializing XML processing instruction. CreateRootElement creates a root element called Finvoice, which is an instance of XElement class, which represents an XML element. Root element Finvoice has three attributes: the version, the namespace, where the schema comes from, and the declaration of schema. Here is the data field and the constructor of the FinvoiceDocument class:

```
public class FinvoiceDocument {
        private XDocument mDocument;
        public FinvoiceDocument(){
            mDocument = CreateDocument();
            mDocument.Add(CreatRootElement());
        }
}
```

As the above code shows, the constructor has the same name as the class. When the class is instantiated, its constructor is called. A new object of the FinvoiceDocument type is initialized by two methods: CreateDocument and CreateRootElement. They are both private methods used only by this class. The following message sequence char shows how the FinvoiceDocument is created:

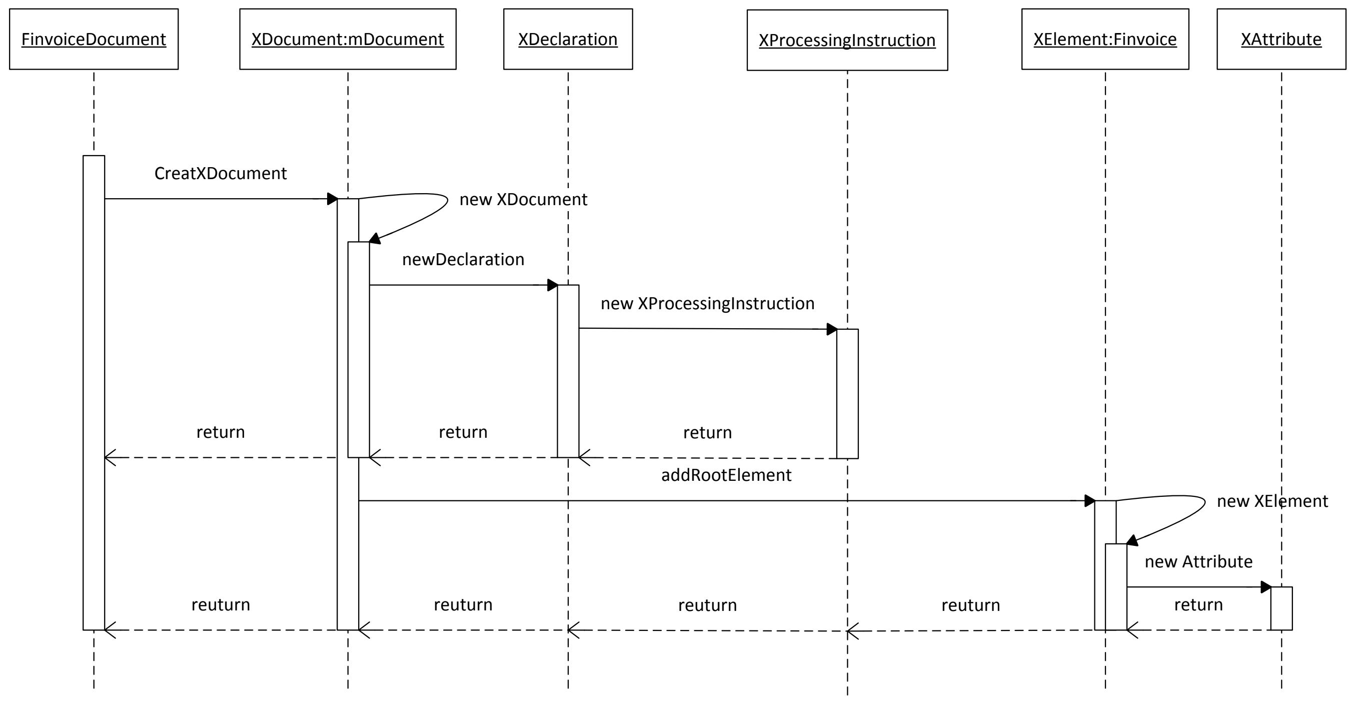


FIGURE 6. Message sequence char of creating FinvoiceDocument

4.5.1 Create XDocuments

CreateDocument is a private static method, which returns an XDocument object with the declared information of the XML version, the encoding type and the XML processing instruction. It is used to assign this object to the data field of the class, mDocument.

To be a valid Finvoice XML file according to its schema, there are a couple of properties which must be declared: the version of XML and the type of encoding. For example, the declarations in the beginning of the Finvoice XML file like this:

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<?xml-stylesheet type="text/xsl" href="Finvoice.xsl"?>
```

The first line shows that the Finvoice XML files use the XML version 1.0 and it is specified with ISO-8859-15 which is a single-byte encoding type. Because the Finvoice XML file may contain non ASCII characters, such as ö, ä, å, which will occur as an error like “An invalid character was found in the text content”, to specify the XML encoding can avoid these kinds of errors.

The first line could be implemented by using XDeclaration, which is a property of XDocument. XDeclaration represents an XML declaration. Here is the syntax for XDeclaration in C#:

```
public XDeclaration(
        string version,
        string encoding,
        string standalone
)
```

It initializes a new instance of the XDeclaration class with the specified version, encoding, and standalone status. Standalone is a string containing “yes” or ”no” that specifies whether the XML is standalone or requires external entities to be resolved. (MSDN, date of retrieval 24.4.2013a) For the Finvoice document, standalone status is null.

The second line in the Finvoice declaration example shows the XML processing instruction. It specifies the media type for a style sheet as text/xsl. It also specifies which specific XSL file is used for transforming the Finvoice XML file. XSL is a style sheet language for XML documents. With the underlying XSL file, a Finvoice XML file placed in the same directory can be transformed as a regular paper invoice and displayed by Internet Explorer (at least version 6.0).

XProcessingInstruction class can represent a style sheet instruction. Here is the syntax for XDeclaration in C#:

```
public XProcessingInstruction(
        string target,
        string data
)
```

It initializes a new instance of the XProcessingInstruction class (MSDN, date of retrieval 23.4.2013b). The first parameter contains the target application which in this case is xml-stylesheet. And the second parameter is the content of a processing instruction, the type and file name.

4.5.2 Create Root Elements

After the Finvoice document is created and declared, a root element must be added before processing the input data. The method CreateRootElement is a private static function that returns an XElement type object. The data field, mDocument, calls a public method Add (Object), which adds the XElement returned by the method CreateRootElement to mDocument.

The root element of Finvoice XML document called Finvoice is an instance of XElement which represents an XML element. It is initialized by the operator new. The root element Finvoice also has three attributes: the version, the namespace and the location of associated schema. To create an element with attributes in LINQ to XML is simple, as the following example shows:

```
XElement person = new XElement("Person",
                       new XAttribute("name", "Jim"),
                       new XAttribute("age", "23"),
                    new XAttribute("gender", "male" ));
```

And the example produces the following output:

```
<Person name="Jim" age="23" gender="male">
```

Because processing the Finvoice XML document in the by Finnish bank system requires a specific control of a namespace prefix, it is important that the Finvoice XML files are serialized with certain prefixes as can be seen in the following line:

```
<Finvoice Version="2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Finvoice.xsd">
```

To create an attribute that declares a namespace with a prefix, you will have to create an attribute where the name of the attribute is the namespace prefix, and this name is in the Xmlns namespace. The value of this attribute is the URI of the namespace. (MSDN, date of retrieval 23.4.2013a). Here is an example showing how this kind of attribute is generated in LINQ to XML:

```
XNamespace aw = "http://www.w3.org/2001/XMLSchema-instance";
XNamespace noNamespace = XNamespace.Get("Finvoice.xsd");
XElement root = new XElement("Finvoice",
                 new XAttribute(XNamespace.Xmlns + "xsi", aw),
                new XAttribute(aw + "noNamespaceSchemaLocation",
noNamespace));
```

The output of this code example will be the same as the previous example of a Finvoice element. XNamespace is a type representing an XML namespace. Xmlns is a property which gets the XNamespace object that corresponds to the xmlns URI (http://www.w3.org/2000/xmlns/). (MSDN, date of retrieval 24.4.2013b). The attribute noNamespaceSchemaLocation means the XML schema referenced in this attribute does not have a namespace.

4.6 Put Elements and Attributes

This is a public method that creates child elements and their attributes into the "empty" Finvoice XML document. It should be called in the order that the Finvoice schema defines. This function takes two parameters: Path and Value. They are both strings provided by REX. The path tells where the element or attribute should be and what value will be assigned to it. The value is the actual content of the element or attribute. The syntax shows as below:

```
public void putElementAttribute( string path, string value)
```

The parameter Value is the actual content of elements or attributes. It has to be a certain type which is defined by the Finvoice schema. The parameter Path is used as a navigator. It is similar to the XPath expression which uses a path notation (MSDN, date of retrieval 15.4.2013). But their syntax is slightly different. The following table shows the comparison between XPath and Path in this thesis work.

path expression	XPath	Path
element	/nodename	/nodename
attributes	[@nodename]	@nodename
array	nodename[interger]	nodename[integer]

TABLE 1. Comparison between XPath and Path

The only difference is how to address attributes. For example, to express a <Person> element with an attribute age in XPath is /Person [@age]. But the Path defined by me is /Person/@age. The reason why the Path expression is designed in a different way is to make the parsing Path and then addressing elements and attributes easily.

The Path is a string that contains characters, intergers and notations including '/', '@' and '[]'. The Path can be described in the Backus Normal Form:

```
<Path> :: = <node>* <final_node>
<node> ::= '/'<element>
<element> ::= <single_element> | <array_element>
<single_element> ::= IDENTIFIER
<array_element> ::= IDENTIFIER '[' INTEGER ']'
<final_node> ::= <element> | <attribute>
<attribute> ::= '@' IDENTIFIER
```

Here are the examples of all the possible Paths:

```
/node1/node2
/node1/node2[2]/
/node1/node2[2]/node3
/node1/@attribute
```

For instance, BuyerPartyDetails/BuyerPostalAddressDetails/BuyerTownName is a Path. After it is split by '/', there are three nodes: BuyerPartyDetails, BuyerPostalAddressDetails and BuyerTownName. The last node, BuyerTownName, is a child element of BuyerPostalAddressDetails which is a child element of BuyerPartyDetails.

For each Path, it will be split into several nodes and stored into an array of nodes. Each node may contain '@' or '[]', which is invalid for being a name of an element or an attribute. Therefore, before creating and locating them in the right position, the actual name has to be got first.

The private static method getNamePart takes a string as a parameter and returns a string. The compiler checks the parameter, a node, and returns the name part. There are three cases:

- The first character is ‘@’, returns the rest of the node
- The node contains ‘[‘, returns the first character until the ‘[‘
- The node does not contain ‘@’ or ‘[‘, returns the node

For example:

```
getNamePart("InvoiceTypeCode"); return "InvoiceTypeCode";
getNamePart("@CodeListAgencyIdentifier"); return "CodeListAgencyIdentifier";
getNamePart("SellerOrganisationName[1]"); return "SellerOrganisationName";
```

In the case that the node is an element in an array, the program must create enough elements, which means it has to create the element as many times as the integer between ‘[‘ and ‘]’. Hence, knowing the number inside the square brackets is important.

The private static method getInt takes a node string as a parameter and returns an integer value. The compiler checks if the node contains a ‘[’. If it does, then it checks whether there is an integer after it or not. When there is a string of integers in the square brackets, the function converts the string into an integer value and returns it to the program. But if the node does not contain a ‘[‘, or there is not an integer inside the square brackets, then the function returns -1 to the program.

For example:

```
getInt("InvoiceTypeCode"); return -1;
getInt("InvoiceRow[1]"); return 1;
getInt("InvoiceRow[a]"); return -1;
```

There is also a private static method called isAttribute. It checks whether the node is an attribute or not. It takes a node string as a parameter, then checks if the first character of the node is a ‘@’. If a ‘@’ is the first character, it means this node is an attribute, and the function will return a Boolean value “true” to the program. Otherwise it returns a “false”.

For example:

