Fiscalization Service

Technical Specification

Version description

Version	Description of Change
v01	Initial Version
v02	Terminology which is used in the Law inserted into the introduction chapter Final version

CONTENTS

1. INTRODUCTION 5

1.1 USED ABBREVIATIONS 5
1.2 TERMINOLOGY 6

2. ENVIRONMENTS 7

2.1 PREPARATION WORKS FOR FISCALIZATION SERVICE USE 7
2.2 TOPOLOGY 7
2.2.1 CIS access point 8
2.2.2 Internet connection 8
2.2.3 Information system of the client 8
2.3 CONDITIONS FOR CONNECTION TO CIS 8
2.3.1 Network preconditions and recommendations 8
2.3.2 Security preconditions 9
2.3.3 Aplication preconditions 9

3. INTERFACE 10

3.1 INTERFACE VERSIONING 10
3.2 DATA MESSAGE CODING 10
3.3 DATA MESSAGE STRUCTURE 11
3.4 REGISTER NEW TCR 11
3.4.1 Register TCRRequest Data Message 13
3.4.2 Register TCR Response Data Message 15
3.4.3 Mandatory Controls 16
3.4.4 Error Message 17
3.4.5 Example XML 17
3.5 REGISTER TCR CASH BALANCE 18
3.5.1 REGISTER TCR CASH BALANCE REQUEST DATA MESSAGE 20
3.5.2 Register TCR Cash Balance Response Data Message 22
3.5.3 Mandatory Controls 23
3.5.4 Error Message 23
3.5.5 Example XML 23
3.6 REGISTER INVOICE 25
3.6.1 UML process diagram 27
3.6.2 Register invoice request data message 28
3.6.3 Register Invoice Response Data Message 42
3.6.4 Error Message 43
3.6.5 Mandatory Controls 43
3.6.6 Optional Controls 44
3.6.7 Example XML 44
3.6.8 Bulk upload of invoices 45
3.6.9 Checking invoice registration in web application "Invoice check" 47
3.7 WAREHOUSE TRANSFER NOTE 48
3.7.1 Warehouse transfer note request message 50

3.7.2 Warehouse transfer note response message *56*
3.7.3 Error Message *57*
3.7.4 Example XML *57*
3.8 ERROR MESSAGES 58
3.8.1 XML Format *58*
3.8.2 Error Codes *61*
3.8.3 Example XML *62*
4. DATA EXPORT OF THE NON-FISCALIZED INVOICES 63
5. SECURITY 64
5.1 CERTIFICATES 64
5.2 TRANSPORT SECURITY 64
5.3 MESSAGE SECURITY 64
5.3.1 Request And Response Data Messasge Signing *65*
5.3.2 IIC Data Element *65*
5.3.3 WTNIC Data Element *67*
6. CODE EXAMPLES 70
6.1 IIC GENERATION CODE 70
6.1.1 Java Example *70*
6.1.2 C# Example *71*
6.2 WTNIC GENERATION CODE 71
6.2.1 Java Example *71*
6.2.2 C# Example *72*
6.3 SIGNATURE GENERATION CODE 73
6.3.1 Java Example *73*

1. Introduction

This document provides a description of the elements of the technical specifications for the Fiscalization Service.

Examples containing XML schema definitions and the Web service (WSDL), which describe the structure of the registered invoice data messages and the Web service used to receive them are provided as Annexes to this document.

1.1 USED ABBREVIATIONS

Abbreviation	Description	Terminology used in the Draft Law (if it is different)
CA	Certificate Authority	-
CIS	Central Invoice System	Central invoice platform
CPCM	CPCM is central portal for the control and management of taxpayers in the cash transaction subsystem	-
CRL	Certificate Revocation List	-
CRN	Cash Register Number	Electronic cash device number
FIC	Fiscal Identification Code (generated at server side after successful verification of the invoice)	UII – Unique invoice identifier
UII	Unique Invoice Identifier	-
GUID	Global Unique Identifier	-
IIC	Issuer's invoice code	ISC - Invoice Issuer's Security Code
WTNIC	Note Identification Code (warehouse transfer note identification code)	-
NUIS	National Unique Identification Number	-
OCSP	On-Line Certificate Status Protocol	-
SOAP	Message exchange protocol for XML messages as specified at: https://www.w3.org/TR/soap/	-
TCR	Taxpayer Cash Register. The same as billing device or electronic billing device.	Taxpayer's electronic cash device
TCRN	Taxpayer Cash Register Number	Taxpayer's electronic cash device number
UC	Use case	-
UUID	Universally Unique Identifier	-
TLS	"Transport Layer Security" is a protocol that provides communication security between client/server applications that communicate with each other over the Internet.	
WSDL	Web Services Description Language –XML-based language for description of functions offered by a WWW service as specified at http://www.w3.org/TR/wsdl	-
XML Schema	A XML-based language intended for definition of XML document structure as specified at http://www.w3.org/TR/xmlschema11-1/ and https://www.w3.org/TR/xmlschema11-2/	-

Table 1 – Used abbreviations

1.2 TERMINOLOGY

Term	Definition	Terminology used in the Draft Law (if it is different)
Response data message	A data structure in a defined format prescribed by the responsible authority, which contains the Fiscal Identification Code (FIC) and is used as acknowledgement of invoice and formal correctness of the registered invoice data message sent.	A data structure in a defined format prescribed by the responsible authority, which contains Unique invoice identifier (UII) and is used as acknowledgement of invoice and formal correctness of the registered invoice data message sent.
Error Data Message	A data structure in a defined format prescribed by the responsible authority, which contains an error code and its text description as a reaction to a registered invoice data message received containing critical errors preventing it from being processed, or when another error occurs which prevents the message being processed at the tax authority's side.	-
Invoice	An invoice is a proof of sale issued (in paper form or electronically) by a taxpayer to a person or entity making a purchase, which contains all information regarding totals of the sale and items. Invoice shall mean any document issued in paper or in electronic form, which satisfies the requirements provided under "ON INVOICES AND THE SYSTEM FOR MONITORING TRANSACTIONS Draft Law"	
Issuer	Person who is issuing the invoice. Issuer of the invoice is responsible for the fiscalization of the invoice in CIS. This person is in most cases the seller of goods and services but in case of self-billing invoice, the issuer is the buyer of goods and services.	-
Registered Invoice	Invoice which is registered on CIS containing FIC.	Invoice which is registered on Central Invoice Platform containing UII.
Registered invoice data message	A data structure in a defined format prescribed by the responsible authority, which contains information about the e-sale and other technical information necessary. This is a complete XML message containing information described in the relevant Web service standards: SOAP/WSDL/WS-Security, etc. A registered invoice data message is sent by a cash register to the tax authority's common technical equipment.	A data structure in a defined format prescribed by the responsible authority, which contains information about the e-sale and other technical information necessary. This is a complete XML message containing information described in the relevant Web service standards: SOAP/WSDL/WS-Security, etc. A registered invoice data message is sent by an electronic cash device to the tax authority's common technical equipment (Central invoice system).
Self-care portal	Self-care portal is a web application for taxpayers providing support for invoice fiscalization processes.	Self-care portal is a web application for taxpayers providing support for invoice fiscalization processes. This is part of Central Invoice System
Taxpayer's cash register	A device on the taxpayer's side, which sends information on registered invoices to the tax authority. This may signify, depending on the context, an end device such as a cash register, or additional SW and HW actually sending the registered invoices information. The data messages include an item marked as "Cash register ID", which identifies the end device (cash register). In other parts of the text, this term usually means the end device and the relevant SW and HW sending the data messages.	A device on the taxpayer's side, which sends information on registered invoices to the tax authority (to the Central invoice platform). This may signify, depending on the context, an end device such as a cash register, or additional SW and HW actually sending the registered invoices information. The data messages include an item marked as "Cash register ID", which identifies the end device (electronic cash device). In other parts of the text, this term usually means the end device and the relevant SW and HW sending the data messages.

Table 2 - Terminology

2. Environments

The government will publish Web service addresses for two types of environments: production environment and one or more test environments:

- **Test environment** will be used solely by software developers (developing software for cash registers), not by cash registers' end users. Sending a data message to the test environment shall not be considered sending of registered invoice information. The FIC returned by the test environment is not a valid FIC (it is different per prefix). In the test environment, digital certificates for cash registers may be issued using a simplified process by AKSHI.
- **Production environment** is intended for the taxpayers and will be used for routine operations, i.e. receipt and acknowledgement of data messages containing information on registered sales.

Endpoints:

- Test environment:
  - https://eFiskalizimi-test.tatime.gov.al/FiscalizationService-v1/FiscalizationService.wsdl
- Production environment:
  - https://eFiskalizimi.tatime.gov.al/FiscalizationService-v1/FiscalizationService.wsdl

2.1 PREPARATION WORKS FOR FISCALIZATION SERVICE USE

Details on this matter can be found in the chapter that covers this subject. Below is the process diagram.

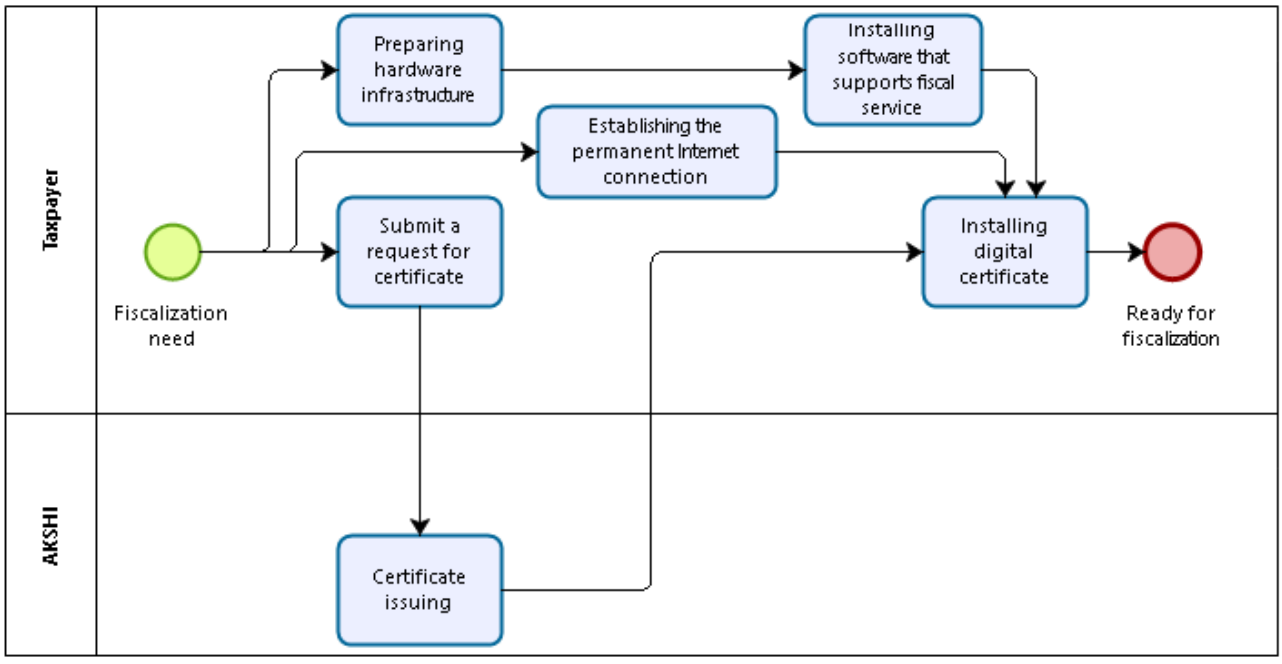

Figure 1 – Preparation activities for using a fiscal service

2.2 TOPOLOGY

Users access the CIS by initiating 1-way Transport Layer Security (TLS) connection. Clients exchange messages with Tax administration's access point using TLS channel by described procedure. Data exchange is synchronous, meaning access point answers on user's request immediately. Request and response messages formats are specified through XML schema.

2.2.1 CIS ACCESS POINT

Implementation and maintenance of the access point is a AKSHI. AKSHI will provide its users connection to the access point in two environments: production and test.

2.2.2 INTERNET CONNECTION

Access point will be available through internet networks in HTTPS protocol.

2.2.3 INFORMATION SYSTEM OF THE CLIENT

Clients are obliged to provide hardware and software support for messages exchange with access point. As shown on image below, there is no mediatory component development planned. Development of the hardware-software solutions is in client's domain of business. Client is also obliged to secure internet connection to CIS access point with needed bandwidth. Platform choice and software solution implementation Is in client's domain and such information is not needed to be told to AKSHI, but this solution is object of certification from AKSHI and DPT.

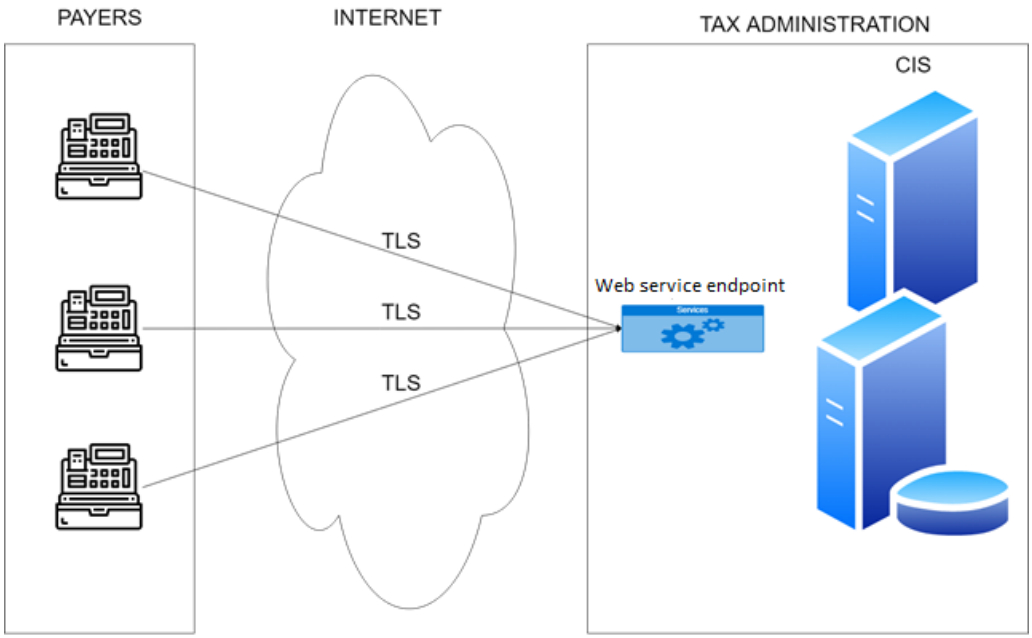

Figure 2 – Information system of the client

2.3 CONDITIONS FOR CONNECTION TO CIS

Central information system (CIS) of the Tax administration will be available in two environments: production and test.

Connection conditions are similar but differ in addresses of their access points and certificates. For production environment and test environment, different certificates are issued by AKSHI. Environments are not different in its functionality (besides new functionalities development), only difference is in data – test environment uses test data.