```
isAttribute("InvoiceTypeCode"); return false;
isAttribute("@Format");  return true;
```

After dealing with nodes, they are ready to be put into the document. A node can be an element, one of the element array, or an attribute. Therefore, parsing the node is just to check whether the node contains one of those notations or not. Depending on which kind of node it is, the program will process it differently, as the flow chart shows:

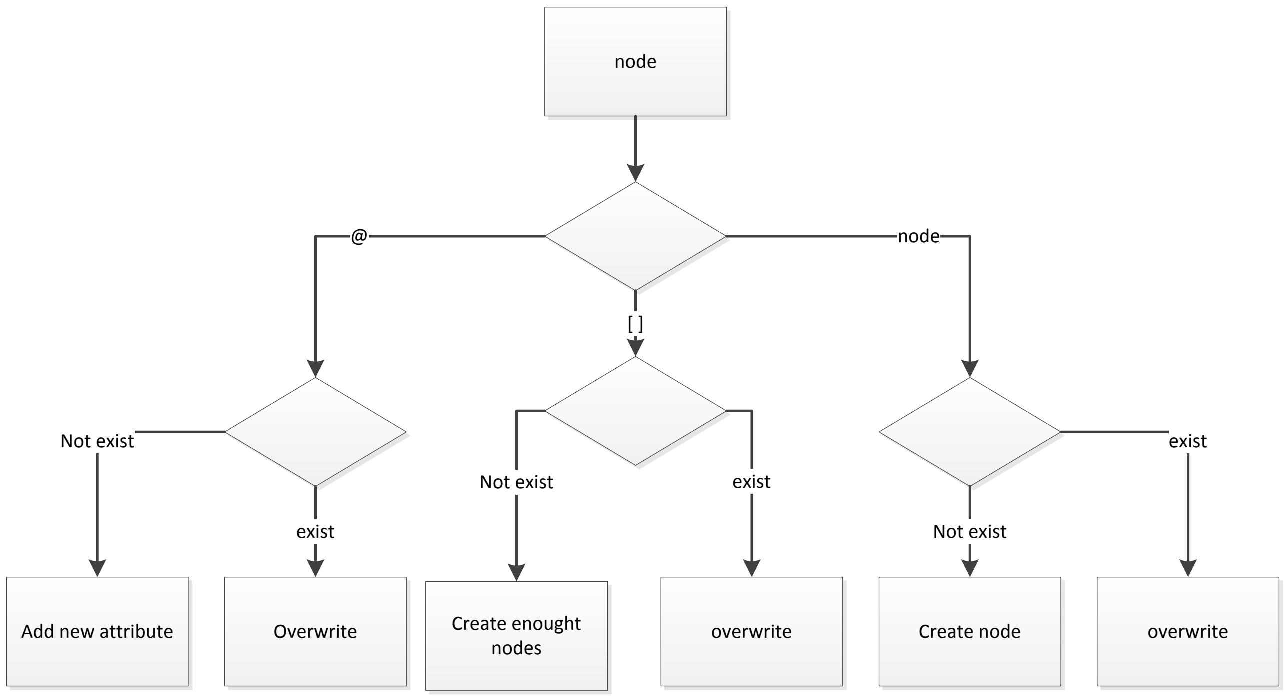


FIGURE 8 Flow chart of node

The public method putElementAttribute takes the Path and the Value as parameters, for each node in the array of nodes is split from the Path. Firstly, the program calls the method getNamePart, and then stores the name to a string called "name". Secondly, the program calls the function getInt, and stores the value to an integer named "count". Then the compiler will check which case the node is in.

- The program calls method isAttribute first to check if the node is an attribute. If it is true, then it checks if the parent has this attribute. If it does, then it overwrites the attribute with the value. If it does not have this attributes yet, then the parent element adds an attribute with the "name", and assigns the value to the attribute.
- The program checks the value of "count", if it is bigger than 0, which means there is an array of this element. Then the program counts how many same elements there are in the parent element are. If there are fewer elements than the value of "count", then the parent element

creates this element as many times as it is needed, and it assigns the value to this element. If there are enough elements, then it just overwrites the value to it.

- If the method isAttribute returns a Boolean value "false" and the method getInt returns "count", the value of which is negative, it means this node is an element. The program checks if the parent element has this child element or not. If there is one, then it just overwrites the value to it. If there is not any, then the parent element adds this element as a child element and assigns the value to it.

4.7 Validate Finvoice Document

After you put all the elements and attributes that are needed into the XDocument, mDocument, it has to be validated against the Finvoice schema before saving it. The public method validateXml does the entire job.

This method is used to validate the Finvoice document created by previous processes against the Finvoice schema. It takes two parameters: a string and a TextWriter. The string is the name of the schema used to validate the document. And the TextWriter represents a writer that can write a sequential series of characters. This class is abstract. (MSDN, date of retrieval 9.4.2013a) It is used to write the error message to the console window.

The program firstly loads the schema file, stores it into a File Steam inheriting from IO stream. The File Stream is read into an instance of XmlSchema class, which represents an XML schema, and is added into an XmlSchemaSet object. The XmlSchemaSet is a parameter for an extension method Validate, in System.Xml.Schema namespace. This method validates that an XDocument conforms to an XSD in an XmlSchemaSet (MSDN, date of retrieval 9.5.2013). The syntax is:

```
public static void Validate(
    this XDocument source,
    XmlSchemaSet schemas,
    ValidationEventHandler validationEventHandler
)
```

The following example, from Microsoft Developer Network (MSDN), creates an XmlSchemaSet, and then validates two XDocument objects against the schema set. One of the documents is valid, the other is not. (MSDN, date of retrieval 9.5.2013)

```
string xsdMarkup =
    @"<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
       <xsd:element name='Root'>
        <xsd:complexType>
         <xsd:sequence>
          <xsd:element name='Child1' minOccurs='1' maxOccurs='1'/>
          <xsd:element name='Child2' minOccurs='1' maxOccurs='1'/>
         </xsd:sequence>
        </xsd:complexType>
       </xsd:element>
      </xsd:schema>";
XmlSchemaSet schemas = new XmlSchemaSet();
schemas.Add("", XmlReader.Create(new StringReader(xsdMarkup)));

XDocument doc1 = new XDocument(
    new XElement("Root",
        new XElement("Child1", "content1"),
        new XElement("Child2", "content1")
    )
);

XDocument doc2 = new XDocument(
    new XElement("Root",
        new XElement("Child1", "content1"),
        new XElement("Child3", "content1")
    )
);

Console.WriteLine("Validating doc1");
bool errors = false;
doc1.Validate(schemas, (o, e) =>
                     {
                         Console.WriteLine("{0}", e.Message);
                         errors = true;
                     });
```

```
Console.WriteLine("doc1 {0}", errors ? "did not validate" : "validated");

Console.WriteLine();
Console.WriteLine("Validating doc2");
errors = false;
doc2.Validate(schemas, (o, e) =>
                     {
                         Console.WriteLine("{0}", e.Message);
                         errors = true;
                     });
Console.WriteLine("doc2 {0}", errors ? "did not validate" : "validated");
```

This example produces the following output:

```
Validating doc1
doc1 validated

Validating doc2
The element 'Root' has invalid child element 'Child3'. List of possible
elements expected: 'Child2'.
doc2 did not validate
```

4.8 Save Finvoice Document

The last step is to save the Finvoice document, mDocument, into a file. Since mDocument is an instance of XDocument, it can call the XDocument.Save Method (Stream). The syntax is:

```
public void Save(
    Stream stream
)
```

For example:

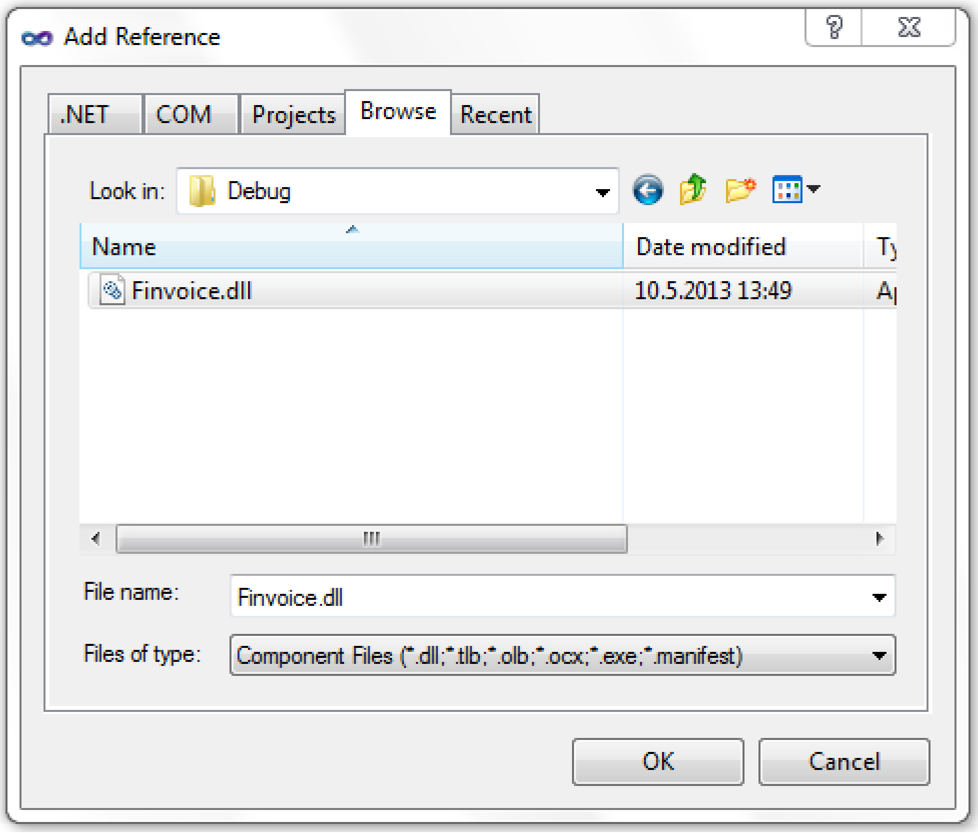
```
mDocument.Save("Finvoice.xml");
```

The result of this example is that the mDocument is saved into a file called Finvoice.xml.

4.9 Build reference

After creating a class library project and writing a piece of code there, the next important step is to compile it and add it to the reference so that it can be used by other programs.

To build a DLL you can just press Ctrl+Shift+B buttons from the keyboard, or you can click on Build Solution icon from the menu bar. Next step is to add it to the reference. On the right side of the Visual Studio window, right click on any empty area in the Solution explorer, and a window pops up and shows an option: Add reference. By selecting this option, the Add reference dialog shows:



Look in: Debug

Name	Date modified	Ty
Finvoice.dll	10.5.2013 13:49	A

File name: Finvoice.dll

Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)

FIGURE 9 Screenshot of the Add Reference dialog

Click on Browse section, select the DLL from where it is located and then click on OK button. Now the DLL is ready to be used by other applications.

5 TESTING

When the dynamic link library is ready we have to create an application to test it. This chapter explains how the testing application was created, how the DLL was used and what the testing result is.

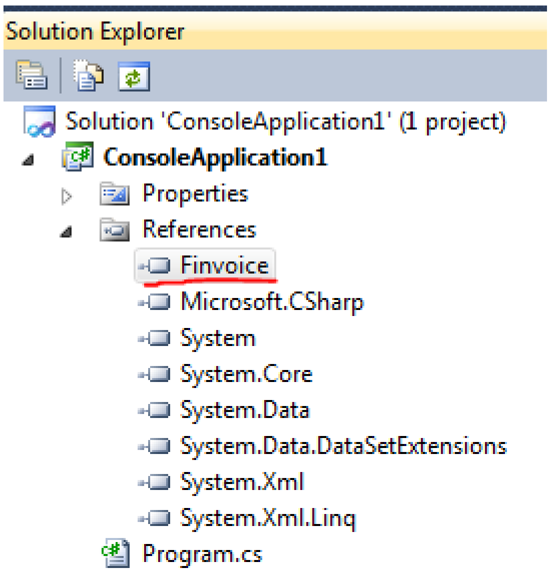
5.1 Test Plan

The Finvioce.dll has to be able to do the following tasks:

- Be able to be referenced by other applications.
- Generate valid Finvoice XML documents.
- Transform the output according to Finvoice.xsl.

The testing application was made with the C# programming language and implemented using Visual Studio 2010. There are a few steps to create a new application and add the reference to the DLL using Visual Studio 2010:

1. Click on the New Project option on the left side on the Start page. Or select New and then Project from the File menu.
2. On the left side of the New Project pane, select Windows under the Visual C#, and then select Console Application.
3. Give a name to the application and browse a location for it, and click on OK button.
4. Right click on the Reference section in the Solution Explorer and choose the Add Reference option.
5. On the Add Reference dialogue, select Browse section and then select the Finvoice.dll from where it is located.
6. After clicking on OK button, the Finvoice.dll appears in the Solution Explorer, as the following FIGURE 10 shows:


Solution Explorer