2.3.1 NETWORK PRECONDITIONS AND RECOMMENDATIONS

To connect to the CIS of the Tax administration, client system needs to fulfil these conditions:

Network type	**Internet**

Recommended open TCP ports to CIS	**443**

Network recommendations for client system are:

Link characteristics	**Permanent symmetric link**
Bandwidth	**2 Mb/s at minimum (up to 40 messages per second with assumption that message takes 6 KB)**

2.3.2 SECURITY PRECONDITIONS

All communication with CIS of the Tax administration is protected by 1-way TLS encryption at the transport layer. In production environment CIS presents itself to client with a TLS certificate issued by AKSHI production CA, while in test environment the certificate is issued by AKSHI test CA.

Protection at the transport layer	**HTTPS (TLS v1.1 and v1.2, AES_256 encryption at least)**
Certificates for the electronic signing	**Certificate type: application digital certificate for fiscalization**

2.3.3 APLICATION PRECONDITIONS

CIS' functionality is available to its clients using web-service technology. That is the reason for client application (or infrastructure, depending on realization) needs to fulfil these preconditions:

Client creation standards	**WS-1**
Service type	**Document-literal**
Application protocol	**SOAP/HTTPS (SOAP 1.1)**
Code site of the request message XML	**UTF-8**

3. Interface

Interface for exchanging the data between the taxpayer and CIS regarding the fiscalization will be SOAP web service. Messages are in XML format according to the standards of SOAP messages.

The web service has several operations which will be used by the taxpayer who needs to do the fiscalization of the invoices. Invoices are issued by the electronic billing device represented by its code. The code is assigned in operation of registration of electronic billing device which needs to be executed during the installation of each electronic billing device. At the beginning of each day, electronic billing device which handles cash transactions must register the amount of cash in the deposit and only then it should start to issue the invoices. Each invoice should be registered to the fiscalization service and upon successful registration the invoice is assigned FIC which is printed on the invoice. In case that the invoice needs to be corrected, new corrective invoice is issued referencing the invoice which needs to be corrected. This new corrective invoice describes the new changed final state and not the changes from the original invoice. During the day, electronic billing devices for cash payments should register current cash balance (it is recommended to do it when the operators of electronic billing device changes)as well as any withdrawal or deposit of cash in the electronic billing device. Each of this operation is explained in its chapter together with the list of elements of the exchanged messages.

Taxpayers should also register warehouse transfer notes for all goods transferred between warehouses and sale premises inside territory of Republic of Albania.

Message sent by the taxpayer to CIS is the request message to which CIS replies by sending the response message. In case of an error, the error message is sent in the response with its structure. Request and response messages (except for the error message) all have the following parts: header (general info about the message), data (data specific for the operation), signature (digital signature signed by the person who is sending the message which provides the identity of the sender and info to verify that the data of the message is not changed). Signature is explained in chapter 5.3.

3.1 INTERFACE VERSIONING

Versioning of the fiscalization service will be based on semantic versioning schema. Each version has a version number assigned expressed as “MAJOR.MINOR.PATCH” each of which are integers incremented according to these rules:

- MAJOR version is increased when there are incompatible API changes. New interface will be provided, and old interface will remain for some period. Clients are expected to upgrade to new version as described in release notes of the new version.
- MINOR version is increased when a functionality is added in a backwards-compatible manner. Current interface remains compatible with current clients, but new functionalities are added which can or should be used. Clients are expected to upgrade to new version as described in release notes of the new version.
- PATCH version is increased when there are backwards-compatible bug fixes. Current interface remains the same.

Service endpoint will have a context suffix -vMAJOR, e.g. /FiscalizationService-v1. This means that at one moment there might be several active service endpoints with different MAJOR versions but each of them will always have the latest MINOR and PATCH versions.

3.2 DATA MESSAGE CODING

All items in all data messages will only use selected characters encoded as a single byte in a standard decimal ASCII character set. The allowed decimal codes are 9, 10, 13, or 32 to 126.

UTF-8 must be used for encoding the data messages as XML documents, i.e. first line of the XML SOAP envelope will always be:

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

All XML elements of the fiscalization service are part of the same namespace, referenced in the Web service definition (WSDL).

The data format mask for individual items, which is listed along with their detailed description below, is a regular expression in the sense of the XML Schema, which defines the required syntax of the given item.

3.3 DATA MESSAGE STRUCTURE

All types of data messages have a common basic data format based on the SOAP 1.1 (Simple Object Access Protocol) protocol, i.e. application XML data structures are inserted into the body of the SOAP envelope. Unlike SOAP envelope header which remains empty.

Every request and response data message shall be signed with a private key belonging to the issuer or fiscalization service respectably. Exception to that rule are error messages (described in the chapter 0) which are not signed by the fiscalization service.

Digital signature is calculated only for the data message that resides inside SOAP envelope body element and is incorporated inside that data message as a envelop signature XML element.

3.4 REGISTER NEW TCR

Each electronic billing device should be registered on CIS in order to receive the code which represents that electronic billing device. This code is used for identification of electronic billing device in messages which are exchanged between CIS and the electronic billing device. This registration should be done only once when the electronic billing device is installed in the business unit where it is used.

Before this, taxpayer needs to be registered in Tax administration in active Registry of taxpayers. Taxpayer must also register the business unit (in application Self-care portal) in which the TCR is located prior to registration of the TCR.

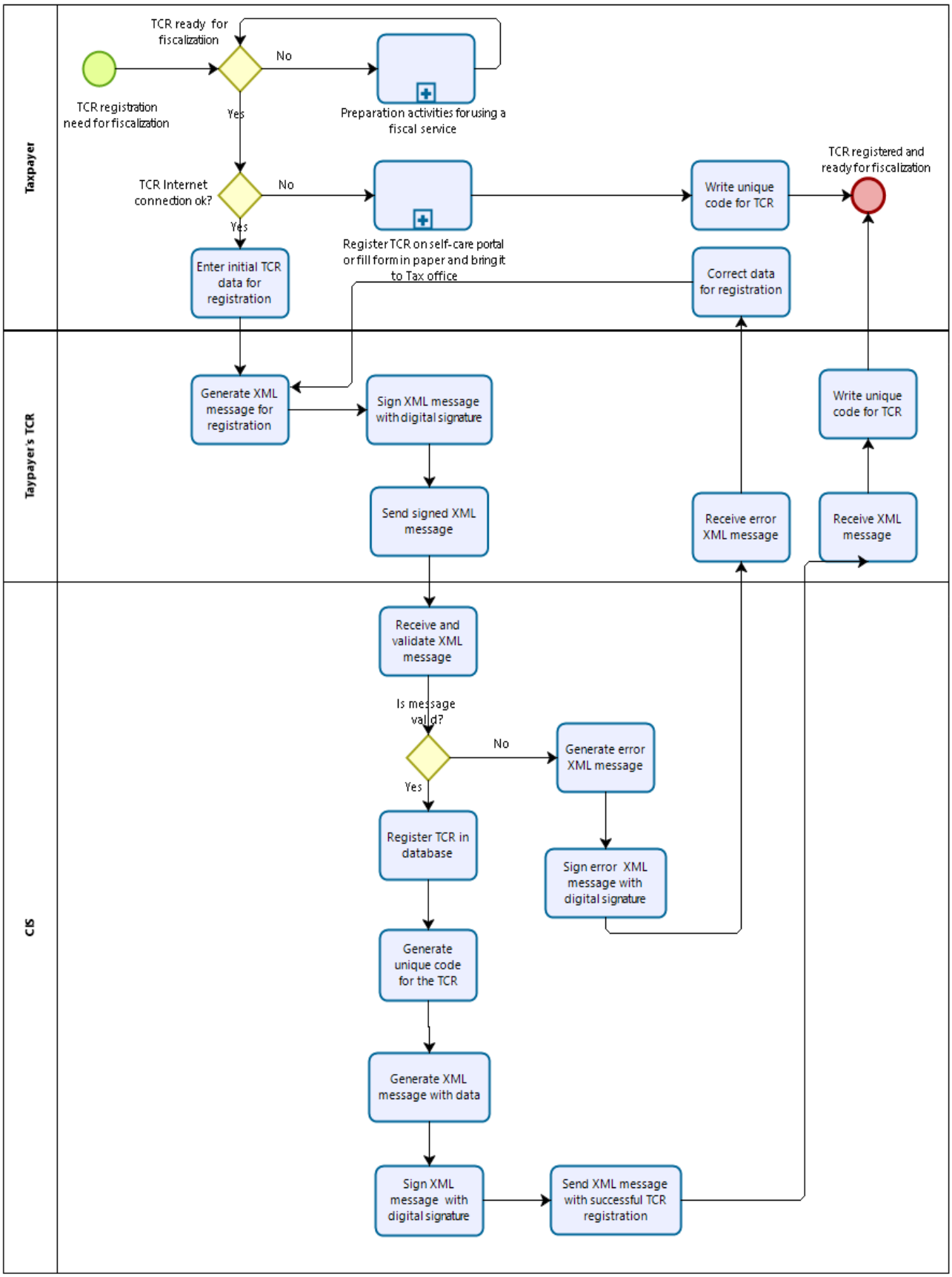

Figure 3 - Registration of taxpayer's cash register

3.4.1 REGISTER TCRREQUEST DATA MESSAGE

Name	Field type	Occurrence [Min, Max]	Description
RegisterTCRRequest	Element	[1, 1]	Root XML element representing registration of TCR
Header	Element	[1, 1]	XML element representing header
UUID	Attribute	[1, 1]	ID of the message
SendDateTime	Attribute	[1, 1]	Date and time of sending the message to the Tax administration
TCR	Element	[1, 1]	XML element representing a single TCR registration message
RegDateTime	Attribute	[1, 1]	Date and time of the TCR registration
IssuerNUIS	Attribute	[1, 1]	Taxpayer's NUIS
BusinUnit	Attribute	[1, 1]	Business unit number
TCROrdNum	Attribute	[1, 1]	Order number of the TCR
SoftNum	Attribute	[1, 1]	Number of the software used by TCR
ManufacNum	Attribute	[1, 1]	Number of the software manufacturer
Signature	Element	[1, 1]	XML element with digital signature

Table 3

3.4.1.1 Header

Element representing the header of the request data message.

3.4.1.2 Header UUID

Element generated by the TCR. It uniquely identifies the request message sent from TCR to CIS. UUID should be constructed according to the RFC4122, version 4.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 4

3.4.1.3 Header SendDateTime

Element represents date and time of sending the request message to the CIS. Date and time should be in ISO 8601 format.

Data type	dateTime
Length	23 characters
Pattern	[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}[+-][0-9]{2}:[0-9]{2}
Example	2019-01-24T22:00:58+01:00 2019-01-24T22:00:58-01:00

Table 5

3.4.1.4 TCR

Element representing a single TCR registration request.

3.4.1.5 TCR RegDateTime

Element representing registration date and time.

Data type	dateTime
Length	23 characters
Pattern	[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}[+-][0-9]{2}:[0-9]{2}
Example	2019-01-24T22:00:58+01:00 2019-01-24T22:00:58-01:00

Table 6

3.4.1.6 TCR IssuerNUIS

Element representing issuer's NUIS.

Data type	string
Length	10
Pattern	[a-zA-Z]{1}[0-9]{8}[a-zA-Z]{1}
Example	K72001008V

Table 7

3.4.1.7 TCR BusinUnit

Code (ID) of the business unit in which the TCR is registered.

Data type	string
Length	10 characters
Pattern	[a-z]{2}[0-9]{3}[a-z]{2}[0-9]{3}
Example	ab123ab123

Table 8

3.4.1.8 TCR TCROrdNum

Element representing the order number of the TCR.

Data type	integer
Pattern	([1-9][0-9]*)
Example	2

Table 9

3.4.1.9 TCR SoftNum

Code of the software used for registering TCR.

Data type	string
Length	10 characters

Pattern	[a-z]{2}[0-9]{3}[a-z]{2}[0-9]{3}
Example	ab123ab123

Table 10

3.4.1.10 TCR ManufacNum

Code of the maintainer of the software used for registering TCR.

Data type	string
Length	10 characters
Pattern	[a-z]{2}[0-9]{3}[a-z]{2}[0-9]{3}
Example	ab123ab123

Table 11

3.4.1.11 Signature

XML element stores enveloped digital signature described in the chapter 5.3.1.

3.4.2 REGISTER TCR RESPONSE DATA MESSAGE

Name	Field type	Occurrence [Min, Max]	Description
RegisterTCRResponse	Element	[1, 1]	Root XML element representing registration of TCR.
Header	Element	[1, 1]	XML element representing header…
UUID	Attribute	[1, 1]	ID of the message.
RequestUUID	Attribute	[1, 1]	UUID of the request message for which this response message was sent.
SendDateTime	Attribute	[1, 1]	Date and time of sending the message to the Tax administration.
TCRNumber	Element	[1, 1]	ID of the TCR generated by the CIS.
Signature	Element	[1, 1]	XML element with signature.

Table 12

3.4.2.1 Header

Element representing the header of the response data message.

3.4.2.2 Header UUID

Element generated by the CIS. It uniquely identifies the response message sent from CIS to TCR. UUID should be constructed according to the RFC4122, version 4.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 13

3.4.2.3 Header RequestUUID

Element generated by the TCR and referenced by the CIS. It uniquely identifies the request message for which the response message was sent to the TCR. UUID should be constructed according to the RFC4122, version 4.

Data type	String
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 14

3.4.2.4 Header SendDateTime

Element represents date and time of sending the response message to the TCR. Date and time should be in ISO 8601 format.

Data type	dateTime
Length	23 characters
Pattern	[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}[+-][0-9]{2}:[0-9]{2}
Example	2019-01-24T22:00:58+01:00 2019-01-24T22:00:58-01:00

Table 15

3.4.2.5 TCRNumber

ID of the TCR device, generated by the CIS.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 16

3.4.2.6 Signature

XML element stores enveloped digital signature described in the chapter 5.3.1.

3.4.3 MANDATORY CONTROLS

Mandatory (critical) controls shall be performed by CIS system in process of registration of new TCR. In case that the control is not passed, an error message will be returned with error code defined here.

The critical controls include the following:

Control Name	Control Description
Registered taxpayer	Taxpayer is member of active Registry of taxpayers
Registered business unit	Business unit code references active business unit of the taxpayer

Registered software maintainer	Software maintainer code referencesactive software maintainer
Registered software version	Software version code references active software version of the billing device

Table 17

3.4.4 ERROR MESSAGE

Error message is defined in chapter 0

3.4.5 EXAMPLE XML

3.4.5.1 Request XML

```
<SOAP-ENV:Envelopexmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<RegisterTCRRequestxmlns="https://eFiskalizimi.tatime.gov.al/FiscalizationService/schema"xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"Id="Request
">
<HeaderSendDateTime="2019-09-03T14:37:31+02:00"UUID="688b3a2d-bcdf-4410-b76f-6088d2792923"/>
<TCRBusinUnit="bb123bb123"IssuerNUIS="I12345678I"ManufacNum="mm123mm123"RegDateTime="2019-09-03T14:37:31+02:00"SoftNum="ss123ss123"TCROrdNum="1"/>
<Signaturexmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethodAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<SignatureMethodAlgorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
<ReferenceURI="#Request">
<Transforms>
<TransformAlgorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<TransformAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</Transforms>
<DigestMethodAlgorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<DigestValue>ymd8aMSVvaXJJ1f+z4vDLNsh0ncLySyhslvdt7EIHbc=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>m74RG/UMsv3g9NyIT2+Tfk5.....aaAdBIa23VlHIJk6PKTL6nBy7a5r0Q5K8LXDug==</SignatureValue>
<KeyInfo>
<X509Data>
<X509Certificate>MIIE6TCCAtGgAwIBAgICEA8wDQ.....uoqWsSuLmA==</X509Certificate>
</X509Data>
</KeyInfo>
</Signature>
</RegisterTCRRequest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.4.5.2 Response XML