```
Solution 'ConsoleApplication1' (1 project)
  ConsoleApplication1
    Properties
    References
      Finvoice
      Microsoft.CSharp
      System
      System.Core
      System.Data
      System.Data.DataSetExtensions
      System.Xml
      System.Xml.Linq
    Program.cs
```

FIGURE 10. Screenshot of the Solution Explorer

7. The last step is to add the reference in the Code Editor like this:

```
using Finvoice;
```

The keyword using in C# could be a directive or a statement. In this case, it is used to import the types defined in the Finvoice namespace. The using directive allows you to use unqualified class name to reference the DLL classes or methods at compile time (MSDN, date of retrieval 2.5.2013), see following examples:

```
using Finvoice;
FinvoiceDocument doc = new FinvoiceDocument();
```

Otherwise, you have to use the fully qualified name like this:

```
Finvoiec.FinvoiceDocument doc = new Finvoice.FinvoiceDocument();
```

5.2 Test

After a new application has been created and the reference has been added successfully to Finvoice.dll, the application can use the DLL to generate a Finvoice document with the following steps:

1. Create an instance of FinvoiceDocument class with the new operator.

For example: `FinvoiceDocument doc = new FinvoiceDocument();`

The FinvoiceDocument constructor is invoked by the new operator and then the object doc is instantiated.

2. Call methods from Finvoice,dll using the dot operator (.).

   For example: `doc.putElementAttribute("DeliveryDetails/DeliveryDate", "20120808");`

   The object doc can access any public methods from the same class, FinvoiceDocument, with the dot operator. This function parses the parameters and locates the value in the right node in the document. And this function will be called as many times as it is needed depending on input.

3. Validate doc.

   For example: `doc.validateXml("Finvoice.xsd", Console.Out);`

   After putting all the elements and attributes have been put to the doc whose type is FinvoiceDocument, the doc will have to be validated against Finvoice.xsd before it will be saved. The second parameter will get the standard output stream which displays the error message on the console dialogue window.

4. Save doc into an .xml file.

   For example: `doc.saveFinvoiceDocument();`

   When the doc has been validated, it will be saved into the Finvoice.xml file placed in the project folder.

5. Build the executable and run it. Press Ctrl+Shift+B buttons or by click on Build Solution icon from the menu bar. Then press the F5 button or click on the debug icon from the menu bar to run the application.

In order to produce a valid Finvoice XML document, a minimum set of elements have to be added. According to the Finvoice Implementation Guidelines Version 2.0, the following table of elements and attributes are required:

Level	Name	Occurs	Length
1	SellerPartyDetails	1	
2	SellerOrganisationName	1..n	2..70

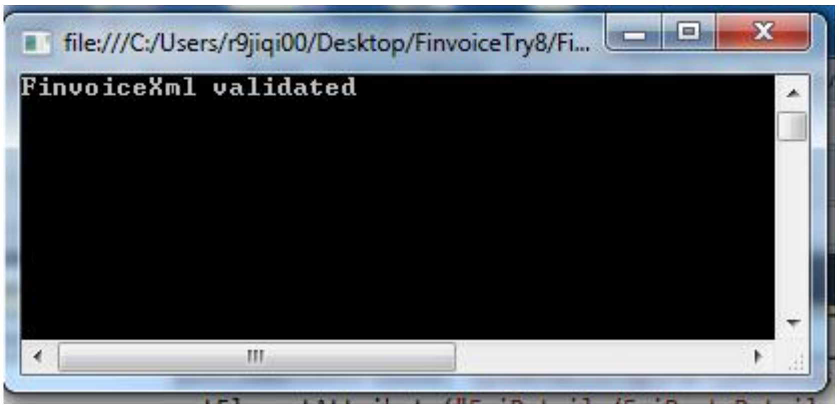
1	BuyerPartyDetails	1	
2	BuyerOrganisationName	1..n	2..70
1	InvoiceDetails	1	
2	InvoiceTypeCode	1	5
	Attribute: CodeListAgencyIdentifier	0..1	
2	InvoiceTypeText	1	1..35
2	OriginCode	1	
2	InvoiceNumber	1	1..20
2	InvoiceDate	1	8
	Attribute: Format	1	
2	InvoiceTotalVatIncludedAmount	1	1..22
	Attribute: AmountCurrencyIdentifier	1	3
1	InvoiceRow	1..n	
1	EpiDetails	1	
2	EpiIdentificationDetails	1	
3	EpiDate	1	8
	Attribute: Format	1	
3	EpiReference	1	0..35
2	EpiPartyDetails	1	
3	EpiBfiPartyDetails	1	
3	EpiBeneficiaryPartyDetails	1	
4	EpiAccountID	1	1..34
	Attribute: IdentificationSchemeName	1	
2	EpiPaymentInstructionDetails	1	
3	EpiInstructedAmount	1	4..19
	Attribute: AmountCurrencyIdentifier	1	3
3	EpiCharge	1	0
	Attribute: ChargeOption	1	
3	EpiDateOptionDate	1	8
	Attribute: Format	1	

TABLE 2. The minimum set of mandatory elements and attributes

In a real case, a Finvoice document can be very detailed. It contains much more data than just those mandatory elements. But it only requires calling the putElementAttribute method more times to build the complicated Finvoice documents, see testing example (Appendix 5) and it produces a Finvoice document as Appendix 6.

5.3 Testing result

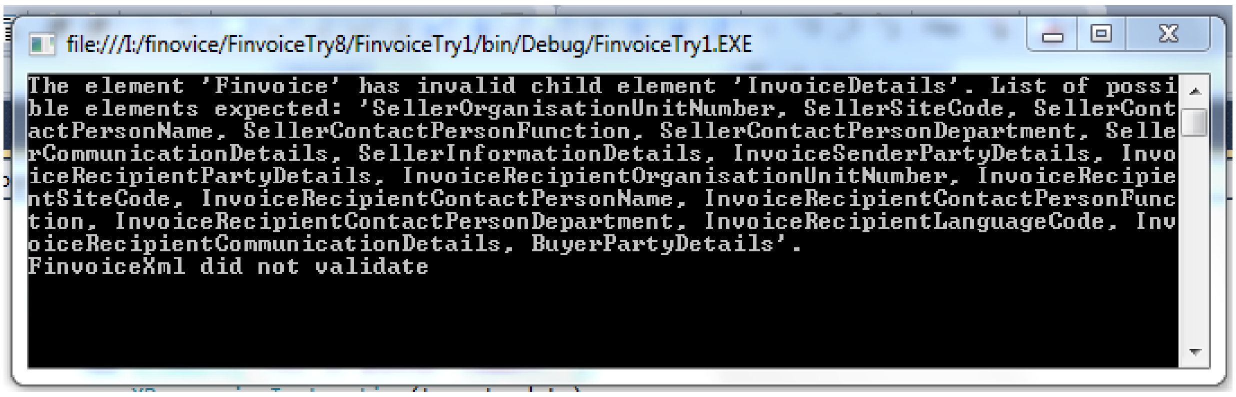
Here is the result of a testing application (see appendix 3) that references the Finvoice.dll (see appendix 2) file with the mandatory elements and attributes only. When those mandatory elements and attributes have been added correctly, it produces a Finvoice.xml file (see appendix 4). And the console dialogue shows as the FIGURE11:



```
FinvoiceXml validated
```

FIGURE11. Screenshot of console window1

But if any of those mandatory elements are missing, the console dialogue shows an error message. For example, when the BuyerPartyDetails element is missing, then the program thinks that the rest of elements are invalid child elements. And the error message is as shown below:

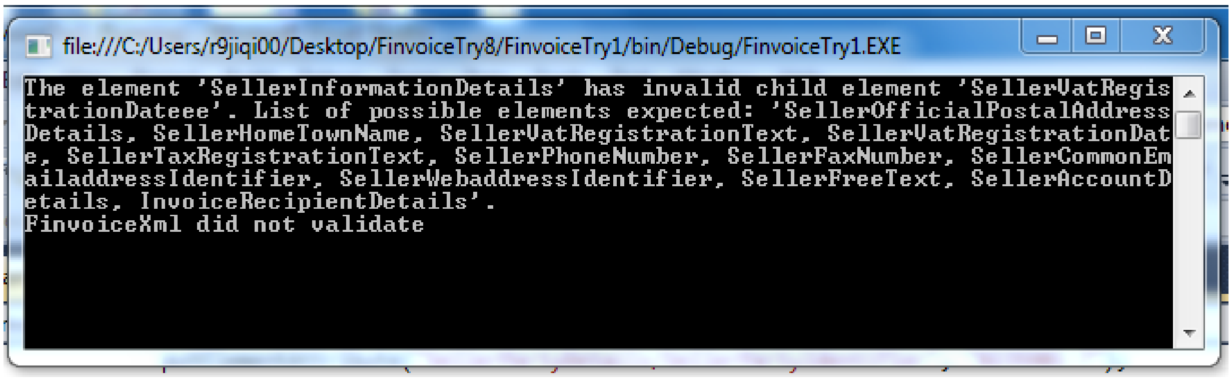


file:///I:/finovice/FinvoiceTry8/FinvoiceTry1/bin/Debug/FinvoiceTry1.EXE

```
The element 'Finvoice' has invalid child element 'InvoiceDetails'. List of possi
ble elements expected: 'SellerOrganisationUnitNumber, SellerSiteCode, SellerCont
actPersonName, SellerContactPersonFunction, SellerContactPersonDepartment, Selle
rCommunicationDetails, SellerInformationDetails, InvoiceSenderPartyDetails, Invo
iceRecipientPartyDetails, InvoiceRecipientOrganisationUnitNumber, InvoiceRecipie
ntSiteCode, InvoiceRecipientContactPersonName, InvoiceRecipientContactPersonFunc
tion, InvoiceRecipientContactPersonDepartment, InvoiceRecipientLanguageCode, Inv
oiceRecipientCommunicationDetails, BuyerPartyDetails'.
FinvoiceXml did not validate
```

FIGURE 12. Screenshot of console window2

When the path of an element is incorrect, or the names are misspelled, then the program knows that there is an invalid element, and the console dialogue is as shown below:

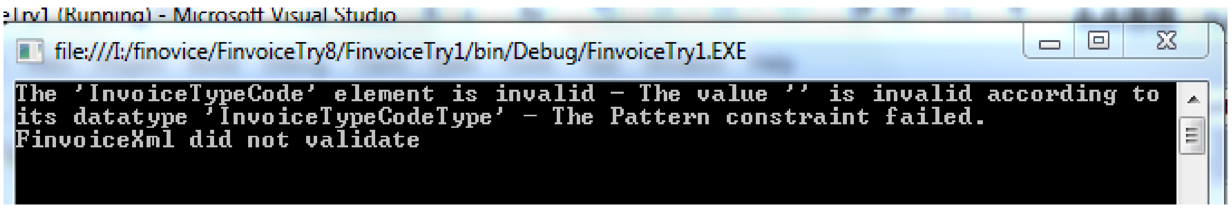


file:///C:/Users/r9jiqi00/Desktop/FinvoiceTry8/FinvoiceTry1/bin/Debug/FinvoiceTry1.EXE