```
<env:Envelopexmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header/>
<env:Body>
<RegisterTCRResponseId="Response"xmlns="https://eFiskalizimi.tatime.gov.al/FiscalizationService/schema"xmlns:ns0="http://www.w3.org/2000/09/xmldsi
g#">
<TCRNumber>cc123cc123</TCRNumber>
<Signaturexmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethodAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<SignatureMethodAlgorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
<ReferenceURI="#Response">
<Transforms>
<TransformAlgorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<TransformAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</Transforms>
<DigestMethodAlgorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<DigestValue>eynNl8M9tC3pGxVNEURb6MGrbw9HrSIylIso64gaIsE=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>qOgiTP5LGtELhdGOJ1FfOtilEIU.....YumiyWZKQ5o3eyRQ/kB8UJywbNMaw==</SignatureValue>
<KeyInfo>
<X509Data>
<X509Certificate>MIIFOTCCAyGgAwIBAgICEAwwDQYJ.....FbRiruyYIHYKTCznxUZu2SQ/hsah</X509Certificate>
</X509Data>
</KeyInfo>
</Signature>
</RegisterTCRResponse>
</env:Body>
</env:Envelope>
```

3.5 REGISTER TCR CASH BALANCE

Every day before registering the first invoice of the day on CIS, each TCR for handling cash transactions should register the amount of cash in the deposit. During the day, the operator can put (deposit) or take (withdraw) cash from the TCR and each of these actions should be registered.

In special cases defined by the Draft Law (when there is failure of internet connection or if the taxpayer operates in the area without internet connection), information about opening deposit or withdrawal must be stored in TCR memory and reported together with fiscalization process in defined time and in defined way. If there is failure of TCR, data of the deposit/withdrawal is written in daily book of invoices and delivered to the Tax Administration later together with all invoices that are to be fiscalized through that process.

The deposit can be 0.00.

Cash balance registration is not required in the business premises that operates only (i.e. issues only) with non-cash invoices.

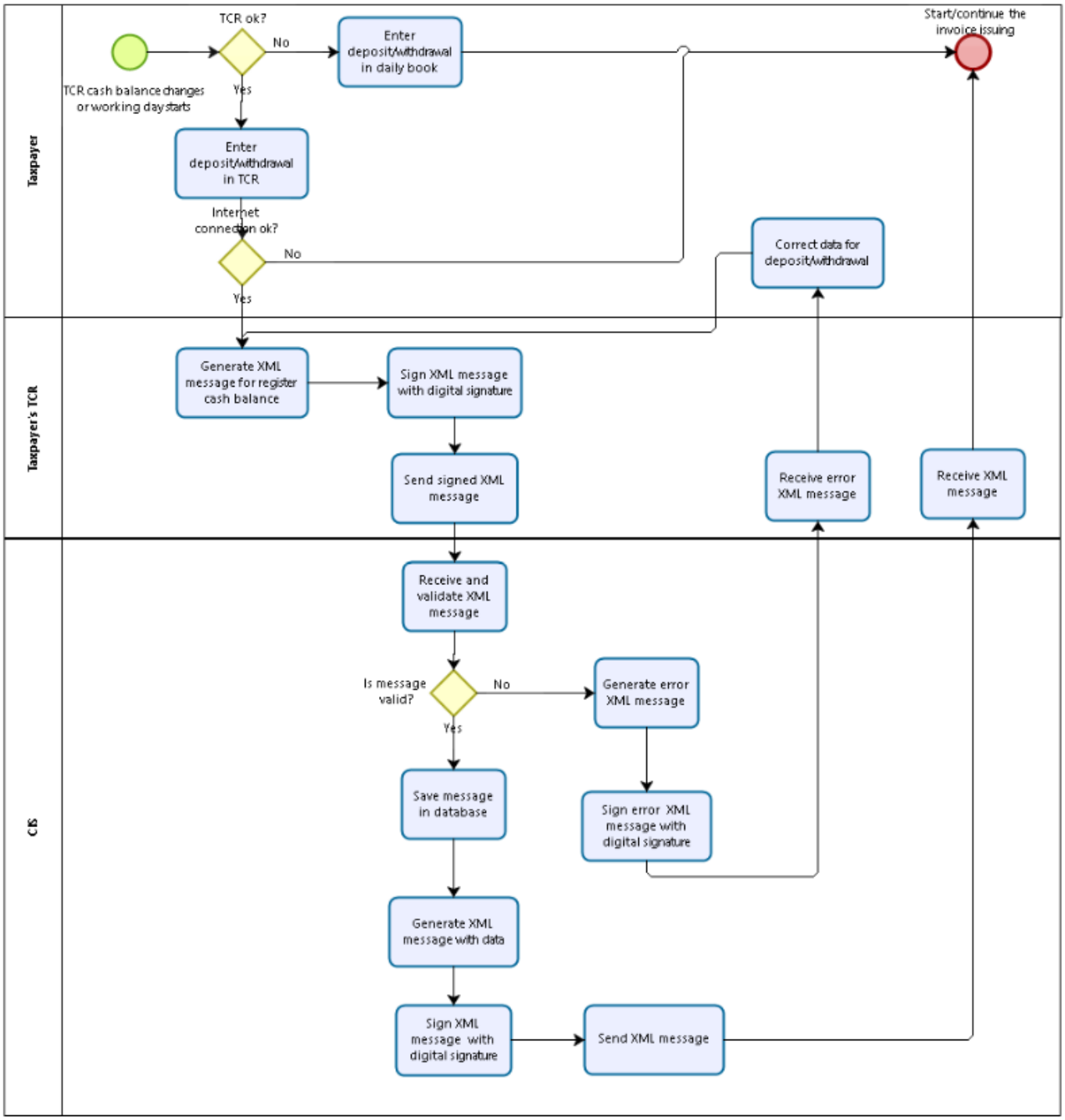

Figure 4 – Registration of the cash balance

3.5.1 REGISTER TCR CASH BALANCE REQUEST DATA MESSAGE

Name	Field type	Occurrence [Min, Max]	Description
RegisterTCRCashBalanceRequest	Element	[1, 1]	Root XML element representing registration of TCR first deposit.
Header	Element	[1, 1]	XML element representing header of the message.
UUID	Attribute	[1, 1]	UUID generated by a TCR for every register sale data message send to the CIS.
SendDateTime	Attribute	[1, 1]	Date and time of sending the message from a TCR to the CIS.
CashBalance	Element	[1, 1]	XML element representing a single cash balance request.
BalChkDatTim	Attribute	[1, 1]	Date and time when the cash balance was checked.
Operation	Attribute	[1, 1]	Operation made at the register.
CashAmt	Attribute	[1, 1]	Amount of the cash balance in the TCR.
TCRNumber	Attribute	[1, 1]	Number of the TCR for which the cash balance is registered.
IssuerNUIS	Attribute	[1, 1]	Taxpayer's NUIS.
Signature	Element	[1, 1]	XML element with signature.

Table 18

3.5.1.1 Header

Element representing the header of the request data message.

3.5.1.2 Header UUID

Element generated by the TCR. It uniquely identifies the request message sent from TCR to CIS. UUID should be constructed according to the RFC4122, version 4.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 19

3.5.1.3 Header SendDateTime

Element represents date and time of sending the request message to the CIS. Date and time should be in ISO 8601 format.

Data type	dateTime
Length	23 characters
Pattern	[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}[+-][0-9]{2}:[0-9]{2}
Example	2019-01-24T22:00:58+01:00 2019-01-24T22:00:58-01:00

Table 20

3.5.1.4 CashBalance

Element representing a singlecash balance registration.

3.5.1.5 CashBalance BalChkDatTim

Element representing date and time when the balance of cash was checked in the cash register.

Data type	dateTime
Length	23 characters
Pattern	[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}[+-][0-9]{2}:[0-9]{2}
Example	2019-01-24T22:00:58+01:00 2019-01-24T22:00:58-01:00

Table 21

3.5.1.6 CashBalance Operation

This is an XML element that represents the operation made at the cash register.

Data type	string
Constraint	Enumeration, described in the table below.
Example	Deposit

Table 22

Enumeration values for the operation are listed in the table below.

Value	Description
Balance	Current balance in the TCR.
Deposit	Amount of cash deposited into the TCR.
Credit	Amount of cash withdrawn from the TCR.

Table 23

3.5.1.7 CashBalance CashAmt

Element representing the amount of cash found in the cash register after the operation.

Data type	decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	212.12 -212.12

Table 24

3.5.1.8 CashBalance TCRNumber

Element representing the unique number of the TCR in question.

Data type	string
Length	36charaters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 25

3.5.1.9 CashBalance IssuerNUIS

Element representing issuer's NUIS.

Data type	string
Length	10
Pattern	[a-zA-Z]{1}[0-9]{8}[a-zA-Z]{1}
Example	K72001008V

Table 26

3.5.1.10 Signature

XML element stores enveloped digital signature described in the chapter 5.3.1.

3.5.2 REGISTER TCR CASH BALANCE RESPONSE DATA MESSAGE

Name	Field type	Occurrence [Min, Max]	Description
RegisterTCRCashBalanceResponse	Element	[1, 1]	Root XML element representing registration of TCR
Header	Element	[1, 1]	XML element representing headeR
UUID	Attribute	[1, 1]	ID of the message
RequestUUID	Attribute	[1, 1]	UUID of the request message for which this response message was sent
SendDateTime	Attribute	[1, 1]	Date and time of sending the message to the Tax administration
Signature	Element	[1, 1]	XML element with signature

Table 27

3.5.2.1 Header

Element representing the header of the response data message.

3.5.2.2 Header UUID

Element generated by the CIS. It uniquely identifies the response message sent from CIS to TCR. UUID should be constructed according to the RFC4122, version 4.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 28

3.5.2.3 Header RequestUUID

Element generated by the TCR and referenced by the CIS. It uniquely identifies the request message for which response message was sent to the TCR. UUID should be constructed according to the RFC4122, version 4.

Data type	String
Length	36 characters

Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 29

3.5.2.4 Header SendDateTime

Element represents date and time of sending the response message to the TCR. Date and time should be in ISO 8601 format.

Data type	dateTime
Length	23 characters
Pattern	[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}[+-][0-9]{2}:[0-9]{2}
Example	2019-01-24T22:00:58+01:00 2019-01-24T22:00:58-01:00

Table 30

3.5.2.5 Signature

XML element stores enveloped digital signature described in the chapter 5.3.1.

3.5.3 MANDATORY CONTROLS

Mandatory (critical) controls shall be performed by CIP system in process of registering the cash balance of TCR. In case that the control is not passed, an error message will be returned with error code defined here.
The critical controls include the following:

Control Name	Control Description
Issuer in active	Issuer NUIS exists in active Registry of taxpayers
Active TCR	TCR code is registered and TCR is active and it belongs to the business unit of the issuer
Check time in the past	Cash balance check / operation (deposit, withdrawal) date and time is in the past with respect to the daylight changing time

Table 31

3.5.4 ERROR MESSAGE

Error message is defined in chapter0.

3.5.5 EXAMPLE XML

3.5.5.1 Request XML