```
The element 'SellerInformationDetails' has invalid child element 'SellerVatRegis
trationDateee'. List of possible elements expected: 'SellerOfficialPostalAddress
Details, SellerHomeTownName, SellerVatRegistrationText, SellerVatRegistrationDat
e, SellerTaxRegistrationText, SellerPhoneNumber, SellerFaxNumber, SellerCommonEm
ailaddressIdentifier, SellerWebaddressIdentifier, SellerFreeText, SellerAccountD
etails, InvoiceRecipientDetails'.
FinvoiceXml did not validate
```

FIGURE 13. Screenshot of console window3

When only the path of an element is given and the value is an empty string, then the error message is as shown below:



file:///I:/finovice/FinvoiceTry8/FinvoiceTry1/bin/Debug/FinvoiceTry1.EXE

```
The 'InvoiceTypeCode' element is invalid - The value '' is invalid according to
its datatype 'InvoiceTypeCodeType' - The Pattern constraint failed.
FinvoiceXml did not validate
```

FIGURE 14 Screenshot of console window4

In all error situations, there will not be a Finvoice,xml file saved. If there are no errors, the valid Finvoice XML documents can be displayed as a regular paper invoice by Internet Explore when the Finvoice.xsl file is placed in the same folder (see appendix 7). In conclusion, the Finvoice.dll has passed all the testing tasks.

6 POSSIBILITIES OF FURTHER DEVELOPMENT

The Finvoice.dll reached all of the requirements. And of course it could have been done better, such as by warping some frequently used methods in a class to make the code more elegant. There are some different ways to implement the Finvoice.dll.

For example, use the regular expression to design patters for three kinds of Path parameters: one for a single element, one for an attribute and one for an array element, for example:

```
string elementPattern = @"\W+";
string attributePattern = @"\@.\W+";
string arrayPattern = @"\W+.\[.\d+.\]";
```

Instead of calling different functions to determine the path, using the IsMatch method of System.Text.RegularExpressions.Regex in .NET Framework 4.5 to indicate whether the specified regular expression finds a match in the specified input string provides a flexible and efficient way for parsing the path (MSDN, date of retrieval 11.5.2013a). Here is the syntax for IsMatch method in C#:

```
public static bool IsMatch(
    string input,
    string pattern
)
```

For example:

```
using System;
using System.Text.RegularExpressions;

public class Example
{
   public static void Main()
   {
      string[] nodes= { "SellerAccountInfo", "@AccountID", "SellerName[2]" };
      string elementPattern = @"\W+";

      foreach (string node in nodes)
         Console.WriteLine("{0} {1} an element.",
                                   node,
                                   Regex.IsMatch(node, elementPattern) ? "is" : "is
not");
   }
}
```

The example displays the following output:

```
SellerAccountInfo is an element.
@AccountID is not an element.
SellerName[2] is not an element.
```

7 CONCLUSION

This thesis work was proposed by Jukka Penttilä from Sunwell Trade. The aim was that to make a DLL, which REX can call with the API and to generate the Finvoice documents.

The Finvoice.dll works as was required by the client. Although it did not require too much code to accomplish this thesis work, I spent a lot of time on reaching and trying different approaches to implement the DLL. And I have gained new knowledge and skills for sure, such as the C# programming language and XML. I knew them earlier but I had not really worked with them before in such a detail. And after the thesis work I understand them better and I am able to implement them in new ways. Moreover, I understand more about software designing and project planning. All of them are really useful for my future career.

Besides the technical knowledge and skills, I have also gained the capability of researching useful information quickly. In addition, I learned communication skills with customers, especially when the customer and I did not share a strong common language.

LIST OF REFERENCES

Federation of Finnish Financial Services. 2010. Environmentally friendly electronic invoice. Helsinki: Federation of Finnish Financial Services. Date of retrieval 11.4.2013 http://www.fkl.fi/en/material/publications/Publications/Environmentally_friendly_electronic_invoice.pdf.

Federation of Finnish Financial Services. 2012. Finvoice Implementation Guideline, Version 2.0. Helsinki: Federation of Finnish Financial Services. Date of retrieval: 9.3.2013 http://www.fkl.fi/verkkolasku/yrityksen_verkkolasku/ladattavat/Tekniset%20tiedostot/Finvoice_2_0_implementation_guidelines.pdf.

Griffiths, I., Adams, M., Liberty, J. 2010, Programming C# 4.0, Sixth Edition. Sebastobol, CA: O'Reilly Media, Inc.

GXS. A brief history. Date of retrieval 7.4.2013a http://www.einvoicingbasics.co.uk/what-is-e-invoicing/a-brief-history/.

GXS. eInvoice formats. Date of retrieval 7.4.2013b http://www.einvoicingbasics.co.uk/what-is-e-invoicing/einvoice-formats/.

Huhtanen J., 2003. Verkkopankkikoodien käyttö yleistyy sähköisessä tunnistuksessa. Tietokone 3/2003, 13.Date of retrieval 19.4.2013 http://www.tietokone.fi/lehti/tietokone_3_2003/ajankohtaista_3737.

MSDN. DLLs in Visual C++. Date of retrieval 26.3.2013a http://msdn.microsoft.com/en-us/library/1ez7dh12.aspx.

MSDN. Extensions.Validate Method (XDocument, XmlSchemaSet, ValidationEventHandler). Date of retrieval 9.5.2013 http://msdn.microsoft.com/fi-fi/library/bb340331.aspx.

MSDN. How to: Create a Document with Namespaces (C#) (LINQ to XML). Date of retrieval 23.4.2013a
http://msdn.microsoft.com/en-us/library/bb387075.aspx.

MSDN. Regex.IsMatch Method (String, String). Date of retrieval 11.5.2013a
http://msdn.microsoft.com/en-us/library/sdx2bds0.aspx.

MSDN. System.Xml Namespace. Date of retrieval 11.3.2013
http://msdn.microsoft.com/en-us/library/system.xml.aspx.

MSDN. System.Xml.Linq Namespace. Date of retrieval 26.3.2013b
http://msdn.microsoft.com/en-us/library/system.xml.linq.aspx.

MSDN. TextWriter Class. Date of retrieval 9.4.2013a
http://msdn.microsoft.com/en-us/library/system.io.textwriter.aspx.

MSDN. using Directive (C# Reference). Date of retrieval 2.5.2013
http://msdn.microsoft.com/en-us/library/vstudio/sf0df423.aspx.

MSDN. Visual C#. Date of retrieval 19.2.2013
http://msdn.microsoft.com/en-us/library/vstudio/kx37x362.aspx.

MSDN. XDeclaration Constructor (String, String, String). Date of retrieval 24.4.2013a
http://msdn.microsoft.com/en-us/library/bb359725.aspx.

MSDN. XmlDocument Class. Date of retrieval 9.4.2013b
http://msdn.microsoft.com/en-us/library/system.xml.xmldocument.aspx.

MSDN. XML Technology Backgrounder. Date of retrieval 11.5.2013b
http://msdn.microsoft.com/en-us/library/8ktfywf4(v=vs.71).aspx.

MSDN. XNamespace Class. Date of retrieval 24.4.2013b
http://msdn.microsoft.com/en-us/library/system.xml.linq.xnamespace.aspx.

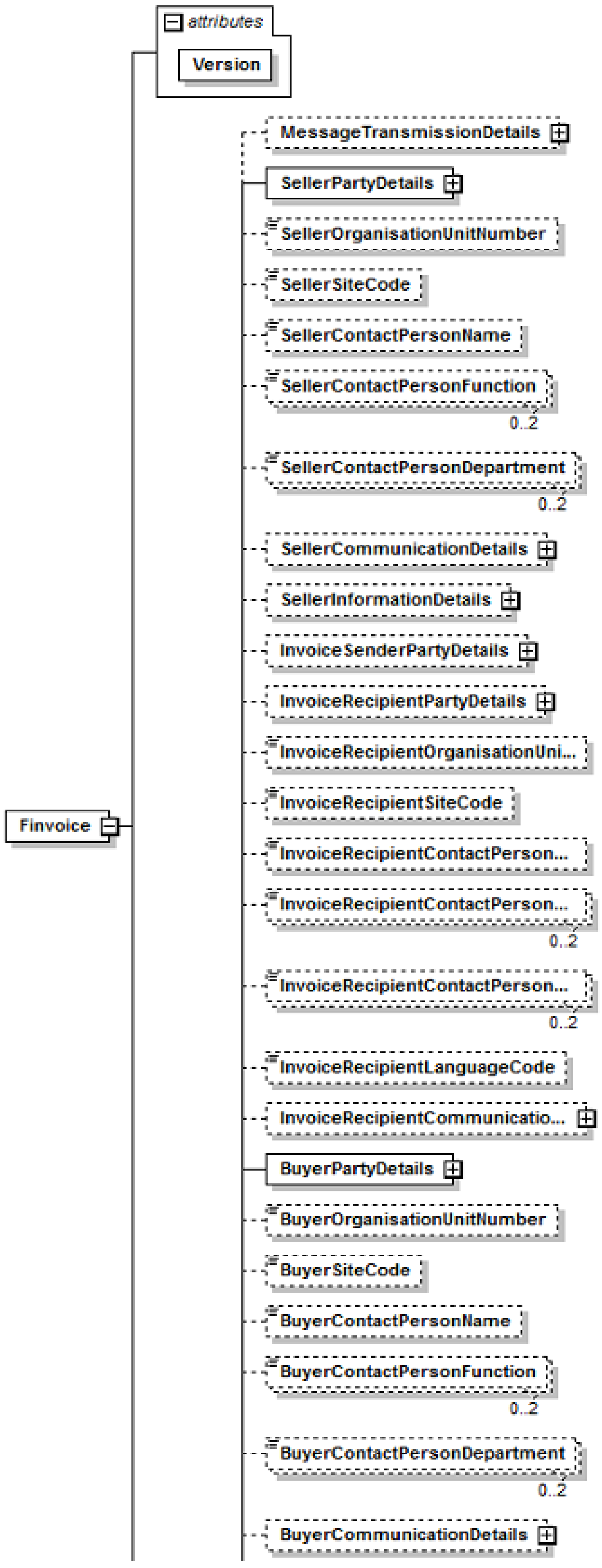
MSDN. XPath Syntax. Date of retrieval 15.4.2013
http://msdn.microsoft.com/en-us/library/ms256471.aspx.

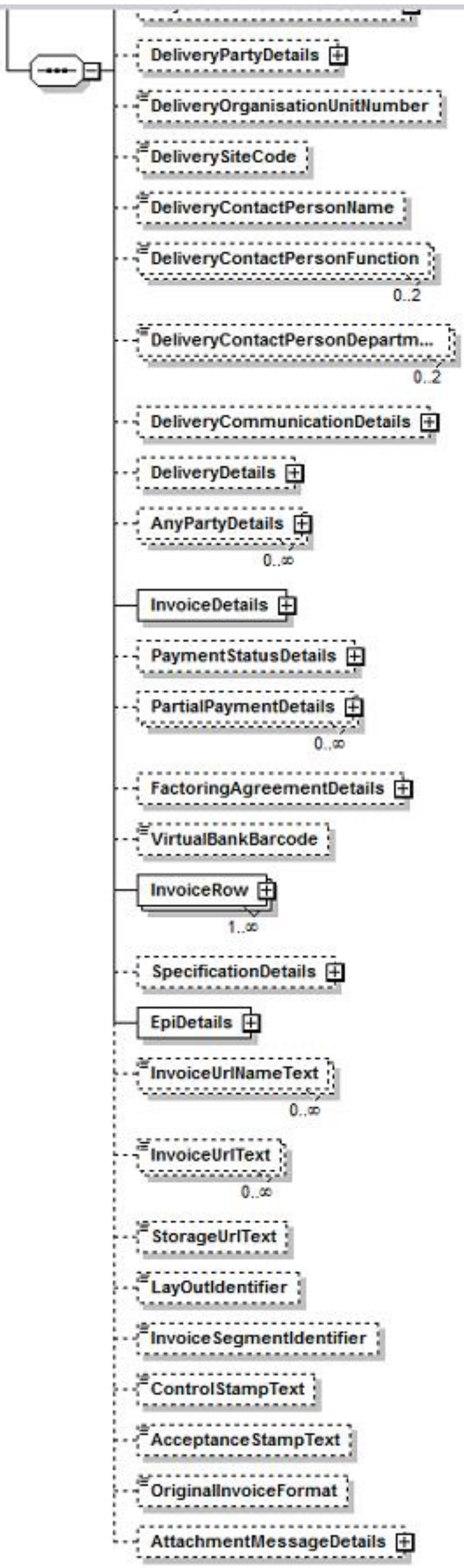
MSDN. XProcessingInstruction Constructor (String, String). Date of retrieval 23.4.2013b
http://msdn.microsoft.com/en-us/library/bb359045.aspx.

Wikipedia. Unified Modeling Language. Date of retrieval 25.3.2013
http://en.wikipedia.org/wiki/Unified_Modeling_Language.

APPENDICES

Appendix 1 Structure of Finvoice (Finvoice Implementation Guideline, Version 2.0)


attributes
Version
MessageTransmissionDetails
SellerPartyDetails
SellerOrganisationUnitNumber
SellerSiteCode
SellerContactPersonName
SellerContactPersonFunction
0..2
SellerContactPersonDepartment
0..2
SellerCommunicationDetails
SellerInformationDetails
InvoiceSenderPartyDetails
InvoiceRecipientPartyDetails
InvoiceRecipientOrganisationUni...
InvoiceRecipientSiteCode
Finvoice
InvoiceRecipientContactPerson...
InvoiceRecipientContactPerson...
0..2
InvoiceRecipientContactPerson...
0..2
InvoiceRecipientLanguageCode
InvoiceRecipientCommunicatio...
BuyerPartyDetails
BuyerOrganisationUnitNumber
BuyerSiteCode
BuyerContactPersonName
BuyerContactPersonFunction
0..2
BuyerContactPersonDepartment
0..2
BuyerCommunicationDetails


DeliveryPartyDetails
DeliveryOrganisationUnitNumber
DeliverySiteCode
DeliveryContactPersonName
DeliveryContactPersonFunction
0..2
DeliveryContactPersonDepartm...
0..2
DeliveryCommunicationDetails
DeliveryDetails
AnyPartyDetails
0..∞
InvoiceDetails
PaymentStatusDetails
PartialPaymentDetails
0..∞
FactoringAgreementDetails
VirtualBankBarcode
InvoiceRow
1..∞
SpecificationDetails
EpiDetails
InvoiceUrlNameText
0..∞
InvoiceUrlText
0..∞
StorageUrlText
LayOutIdentifier
InvoiceSegmentIdentifier
ControlStampText
AcceptanceStampText
OriginalInvoiceFormat
AttachmentMessageDetails

Appendix 2

```
//Finvoice.dll
//FinvoiceDocument.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml.Linq;
using System.Xml.Schema;
using System.IO;

namespace FinvoiceDocument
{
    public class FinvoiceDocument
    {
        private XDocument mDocument;

        public FinvoiceDocument(){
            mDocument = CreateDocument();
            mDocument.Add(CreatRootElement());
        }

        private static XDocument CreateDocument()
        {
            string target = "xml-stylesheet";
            string data = "type=\"text/xsl\" href=\"Finvoice.xsl\"";
            return new XDocument(
                new XDeclaration("1.0", "ISO-8859-15", null),
                new XProcessingInstruction(target, data));
        }

        private static XElement CreatRootElement()
        {
             XNamespace xsiNs = XNamespace.Get("http://www.w3.org/2001/XMLSchema-
             instance");

            XNamespace defaultNamespace = XNamespace.Get("Finvoice.xsd");

            return new XElement("Finvoice",
                    new XAttribute("Version", "2.0"),
                    new XAttribute(XNamespace.Xmlns + "xsi", xsiNs),
                    new XAttribute(xsiNs + "noNamespaceSchemaLocation",
defaultNamespace));
        }

        public void putElementAttribute( string path, string value)
        {
            string[] nodes = path.Split('/');
            XElement parent = mDocument.Element("Finvoice") ;

            foreach (string node in nodes)
            {
                string name = getNamePart(node);
                int count = getInt(node);
                if (isAttribute(node))
                {
                    if (parent.Attribute(name) == null)
                    {
                        parent.Add(new XAttribute(name, value));
                    }
                    parent.Attribute(name).Value = value;
                    return;
                }
                else if (count > 0)
                {

                    for (int i = 0; i < count; i++)
                    {
                        if (parent.Elements(name).Count() < count)
                        {
                            parent.Add(new XElement(name));
                        }
                    }
                    parent = parent.Elements(name).ElementAt(count - 1);
                }
                else
                {
                    if (parent.Element(name) == null)
                    {
                        parent.Add(new XElement(name));
                    }
                    parent = parent.Element(name);
                }
            }
            parent.Value = value;
        }

        public bool validateXml(string schema, TextWriter errorWriter)
        {
            bool errors = false;
            XmlSchema finvoiceSchema = XmlSchema.Read(new FileStream(schema,
FileMode.Open), null);
            XmlSchemaSet finvoiceSchemaSet = new XmlSchemaSet();
            finvoiceSchemaSet.Add(finvoiceSchema);

            if (errorWriter != null)
            {
                mDocument.Validate(finvoiceSchemaSet, (o, e) =>
                {
                    errorWriter.WriteLine("{0}", e.Message);
                    errors = true;
                });
                errorWriter.WriteLine("FinvoiceXml {0}", errors ? "did not
validate": "validated");
            }
            else {
                mDocument.Validate(finvoiceSchemaSet, (o, e) =>
                {
                    errors = true;
                });
            }
            return !errors;
        }
        public void saveFinvoiceDocument()
        {
            mDocument.Save("Finvoice.xml");
        }



        private static string getNamePart(string node)
        {
            if (node.ElementAt(0) == '@')
            {
                return node.Substring(1);
            }
            else if (node.Contains("["))
            {
                return node.Substring(0, node.IndexOf('['));
            }
            else
            {
                return node;
            }
        }

        private static bool isAttribute(string node)
        {
            if (node.Contains("@"))
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        private static int getInt(string node)
        {
            if (node.Contains("["))
            {
                int length = node.IndexOf(']') - node.IndexOf('[') - 1;
                string sub = node.Substring(node.IndexOf('[') + 1, length);
                int number;
                bool isArray = Int32.TryParse(sub, out number);
                if (isArray)
                {
                    return Convert.ToInt32(sub);
                }
                else
                {
                    return -1;
                }
            }
            else
            {
                return -1;
            }
        }
    }
}
```

Appendix 3

```
//testing.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Finvoice;

namespace Test3
{
    class Program
    {
        static void Main(string[] args)
        {
            FinvoiceDocument doc = new FinvoiceDocument();

            doc.putElementAttribute("SellerPartyDetails/SellerOrganisationName",
            "Ramko ry");

            doc.putElementAttribute("BuyerPartyDetails/BuyerOrganisationName",
            "Sensorit Oy");

            doc.putElementAttribute("InvoiceDetails/InvoiceTypeCode", "INV01");

            doc.putElementAttribute("InvoiceDetails/InvoiceTypeCode/@CodeListAge
            ncyIdentifier", "SPY");
            doc.putElementAttribute("InvoiceDetails/InvoiceTypeText", "Lasku");

            doc.putElementAttribute("InvoiceDetails/OriginCode", "Original");

            doc.putElementAttribute("InvoiceDetails/InvoiceNumber", "12345");

            doc.putElementAttribute("InvoiceDetails/InvoiceDate", "20120808");

            doc.putElementAttribute("InvoiceDetails/InvoiceDate/@Format",
"CCYYMMDD");

            doc.putElementAttribute("InvoiceDetails/InvoiceTotalVatIncludedAmoun
            t", "10,98");

            doc.putElementAttribute("InvoiceDetails/InvoiceTotalVatIncludedAmoun
            t/@AmountCurrencyIdentifier", "EUR");

            doc.putElementAttribute("InvoiceRow", "");

            doc.putElementAttribute("EpiDetails/EpiIdentificationDetails/EpiDate
            ", "20120808");

            doc.putElementAttribute("EpiDetails/EpiIdentificationDetails/EpiDate
            /@Format", "CCYYMMDD");

            doc.putElementAttribute("EpiDetails/EpiIdentificationDetails/EpiRefe
            rence", "2004468");

            doc.putElementAttribute("EpiDetails/EpiPartyDetails/EpiBfiPartyDetai
            ls", "");
            doc.putElementAttribute("EpiDetails/EpiPartyDetails/EpiBeneficiaryPa
            rtyDetails/EpiAccountID", "FI7036363001126978");

            doc.putElementAttribute("EpiDetails/EpiPartyDetails/EpiBeneficiaryPa
            rtyDetails/EpiAccountID/@IdentificationSchemeName", "IBAN");

            doc.putElementAttribute("EpiDetails/EpiPaymentInstructionDetails/Epi
            InstructedAmount", "10,98");

            doc.putElementAttribute("EpiDetails/EpiPaymentInstructionDetails/Epi
            InstructedAmount/@AmountCurrencyIdentifier", "EUR");

            doc.putElementAttribute("EpiDetails/EpiPaymentInstructionDetails/Epi
            Charge", "SLEV");

            doc.putElementAttribute("EpiDetails/EpiPaymentInstructionDetails/Epi
            Charge/@ChargeOption", "SLEV");

            doc.putElementAttribute("EpiDetails/EpiPaymentInstructionDetails/Epi
            DateOptionDate", "20121122");
            doc.putElementAttribute("EpiDetails/EpiPaymentInstructionDetails/Epi
            DateOptionDate/@Format", "CCYYMMDD");

            bool valid = doc.validateXml("Finvoice.xsd", Console.Out );
            if (valid) {
                doc.saveFinvoiceDocument();
            }
            Console.Read();
        }

    }
}
```

Appendix 4

```xml
<!--This is Finvice.xml with minimum mandatory data produced by testing.cs-->
<?xml version="1.0" encoding="iso-8859-15"?>
<?xml-stylesheet type="text/xsl" href="Finvoice.xsl'?>
<Finvoice Version="2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Finvoice.xsd">
  <SellerPartyDetails>
    <SellerOrganisationName>Ramko ry</SellerOrganisationName>
  </SellerPartyDetails>
  <BuyerPartyDetails>
    <BuyerOrganisationName>Sensorit Oy</BuyerOrganisationName>
  </BuyerPartyDetails>
  <InvoiceDetails>
    <InvoiceTypeCode CodeListAgencyIdentifier="SPY">INV01</InvoiceTypeCode>
    <InvoiceTypeText>Lasku</InvoiceTypeText>
    <OriginCode>Original</OriginCode>
    <InvoiceNumber>12345</InvoiceNumber>
    <InvoiceDate Format="CCYYMMDD">20120808</InvoiceDate>
    <InvoiceTotalVatIncludedAmount
AmountCurrencyIdentifier="EUR">10,98</InvoiceTotalVatIncludedAmount>
  </InvoiceDetails>
  <InvoiceRow></InvoiceRow>
  <EpiDetails>
    <EpiIdentificationDetails>
      <EpiDate Format="CCYYMMDD">20120808</EpiDate>
      <EpiReference>2004468</EpiReference>
    </EpiIdentificationDetails>
    <EpiPartyDetails>
      <EpiBfiPartyDetails></EpiBfiPartyDetails>
      <EpiBeneficiaryPartyDetails>
        <EpiAccountID
IdentificationSchemeName="IBAN">FI7036363001126978</EpiAccountID>
      </EpiBeneficiaryPartyDetails>
    </EpiPartyDetails>
    <EpiPaymentInstructionDetails>
      <EpiInstructedAmount
AmountCurrencyIdentifier="EUR">10,98</EpiInstructedAmount>
      <EpiCharge ChargeOption="SLEV">SLEV</EpiCharge>
      <EpiDateOptionDate Format="CCYYMMDD">20121122</EpiDateOptionDate>
    </EpiPaymentInstructionDetails>
  </EpiDetails>
</Finvoice>
```

Appendix 5

```
// an example application creats a Finvoice.xml not only with mandatory data
//test2.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using dll2;

namespace Test3
{
    class Program
    {

        static void Main(string[] args)
        {
            FinvoiceDocument doc = new FinvoiceDocument();