```
<SOAP-ENV:Envelopexmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<RegisterTCRCashBalanceRequestxmlns="https://eFiskalizimi.tatime.gov.al/FiscalizationService/schema"xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"
Id="Request">
<HeaderSendDateTime="2019-09-03T14:28:58+02:00"UUID="47984f42-735f-4489-999f-46b204028c61"/>
<TCRCashBalanceBalChkDatTim="2019-09-03T14:28:58+02:00"CashAmt="2000.00"IssuerNUIS="I12345678I"Operation="Balance"TCRNumber="cc123cc123"/>
<Signaturexmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethodAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<SignatureMethodAlgorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
<ReferenceURI="#Request">
<Transforms>
<TransformAlgorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<TransformAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</Transforms>
<DigestMethodAlgorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<DigestValue>I9ZhEXd6mYRWpim7js/xyzcOnn/y76kUVaa+ZUhI4V8=</DigestValue>
</Reference>
```

```
</SignedInfo>
<SignatureValue>MtasafqNnKKvNXL1AZTnOmcq6ttI1qmXB.....y+qo2rvO5Mw5G5Njgh8djmw==</SignatureValue>
<KeyInfo>
<X509Data>
<X509Certificate>MIIE6TCCAtGgAwIBAgICEA8.....uoqWsSuLmA==</X509Certificate>
</X509Data>
</KeyInfo>
</Signature>
</RegisterTCRCashBalanceRequest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.5.5.2 Response XML

```
<env:Envelopexmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header/>
<env:Body>
<RegisterTCRCashBalanceResponseId="Response"xmlns="https://eFiskalizimi.tatime.gov.al/FiscalizationService/schema"xmlns:ns0="http://www.w3.org/2000/09/xmldsig#">
<Signaturexmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethodAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<SignatureMethodAlgorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
<ReferenceURI="#Response">
<Transforms>
<TransformAlgorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<TransformAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</Transforms>
<DigestMethodAlgorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<DigestValue>rkDTQVaVKjxsYCx4YV84q/Nvous7sTu07aGdEpDMXKA=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>gs2UwjvwS/JArNzlm5c1HbC3uQ...../+J4MFXaW/jzwhkbzpt5ZGg==</SignatureValue>
<KeyInfo>
<X509Data>
<X509Certificate>MIIFOTCCAyGgCEAwQYJKoZ.....Zc/WdwruyYTZu2SQ/hsah</X509Certificate>
</X509Data>
</KeyInfo>
</Signature>
</RegisterTCRCashBalanceResponse>
</env:Body>
</env:Envelope>
```

3.6 REGISTER INVOICE

Taxpayer is due to deliver information on each invoice he issues. Information has to be delivered at the moment of issuing. Exceptionally, it can be delivered afterwards (that depends on Draft Law or other sub laws).

Data exchange process starts at the moment when the issuer is about to issue an invoice to the customer. The cash register prepares invoice data and based on that data it creates IIC. After that it prepares XML invoice request message and signs it electronically by its certificate private key (using a certificate that was issued to issuer by CIS with purpose being the implementation of fiscalization). After that the 1-way TLS communication is started and once successful it calls the service.

Central information system receives and processes request message. If the request is successfully processed, central information system prepares XML message that contains Fiscal Identification Code (FIC), which is unique for every invoice, signs it electronically with its certificate, and sends it back to the cash register.

Cash register receives answer message and checks its electronic signature. After that, cashier issues invoice receipt and hands it to the buyer.

Corrective invoice is a special type of invoice which references the original invoice in order to change some data in the original invoice, e.g. some items from the original invoice should be removed because they are returned from the buyer to the seller. Corrective invoice does not contain relative changes to the original invoice, but it contains new changed final state. E.g. in case that an item is returned then the corrective invoice contains all items as the original invoice except for the removed item.