            //1 MessageTransmissionDetails

             doc.putElementAttribute("MessageTransmissionDetails/MessageSenderDet
             ails/FromIdentifier", "FI8636100000000247");

             doc.putElementAttribute("MessageTransmissionDetails/MessageSenderDet
             ails/FromIntermediator", "TAPIFI22");

             doc.putElementAttribute("MessageTransmissionDetails/MessageReceiverD
             etails/ToIdentifier", "FI9836100000000322");

             doc.putElementAttribute("MessageTransmissionDetails/MessageReceiverD
             etails/ToIntermediator", "TAPIFI22");

             doc.putElementAttribute("MessageTransmissionDetails/MessageDetails/M
             essageIdentifier", "FKmalli_Jani3");

             doc.putElementAttribute("MessageTransmissionDetails/MessageDetails/M
             essageTimeStamp", "2012-10-04T08:12:41-0200");

            //1 SellerPartyDetails
             doc.putElementAttribute("SellerPartyDetails/SellerPartyIdentifier",
             "0235901-7");

             doc.putElementAttribute("SellerPartyDetails/SellerPartyIdentifierUrl
             Text", "jhkhjkjl");

             doc.putElementAttribute("SellerPartyDetails/SellerOrganisationName[1
             ]", "FK:n malli");

             doc.putElementAttribute("SellerPartyDetails/SellerOrganisationName[2
             ]", "Pullis Musiken");

             doc.putElementAttribute("SellerPartyDetails/SellerOrganisationDepart
             ment", "");

             doc.putElementAttribute("SellerPartyDetails/SellerOrganisationDepart
             ment", "");
```

```
 doc.putElementAttribute("SellerPartyDetails/SellerOrganisationTaxCod
 e", "FI01999207");

 doc.putElementAttribute("SellerPartyDetails/SellerOrganisationTaxCod
 eUrlText",
 "http://europa.eu.int/comm/taxation_customs/vies/fi/vieshome.htm");

 doc.putElementAttribute("SellerPartyDetails/SellerPostalAddressDetai
 ls/SellerStreetName", "Puukatu 2 F");

 doc.putElementAttribute("SellerPartyDetails/SellerPostalAddressDetai
 ls/SellerTownName", "Helsinki");

 doc.putElementAttribute("SellerPartyDetails/SellerPostalAddressDetai
 ls/SellerPostCodeIdentifier", "00112");

 doc.putElementAttribute("SellerPartyDetails/SellerPostalAddressDetai
 ls/CountryCode", "FI");

 doc.putElementAttribute("SellerPartyDetails/SellerPostalAddressDetai
 ls/CountryName", "Suomi");

 doc.putElementAttribute("SellerPartyDetails/SellerPostalAddressDetai
 ls/SellerPostOfficeBoxIdentifier", "");

//level 1
doc.putElementAttribute("SellerOrganisationUnitNumber", "");
doc.putElementAttribute("SellerSiteCode", "");
doc.putElementAttribute("SellerContactPersonName", "Hanna Paananen");

 doc.putElementAttribute("SellerCommunicationDetails/SellerPhoneNumbe
 rIdentifier", "");

 doc.putElementAttribute("SellerCommunicationDetails/SellerEmailaddre
 ssIdentifier", "hanna.paananen@pullinmusiikki.fi");

//1 SellerInformationDetails

 doc.putElementAttribute("SellerInformationDetails/SellerVatRegistrat
 ionDate", "00000000");

 doc.putElementAttribute("SellerInformationDetails/SellerVatRegistrat
 ionDate/@Format", "CCYYMMDD");
doc.putElementAttribute("SellerInformationDetails/SellerPhoneNumber",
"(09) 1231");

doc.putElementAttribute("SellerInformationDetails/SellerFaxNumber",
"(09) 1232500");

doc.putElementAttribute("SellerInformationDetails/SellerCommonEmailad
dressIdentifier", "webmaster@pullinmusiikki.fi");

doc.putElementAttribute("SellerInformationDetails/SellerWebaddressIde
ntifier", "www.pullinmusiikki.fi");

doc.putElementAttribute("SellerInformationDetails/SellerFreeText",
"Meidän kanssa kannattaa tehdä kauppaa");

//2 SellerAccountDetails
```

```
doc.putElementAttribute("SellerInformationDetails/SellerAccountDetail
s[1]/SellerAccountID", "FI2721221222212227");

doc.putElementAttribute("SellerInformationDetails/SellerAccountDetail
s[1]/SellerAccountID/@IdentificationSchemeName", "IBAN");

doc.putElementAttribute("SellerInformationDetails/SellerAccountDetail
s[1]/SellerBic/@IdentificationSchemeName", "BIC");

doc.putElementAttribute("SellerInformationDetails/SellerAccountDetail
s[1]/SellerBic", "BANKFIHH");

doc.putElementAttribute("SellerInformationDetails/SellerAccountDetail
s[2]/SellerAccountID", "FI2757800750155448");

doc.putElementAttribute("SellerInformationDetails/SellerAccountDetail
s[2]/SellerAccountID/@IdentificationSchemeName", "IBAN");

doc.putElementAttribute("SellerInformationDetails/SellerAccountDetail
s[2]/SellerBic/@IdentificationSchemeName", "BIC");

doc.putElementAttribute("SellerInformationDetails/SellerAccountDetail
s[2]/SellerBic", "BANKFIHH");
//2 InvoiceRecipientDetails

 doc.putElementAttribute("SellerInformationDetails/InvoiceRecipientDe
 tails/InvoiceRecipientAddress", "FI2757800750155448");

 doc.putElementAttribute("SellerInformationDetails/InvoiceRecipientDe
 tails/InvoiceRecipientIntermediatorAddress", "BANKFIXX");

//1 InvoiceSenderPartyDetails

 doc.putElementAttribute("InvoiceSenderPartyDetails/InvoiceSenderPart
 yIdentifier", "765432-1");

 doc.putElementAttribute("InvoiceSenderPartyDetails/InvoiceSenderOrga
 nisationName", "Tilitoimisto AB Oy");

 doc.putElementAttribute("InvoiceSenderPartyDetails/InvoiceSenderOrga
 nisationTaxCode", "FI07654321");

//level1

 doc.putElementAttribute("InvoiceRecipientOrganisationUnitNumber",
 "");
doc.putElementAttribute("InvoiceRecipientSiteCode", "");
doc.putElementAttribute("InvoiceRecipientContactPersonName", "");
doc.putElementAttribute("InvoiceRecipientContactPersonName", "");

 doc.putElementAttribute("InvoiceRecipientContactPersonDepartment",
 "");
doc.putElementAttribute("InvoiceRecipientLanguageCode", "FI");

//1 BuyerPartyDetails

 doc.putElementAttribute("BuyerPartyDetails/BuyerPartyIdentifier",
 "0123456-7");
```

```
doc.putElementAttribute("BuyerPartyDetails/BuyerOrganisationName",
"Sensorit Oy");

doc.putElementAttribute("BuyerPartyDetails/BuyerOrganisationDepartmen
t", "");
 doc.putElementAttribute("BuyerPartyDetails/BuyerOrganisationDepartme
nt", "");
doc.putElementAttribute("BuyerPartyDetails/BuyerOrganisationTaxCode",
"FI01234567");

doc.putElementAttribute("BuyerPartyDetails/BuyerPostalAddressDetails/
BuyerStreetName", "Sempalokatu 2");

doc.putElementAttribute("BuyerPartyDetails/BuyerPostalAddressDetails/
BuyerTownName", "Helsinki");

doc.putElementAttribute("BuyerPartyDetails/BuyerPostalAddressDetails/
BuyerPostCodeIdentifier", "00122");

doc.putElementAttribute("BuyerPartyDetails/BuyerPostalAddressDetails/
CountryCode", "FI");

doc.putElementAttribute("BuyerPartyDetails/BuyerPostalAddressDetails/
CountryName", "Suomi");

doc.putElementAttribute("BuyerPartyDetails/BuyerPostalAddressDetails/
BuyerPostOfficeBoxIdentifier", "");

//level1
doc.putElementAttribute("BuyerOrganisationUnitNumber", "");
doc.putElementAttribute("BuyerSiteCode", "");
             doc.putElementAttribute("BuyerContactPersonName",
 "Hannes Puumalainen");

//<BuyerCommunicationDetails>

 doc.putElementAttribute("BuyerCommunicationDetails/BuyerPhoneNumberI
 dentifier", "puh. 050-1234567");

 doc.putElementAttribute("BuyerCommunicationDetails/BuyerEmailaddress
 Identifier", "hannes.puumalainen@sensorit.fi");

//<DeliveryPartyDetails>

 doc.putElementAttribute("DeliveryPartyDetails/DeliveryPartyIdentifie
 r", "");

 doc.putElementAttribute("DeliveryPartyDetails/DeliveryOrganisationNa
 me", "Helsingin Tanssihalli");

 doc.putElementAttribute("DeliveryPartyDetails/DeliveryPostalAddressD
 etails/DeliveryStreetName", "Satamakatu 2");

 doc.putElementAttribute("DeliveryPartyDetails/DeliveryPostalAddressD
 etails/DeliveryTownName", "Helsinki");

 doc.putElementAttribute("DeliveryPartyDetails/DeliveryPostalAddressD
 etails/DeliveryPostCodeIdentifier", "00100");
 doc.putElementAttribute("DeliveryPartyDetails/DeliveryPostalAddressD
 etails/CountryCode", "FI");

 doc.putElementAttribute("DeliveryPartyDetails/DeliveryPostalAddressD
 etails/CountryName", "Suomi");

 doc.putElementAttribute("DeliveryPartyDetails/DeliveryPostalAddressD
 etails/DeliveryPostofficeBoxIdentifier", "");

//<DeliveryDetails>
doc.putElementAttribute("DeliveryDetails/DeliveryDate", "20120808");

 doc.putElementAttribute("DeliveryDetails/DeliveryDate/@Format",
 "CCYYMMDD");

 doc.putElementAttribute("DeliveryDetails/DeliveryPeriodDetails/Start
 Date", "20120808");

 doc.putElementAttribute("DeliveryDetails/DeliveryPeriodDetails/Start
 Date/@Format", "CCYYMMDD");

 doc.putElementAttribute("DeliveryDetails/DeliveryPeriodDetails/EndDa
 te", "20120808");

 doc.putElementAttribute("DeliveryDetails/DeliveryPeriodDetails/EndDa
 te/@Format", "CCYYMMDD");
doc.putElementAttribute("DeliveryDetails/DeliveryMethodText",
"Noudetaan");
doc.putElementAttribute("DeliveryDetails/DeliveryTermsText",
"Vapaasti varastosta");
doc.putElementAttribute("DeliveryDetails/TerminalAddressText",
"Vantaan postiterminaali");
doc.putElementAttribute("DeliveryDetails/WaybillIdentifier",
"419/2009");
doc.putElementAttribute("DeliveryDetails/WaybillTypeCode", "WBGF");
doc.putElementAttribute("DeliveryDetails/ClearanceIdentifier", "");

 doc.putElementAttribute("DeliveryDetails/DeliveryNoteIdentifier",
 "");
doc.putElementAttribute("DeliveryDetails/DelivererIdentifier", "Del.
ID. 12222");
doc.putElementAttribute("DeliveryDetails/DelivererName[1]", "Packgage
Ltd.");
doc.putElementAttribute("DeliveryDetails/DelivererName[2]",
"Lähettifirma Ab");
doc.putElementAttribute("DeliveryDetails/DelivererCountryCode",
"FI");
doc.putElementAttribute("DeliveryDetails/DelivererCountryName",
"Suomi");
doc.putElementAttribute("DeliveryDetails/ManufacturerIdentifier",
"D13331231233");
doc.putElementAttribute("DeliveryDetails/ManufacturerName", "AKG
International");
doc.putElementAttribute("DeliveryDetails/ManufacturerCountryCode",
"DE");
doc.putElementAttribute("DeliveryDetails/ManufacturerCountryName",
"Germany");
```

```
doc.putElementAttribute("DeliveryDetails/ManufacturerOrderIdentifier"
, "");

//<InvoiceDetails>
doc.putElementAttribute("InvoiceDetails/InvoiceTypeCode", "INV01");

 doc.putElementAttribute("InvoiceDetails/InvoiceTypeCode/@CodeListAge
 ncyIdentifier", "SPY");
doc.putElementAttribute("InvoiceDetails/InvoiceTypeText", "Lasku");
doc.putElementAttribute("InvoiceDetails/OriginCode", "Original");
doc.putElementAttribute("InvoiceDetails/InvoiceNumber", "12345");
doc.putElementAttribute("InvoiceDetails/InvoiceDate", "20120808");

 doc.putElementAttribute("InvoiceDetails/InvoiceDate/@Format",
 "CCYYMMDD");

 doc.putElementAttribute("InvoiceDetails/InvoiceTotalVatIncludedAmoun
 t", "10,98");

 doc.putElementAttribute("InvoiceDetails/InvoiceTotalVatIncludedAmoun
 t/@AmountCurrencyIdentifier", "EUR");

//<InvoiceRow>
doc.putElementAttribute("InvoiceRow[1]/RowSubIdentifier", "221");
doc.putElementAttribute("InvoiceRow[1]/ArticleIdentifier", "123123");
doc.putElementAttribute("InvoiceRow[1]/ArticleGroupIdentifier", "");
doc.putElementAttribute("InvoiceRow[1]/ArticleName", "Apple");
doc.putElementAttribute("InvoiceRow[1]/ArticleInfoUrlText", "");
doc.putElementAttribute("InvoiceRow[1]/BuyerArticleIdentifier", "");
doc.putElementAttribute("InvoiceRow[1]/EanCode", "");
 doc.putElementAttribute("InvoiceRow[1]/RowRegistrationNumberIdentifi
 er", "");
doc.putElementAttribute("InvoiceRow[1]/SerialNumberIdentifier", "");
doc.putElementAttribute("InvoiceRow[1]/RowActionCode", "");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubIdentifier",
"");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowPositionId
entifier", "221");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubArticleIdenti
fier", "");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubArticleGroupI
dentifier", "");
doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubArticleName",
"Toimitusyhteenveto");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowIdentifier
Date", "20090808");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowIdentifier
Date/@Format", "CCYYMMDD");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowDeliveryId
entifier", "TIL12312");
```

```
doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowDeliveryId
entifierUrlText", "");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowDeliveryDa
te", "20090808");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowDeliveryDa
te/@Format", "CCYYMMDD");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowVatRatePer
cent", "22");
doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowVatCode",
"");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowVatAmount"
, "106,42");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowVatAmount/
@AmountCurrencyIdentifier", "EUR");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowVatExclude
dAmount", "456");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowVatExclude
dAmount/@AmountCurrencyIdentifier", "EUR");
doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowAmount",
"600");

doc.putElementAttribute("InvoiceRow[2]/SubInvoiceRow/SubRowAmount/@A
mountCurrencyIdentifier", "EUR");

//<EpiDetails>

doc.putElementAttribute("EpiDetails/EpiIdentificationDetails/EpiDate
", "20120808");

doc.putElementAttribute("EpiDetails/EpiIdentificationDetails/EpiDate
/@Format", "CCYYMMDD");

doc.putElementAttribute("EpiDetails/EpiIdentificationDetails/EpiRefe
rence", "2004468");
doc.putElementAttribute("EpiDetails/EpiPartyDetails/EpiBfiPartyDetai
ls", "");

doc.putElementAttribute("EpiDetails/EpiPartyDetails/EpiBeneficiaryPa
rtyDetails/EpiAccountID", "FI7036363001126978");

doc.putElementAttribute("EpiDetails/EpiPartyDetails/EpiBeneficiaryPa
rtyDetails/EpiAccountID/@IdentificationSchemeName", "IBAN");

doc.putElementAttribute("EpiDetails/EpiPaymentInstructionDetails/Epi
InstructedAmount", "10,98");
doc.putElementAttribute("EpiDetails/EpiPaymentInstructionDetails/Epi
InstructedAmount/@AmountCurrencyIdentifier", "EUR");

doc.putElementAttribute("EpiDetails/EpiPaymentInstructionDetails/Epi
Charge", "SLEV");
```

```
            doc.putElementAttribute("EpiDetails/EpiPaymentInstructionDetails/Epi
            Charge/@ChargeOption", "SLEV");

            doc.putElementAttribute("EpiDetails/EpiPaymentInstructionDetails/Epi
            DateOptionDate", "20121122");

            doc.putElementAttribute("EpiDetails/EpiPaymentInstructionDetails/Epi
            DateOptionDate/@Format", "CCYYMMDD");

           doc.putElementAttribute("InvoiceUrlNameText", "");
           doc.putElementAttribute("InvoiceUrlNameText", "");
           doc.putElementAttribute("InvoiceUrlText", "");
           doc.putElementAttribute("InvoiceUrlText", "");
           doc.putElementAttribute("StorageUrlText", "");
           doc.putElementAttribute("LayOutIdentifier", "");
           doc.putElementAttribute("InvoiceSegmentIdentifier", "");
           doc.putElementAttribute("AcceptanceStampText", "");
           doc.putElementAttribute("OriginalInvoiceFormat", "");


           bool valid = doc.validateXml("Finvoice.xsd", Console.Out );
           if (valid)
           {
               doc.saveFinvoiceDocument();
           }
           Console.Read();
       }

   }
}
```

Appendix 6

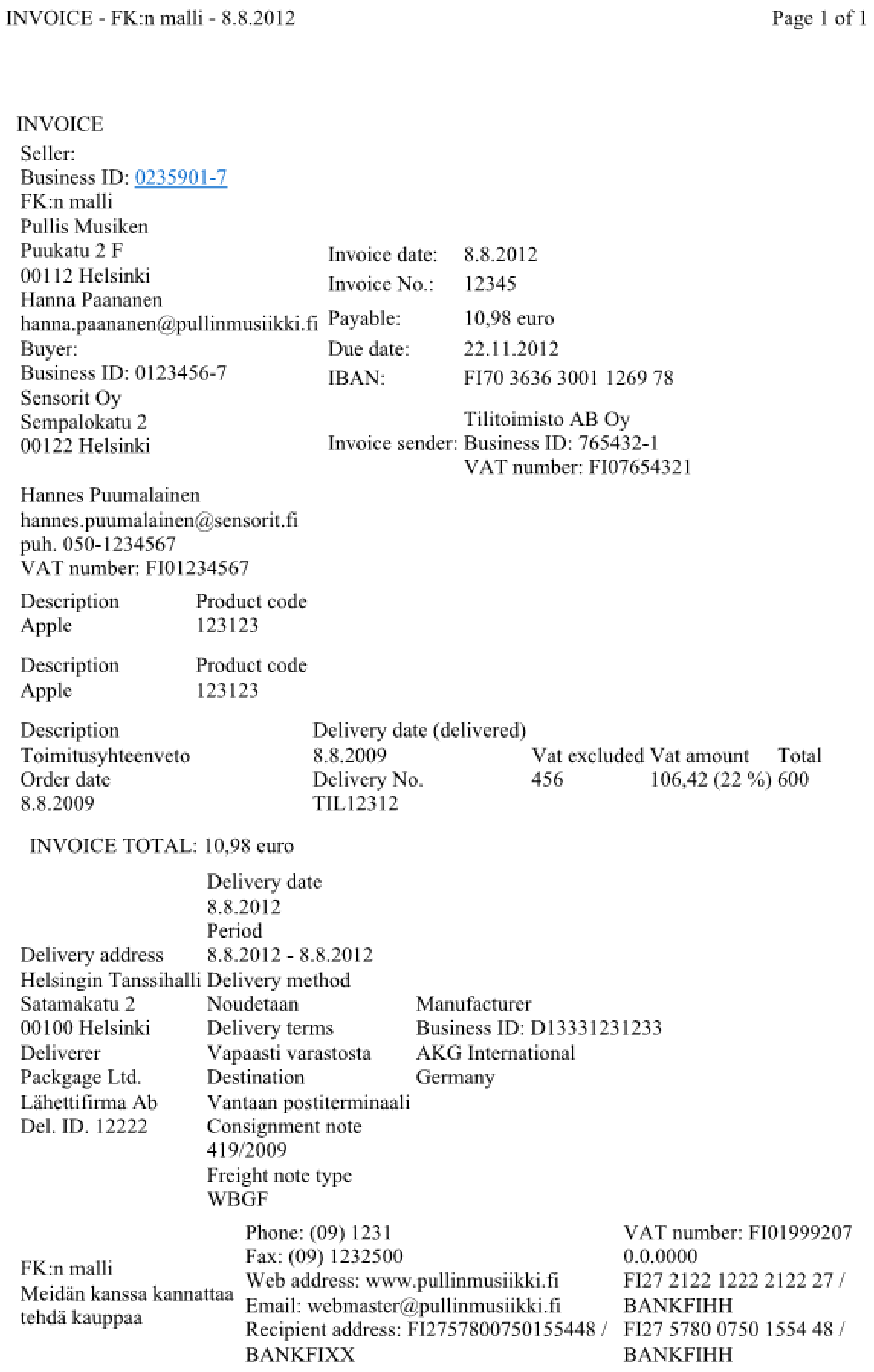
```xml
<!--This is Finvice.xml produced by test2.cs-->
<?xml version="1.0" encoding="iso-8859-15"?>
<?xml-stylesheet type="text/xsl" href="Finvoice.xsl"?>
<Finvoice Version="2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Finvoice.xsd">
  <MessageTransmissionDetails>
    <MessageSenderDetails>
      <FromIdentifier>FI8636100000000247</FromIdentifier>
      <FromIntermediator>TAPIFI22</FromIntermediator>
    </MessageSenderDetails>
    <MessageReceiverDetails>
      <ToIdentifier>FI9836100000000322</ToIdentifier>
      <ToIntermediator>TAPIFI22</ToIntermediator>
    </MessageReceiverDetails>
    <MessageDetails>
      <MessageIdentifier>FKmalli_Jani3</MessageIdentifier>
      <MessageTimeStamp>2012-10-04T08:12:41-0200</MessageTimeStamp>
    </MessageDetails>
  </MessageTransmissionDetails>
  <SellerPartyDetails>
    <SellerPartyIdentifier>0235901-7</SellerPartyIdentifier>
    <SellerPartyIdentifierUrlText>jhkhjkjl</SellerPartyIdentifierUrlText>
    <SellerOrganisationName>FK:n malli</SellerOrganisationName>
    <SellerOrganisationName>Pullis Musiken</SellerOrganisationName>
    <SellerOrganisationDepartment></SellerOrganisationDepartment>
    <SellerOrganisationTaxCode>FI01999207</SellerOrganisationTaxCode>

<SellerOrganisationTaxCodeUrlText>http://europa.eu.int/comm/taxation_customs/vies
/fi/vieshome.htm</SellerOrganisationTaxCodeUrlText>
    <SellerPostalAddressDetails>
      <SellerStreetName>Puukatu 2 F</SellerStreetName>
      <SellerTownName>Helsinki</SellerTownName>
      <SellerPostCodeIdentifier>00112</SellerPostCodeIdentifier>
      <CountryCode>FI</CountryCode>
      <CountryName>Suomi</CountryName>
      <SellerPostOfficeBoxIdentifier></SellerPostOfficeBoxIdentifier>
    </SellerPostalAddressDetails>
  </SellerPartyDetails>
  <SellerOrganisationUnitNumber></SellerOrganisationUnitNumber>
  <SellerSiteCode></SellerSiteCode>
  <SellerContactPersonName>Hanna Paananen</SellerContactPersonName>
  <SellerCommunicationDetails>
    <SellerPhoneNumberIdentifier></SellerPhoneNumberIdentifier>

<SellerEmailaddressIdentifier>hanna.paananen@pullinmusiikki.fi</SellerEmailaddres
sIdentifier>
  </SellerCommunicationDetails>
  <SellerInformationDetails>
    <SellerVatRegistrationDate
Format="CCYYMMDD">00000000</SellerVatRegistrationDate>
    <SellerPhoneNumber>(09) 1231</SellerPhoneNumber>
    <SellerFaxNumber>(09) 1232500</SellerFaxNumber>