If there were errors during the operation (invalid XML, invalid certificate or similar), central information system shows the error as XML message. If that is the case, there is no FIC so issuer will issue the invoice without FIC. The invoice issuing process must not be halted because of the error, but the issuer is obliged to correct the message request mistake and deliver it after he receives correct message.

In all situations when issuer does not get FIC for invoice he issued (loss of Internet connection, computer breakdown, central information system unavailability or similar), he is obliged to make another invoice request. Invoice is found to be properly sent and reported to the CIS once issuer gets FIC for it.

In cases when there are invoices without FIC, those should be sent again later (and in timeframe defined in laws), as invoices processed at the moment have advantage over invoices issued prior. Invoices without FIC should be delivered when traffic load is smaller or when Internet connection becomes available again.

When the taxpayer is operating in the area without internet connection, he can export invoice registration request into special file format which can be then delivered through the Self-care portal with bulk upload of invoice or brought to the Tax administration office.

Maximum time-out for machine to wait for the answer that contains FIC is set by issuer himself. Issuer needs to check the Internet connection quality and time needed for issuing one receipt so that will not affect his business. When calculating maximum time-out, issuer should count in additional two seconds (time needed for request to come in and get out of process).

Invoice registration can be verified by online application. Each invoice contains a QR-code which contains a link to the web page which displays information about the invoice, if it was successfully registered and fiscalized. Details are explained in subchapter 3.6.9.