<SellerCommonEmailaddressIdentifier>webmaster@pullinmusiikki.fi</SellerCommonEmai
laddressIdentifier>
```

```
<SellerWebaddressIdentifier>www.pullinmusiikki.fi</SellerWebaddressIdentifier>
    <SellerFreeText>Meidän kanssa kannattaa tehdä kauppaa</SellerFreeText>
    <SellerAccountDetails>
      <SellerAccountID
IdentificationSchemeName="IBAN">FI2721221222212227</SellerAccountID>
      <SellerBic IdentificationSchemeName="BIC">BANKFIHH</SellerBic>
    </SellerAccountDetails>
    <SellerAccountDetails>
      <SellerAccountID
IdentificationSchemeName="IBAN">FI2757800750155448</SellerAccountID>
      <SellerBic IdentificationSchemeName="BIC">BANKFIHH</SellerBic>
    </SellerAccountDetails>
    <InvoiceRecipientDetails>
      <InvoiceRecipientAddress>FI2757800750155448</InvoiceRecipientAddress>

<InvoiceRecipientIntermediatorAddress>BANKFIXX</InvoiceRecipientIntermediatorAddr
ess>
    </InvoiceRecipientDetails>
  </SellerInformationDetails>
  <InvoiceSenderPartyDetails>
    <InvoiceSenderPartyIdentifier>765432-1</InvoiceSenderPartyIdentifier>
    <InvoiceSenderOrganisationName>Tilitoimisto AB
Oy</InvoiceSenderOrganisationName>

<InvoiceSenderOrganisationTaxCode>FI07654321</InvoiceSenderOrganisationTaxCode>
  </InvoiceSenderPartyDetails>

<InvoiceRecipientOrganisationUnitNumber></InvoiceRecipientOrganisationUnitNumber>
  <InvoiceRecipientSiteCode></InvoiceRecipientSiteCode>
  <InvoiceRecipientContactPersonName></InvoiceRecipientContactPersonName>

<InvoiceRecipientContactPersonDepartment></InvoiceRecipientContactPersonDepartmen
t>
  <InvoiceRecipientLanguageCode>FI</InvoiceRecipientLanguageCode>
  <BuyerPartyDetails>
    <BuyerPartyIdentifier>0123456-7</BuyerPartyIdentifier>
    <BuyerOrganisationName>Sensorit Oy</BuyerOrganisationName>
    <BuyerOrganisationDepartment></BuyerOrganisationDepartment>
    <BuyerOrganisationTaxCode>FI01234567</BuyerOrganisationTaxCode>
    <BuyerPostalAddressDetails>
      <BuyerStreetName>Sempalokatu 2</BuyerStreetName>
      <BuyerTownName>Helsinki</BuyerTownName>
      <BuyerPostCodeIdentifier>00122</BuyerPostCodeIdentifier>
      <CountryCode>FI</CountryCode>
      <CountryName>Suomi</CountryName>
      <BuyerPostOfficeBoxIdentifier></BuyerPostOfficeBoxIdentifier>
    </BuyerPostalAddressDetails>
  </BuyerPartyDetails>
  <BuyerOrganisationUnitNumber></BuyerOrganisationUnitNumber>
  <BuyerSiteCode></BuyerSiteCode>
  <BuyerContactPersonName>Hannes Puumalainen</BuyerContactPersonName>
  <BuyerCommunicationDetails>
    <BuyerPhoneNumberIdentifier>puh. 050-1234567</BuyerPhoneNumberIdentifier>

<BuyerEmailaddressIdentifier>hannes.puumalainen@sensorit.fi</BuyerEmailaddressIde
ntifier>
  </BuyerCommunicationDetails>
  <DeliveryPartyDetails>
    <DeliveryPartyIdentifier></DeliveryPartyIdentifier>
```

```xml
      <DeliveryOrganisationName>Helsingin Tanssihalli</DeliveryOrganisationName>
      <DeliveryPostalAddressDetails>
        <DeliveryStreetName>Satamakatu 2</DeliveryStreetName>
        <DeliveryTownName>Helsinki</DeliveryTownName>
        <DeliveryPostCodeIdentifier>00100</DeliveryPostCodeIdentifier>
        <CountryCode>FI</CountryCode>
        <CountryName>Suomi</CountryName>
        <DeliveryPostofficeBoxIdentifier></DeliveryPostofficeBoxIdentifier>
      </DeliveryPostalAddressDetails>
    </DeliveryPartyDetails>
    <DeliveryDetails>
      <DeliveryDate Format="CCYYMMDD">20120808</DeliveryDate>
      <DeliveryPeriodDetails>
        <StartDate Format="CCYYMMDD">20120808</StartDate>
        <EndDate Format="CCYYMMDD">20120808</EndDate>
      </DeliveryPeriodDetails>
      <DeliveryMethodText>Noudetaan</DeliveryMethodText>
      <DeliveryTermsText>Vapaasti varastosta</DeliveryTermsText>
      <TerminalAddressText>Vantaan postiterminaali</TerminalAddressText>
      <WaybillIdentifier>419/2009</WaybillIdentifier>
      <WaybillTypeCode>WBGF</WaybillTypeCode>
      <ClearanceIdentifier></ClearanceIdentifier>
      <DeliveryNoteIdentifier></DeliveryNoteIdentifier>
      <DelivererIdentifier>Del. ID. 12222</DelivererIdentifier>
      <DelivererName>Packgage Ltd.</DelivererName>
      <DelivererName>Lähettifirma Ab</DelivererName>
      <DelivererCountryCode>FI</DelivererCountryCode>
      <DelivererCountryName>Suomi</DelivererCountryName>
      <ManufacturerIdentifier>D13331231233</ManufacturerIdentifier>
      <ManufacturerName>AKG International</ManufacturerName>
      <ManufacturerCountryCode>DE</ManufacturerCountryCode>
      <ManufacturerCountryName>Germany</ManufacturerCountryName>
      <ManufacturerOrderIdentifier></ManufacturerOrderIdentifier>
    </DeliveryDetails>
    <InvoiceDetails>
      <InvoiceTypeCode CodeListAgencyIdentifier="SPY">INV01</InvoiceTypeCode>
      <InvoiceTypeText>Lasku</InvoiceTypeText>
      <OriginCode>Original</OriginCode>
      <InvoiceNumber>12345</InvoiceNumber>
      <InvoiceDate Format="CCYYMMDD">20120808</InvoiceDate>
      <InvoiceTotalVatIncludedAmount
AmountCurrencyIdentifier="EUR">10,98</InvoiceTotalVatIncludedAmount>
    </InvoiceDetails>
    <InvoiceRow>
      <RowSubIdentifier>221</RowSubIdentifier>
      <ArticleIdentifier>123123</ArticleIdentifier>
      <ArticleGroupIdentifier></ArticleGroupIdentifier>
      <ArticleName>Apple</ArticleName>
      <ArticleInfoUrlText></ArticleInfoUrlText>
      <BuyerArticleIdentifier></BuyerArticleIdentifier>
      <EanCode></EanCode>
      <RowRegistrationNumberIdentifier></RowRegistrationNumberIdentifier>
      <SerialNumberIdentifier></SerialNumberIdentifier>
      <RowActionCode></RowActionCode>
    </InvoiceRow>
    <InvoiceRow>
      <SubInvoiceRow>
        <SubIdentifier></SubIdentifier>
        <SubRowPositionIdentifier>221</SubRowPositionIdentifier>
        <SubArticleIdentifier></SubArticleIdentifier>
      <SubArticleGroupIdentifier></SubArticleGroupIdentifier>
      <SubArticleName>Toimitusyhteenveto</SubArticleName>
      <SubRowIdentifierDate Format="CCYYMMDD">20090808</SubRowIdentifierDate>
      <SubRowDeliveryIdentifier>TIL12312</SubRowDeliveryIdentifier>
      <SubRowDeliveryIdentifierUrlText></SubRowDeliveryIdentifierUrlText>
      <SubRowDeliveryDate Format="CCYYMMDD">20090808</SubRowDeliveryDate>
      <SubRowVatRatePercent>22</SubRowVatRatePercent>
      <SubRowVatCode></SubRowVatCode>
      <SubRowVatAmount AmountCurrencyIdentifier="EUR">106,42</SubRowVatAmount>
      <SubRowVatExcludedAmount
AmountCurrencyIdentifier="EUR">456</SubRowVatExcludedAmount>
      <SubRowAmount AmountCurrencyIdentifier="EUR">600</SubRowAmount>
    </SubInvoiceRow>
  </InvoiceRow>
  <EpiDetails>
    <EpiIdentificationDetails>
      <EpiDate Format="CCYYMMDD">20120808</EpiDate>
      <EpiReference>2004468</EpiReference>
    </EpiIdentificationDetails>
    <EpiPartyDetails>
      <EpiBfiPartyDetails></EpiBfiPartyDetails>
      <EpiBeneficiaryPartyDetails>
        <EpiAccountID
IdentificationSchemeName="IBAN">FI7036363001126978</EpiAccountID>
      </EpiBeneficiaryPartyDetails>
    </EpiPartyDetails>
    <EpiPaymentInstructionDetails>
      <EpiInstructedAmount
AmountCurrencyIdentifier="EUR">10,98</EpiInstructedAmount>
      <EpiCharge ChargeOption="SLEV">SLEV</EpiCharge>
      <EpiDateOptionDate Format="CCYYMMDD">20121122</EpiDateOptionDate>
    </EpiPaymentInstructionDetails>
  </EpiDetails>
  <InvoiceUrlNameText></InvoiceUrlNameText>
  <InvoiceUrlText></InvoiceUrlText>
  <StorageUrlText></StorageUrlText>
  <LayOutIdentifier></LayOutIdentifier>
  <InvoiceSegmentIdentifier></InvoiceSegmentIdentifier>
  <AcceptanceStampText></AcceptanceStampText>
  <OriginalInvoiceFormat></OriginalInvoiceFormat>
</Finvoice>
```

Appendix 7

Screen shot of transformed Finvoice.xml



INVOICE - FK:n malli - 8.8.2012 Page 1 of 1

INVOICE

Seller:
Business ID: 0235901-7
FK:n malli
Pullis Musiken
Puukatu 2 F
00112 Helsinki
Hanna Paananen
hanna.paananen@pullinmusiikki.fi

Buyer:
Business ID: 0123456-7
Sensorit Oy
Sempalokatu 2
00122 Helsinki

Invoice date:	8.8.2012
Invoice No.:	12345
Payable:	10,98 euro
Due date:	22.11.2012
IBAN:	FI70 3636 3001 1269 78
Invoice sender:	Tilitoimisto AB Oy Business ID: 765432-1 VAT number: FI07654321

Hannes Puumalainen
hannes.puumalainen@sensorit.fi
puh. 050-1234567
VAT number: FI01234567

Description	Product code
Apple	123123

Description	Product code
Apple	123123

Description
Toimitusyhteenveto
Order date
8.8.2009

Delivery date (delivered)
8.8.2009
Delivery No.
TIL12312

Vat excluded	Vat amount	Total
456	106,42 (22 %)	600

INVOICE TOTAL: 10,98 euro

Delivery address
Helsingin Tanssihalli
Satamakatu 2
00100 Helsinki
Deliverer
Packgage Ltd.
Lähettifirma Ab
Del. ID. 12222

Delivery date
8.8.2012
Period
8.8.2012 - 8.8.2012
Delivery method
Noudetaan
Delivery terms
Vapaasti varastosta
Destination
Vantaan postiterminaali
Consignment note
419/2009
Freight note type
WBGF

Manufacturer
Business ID: D13331231233
AKG International
Germany

FK:n malli
Meidän kanssa kannattaa tehdä kauppaa

Phone: (09) 1231
Fax: (09) 1232500
Web address: www.pullinmusiikki.fi
Email: webmaster@pullinmusiikki.fi
Recipient address: FI2757800750155448 / BANKFIXX

VAT number: FI01999207
0.0.0000
FI27 2122 1222 2122 27 / BANKFIHH
FI27 5780 0750 1554 48 / BANKFIHH