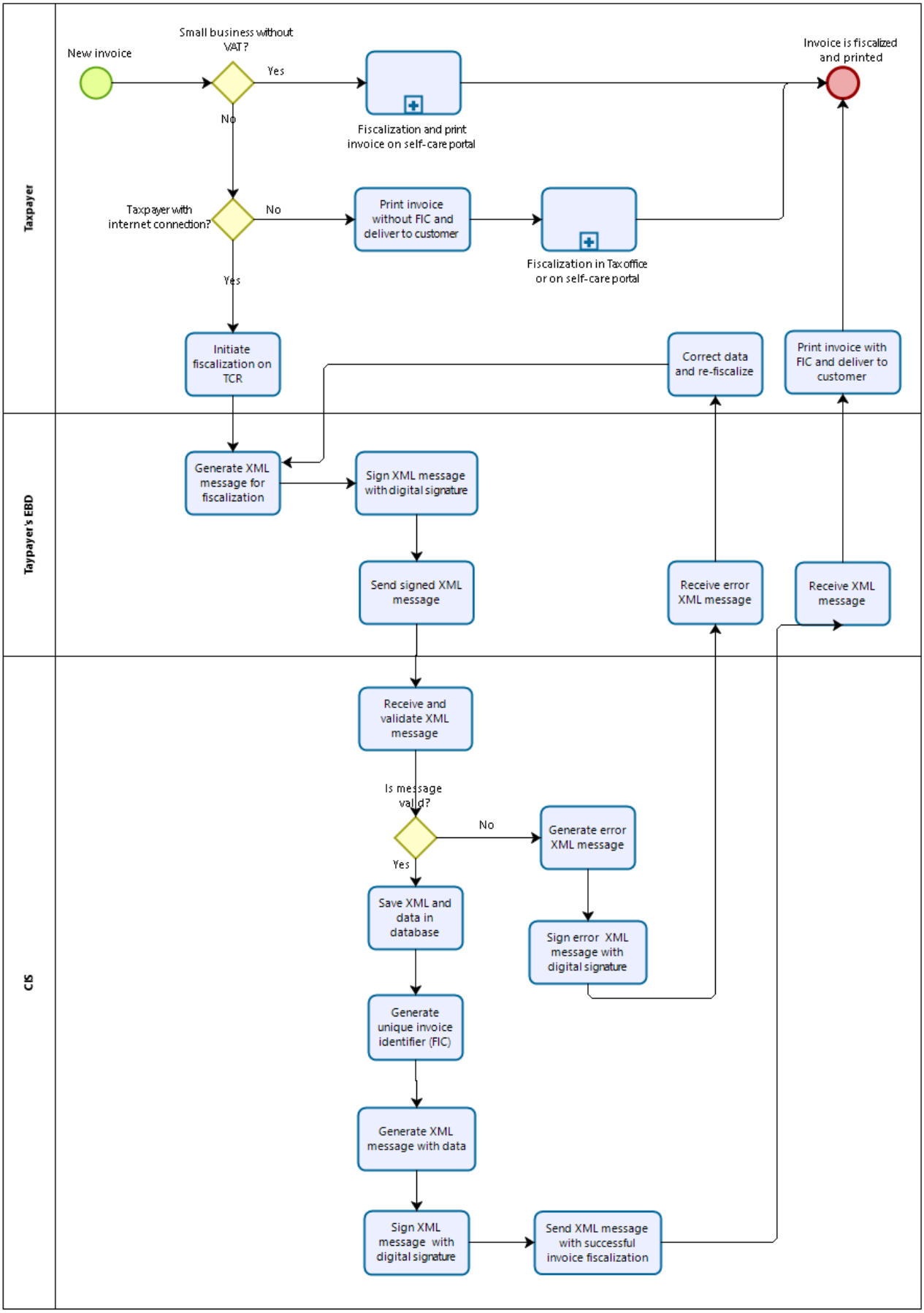

Figure 5 – Fiscalization of invoices

3.6.1 UML PROCESS DIAGRAM

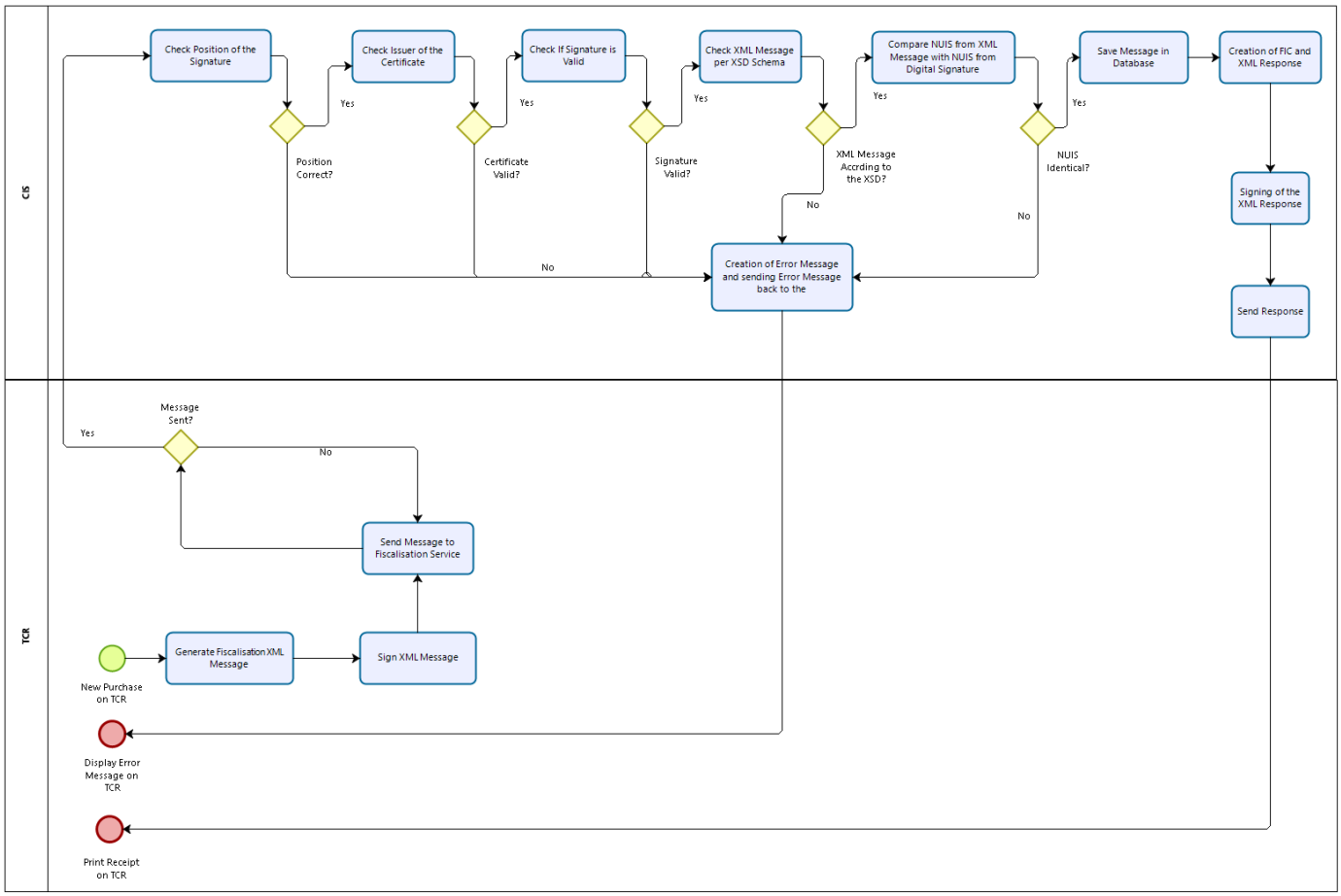

Figure 6 - XML validations

CIS processes the registration of the invoice in the following way:

- TLS communication is initiated between the invoice issuer and CIS
  - In case that certificate provided by the invoice issuer is invalid, communication protocol fails
- CIS checks the size of the message
  - If the message is larger than 200kB, an error message is sent in the response and no further processing is done.
- Further controls are performed. For each passed control, process continues to the next control. If the control fails, no more controls are performed, and error message is prepared. Following controls are performed:
  - SOAP message is a valid XML document
  - SOAP message is structured according to the defined schema
  - Certificate in the signature of the message is checked
  - Signature of the message is checked
  - IIC is verified
  - Date and time written in the message is checked
  - Is issuer in the VAT obligated
- If one of the controls failed, an error message will be sent in the reply to the request message with the error code related to the nature of the error.
- If all of the controls have passed successfully, register invoice response message will be sent.
  - FIC code is generated.

- Response message is prepared with FIC included.
- Response message is signed, and signature is put into the message.

- Invoice data is inserted into the database.
  - Data includes request SOAP message, response SOAP message (or error message) and key elements of the invoice. Key elements are these elements of the invoice:
    - IIC
    - FIC
    - Date Time Created
    - Issuer Tax Number
    - Operator
    - Business Unit
    - Cash Register
    - Total Price
    - Total VAT Amount
    - Payment Method
    - Type Of Invoice (invoice type)
    - TCR Reference

- Prepared response message is sent back to the taxpayer

3.6.2 Register invoice request data message

This is the request message sent by the issuer of the invoice to CIS.

Name	Field type	Occurrence [Min, Max]	Description
RegisterInvoiceRequest	Root	[1, 1]	Root XML element representing register sale message.
Header	Element	[1, 1]	XML element representing header of the invoice containing data about the message (request) sent.
UUID	Attribute	[1, 1]	UUID generated by a TCR for every register sale data message send to the CIS.
SendDateTime	Attribute	[1, 1]	Date and time of sending the register sale data message from a TCR to the CIS.
Invoice	Element	[1, 1]	XML element representing a single invoice.
TypeOfInv	Attribute	[1, 1]	Type of the invoice (cash, non-cash).
SelfIssuing	Attribute	[1, 1]	XML element saying if the invoice is self-issued or not.
TypeOfSelfIss	Attribute	[0, 1]	Entered only if invoice is self-issued.
DateTimeCreated	Attribute	[1, 1]	Date and time when the invoice is created.
InvNum	Attribute	[1, 1]	Invoice number.
InvOrdNum	Attribute	[1, 1]	Invoice order number.
CashRegister	Attribute	[1, 1]	Cash register number aka CRN.
IssuerInVAT	Attribute	[1, 1]	Issuer is in VAT obligation
TaxFreeAmt	Attribute	[0, 1]	Amount on goods that are not under any tax.
MarkUpAmt	Attribute	[0, 1]	Mark-up amount.
GoodsExport	Attribute	[0, 1]	Amount of goods for export from the Republic of Albania.
TotPriceWoVAT	Attribute	[1, 1]	Total price of the invoice excluding VAT.
TotVATAmt	Attribute	[1, 1]	Total VAT amount of the invoice.

TotPrice	Attribute	[1, 1]	Total price of all items including taxes and discounts
PaymentMeth	Attribute	[1, 1]	Method of payment
OperatorCode	Attribute	[1, 1]	Reference to the operator who is operating on TCR.
BusinUnit	Attribute	[1, 1]	Business unit number.
SoftNum	Attribute	[1, 1]	Software number.
IIC	Attribute	[1, 1]	Issuer's invoice code calculated as MD5 hash from IICSignature attribute.
IICReference	Attribute	[0,1]	Reference to the original invoice (added if original invoice had to be changed)
IICSignature	Attribute	[1, 1]	Signed issuer's invoice code concatenated parameters.
IsSubseqDeliv	Attribute	[1, 1]	Is the invoice delivered subsequently?
ReverseCharge	Attribute	[1, 1]	If true, the buyer is obliged to pay the VAT directly to the Tax administration.
BadDebt	Attribute	[0, 1]	If the invoice is not payable, it will get this mark.
Issuer	Element	[1, 1]	Issuer's tax number along with other data in sub-elements.
NUIS	Attribute	[1, 1]	Issuer's NUIS.
Name	Attribute	[1, 1]	Issuer's name.
Address	Attribute	[1, 1]	Issuer's address.
Town	Attribute	[1, 1]	Issuer's town.
Country	Attribute	[1, 1]	Issuer's country.
Buyer	Element	[0, 1]	Buyer's tax number along with other data in sub-elements.
NUIS	Attribute	[1, 1]	Issuer's NUIS.
Name	Attribute	[1, 1]	Buyer's name.
Address	Attribute	[1, 1]	Buyer's address.
Town	Attribute	[1, 1]	Buyer's town.
Country	Attribute	[1, 1]	Buyer's country.
Items	Element	[1, 1]	XML element representing list of invoice items.
I	Element	[1, 1000]	XML element representing one item.
N	Attribute	[1, 1]	Name of the item (goods or services).
C	Attribute	[0, 1]	Code of the item from the barcode or similar representation.
U	Attribute	[1, 1]	What is the item's unit of measure (piece, weight measure, length measure, etc.)
Q	Attribute	[1, 1]	Amount or number (quantity) of items
UP	Attribute	[1, 1]	Unit price
R	Attribute	[1, 1]	Percentage of the rebate.
RR	Attribute	[0, 1]	Is rebate reducing base amount?
PB	Attribute	[1, 1]	Total price of goods and services before the tax
VR	Attribute	[1, 1]	Rate of value added tax
VA	Attribute	[1, 1]	Amount of value added tax for goods and services
PA	Attribute	[1, 1]	Total price of goods after the tax and applying discounts
SameTaxItems	Element	[1, 1]	List of the items that go under same tax rate.
Item	Element	[1, 20]	XML element representing one same tax item.
NumOfItems	Attribute	[1, 1]	Number of items.
PriceBefVAT	Attribute	[1, 1]	Price before VAT.
VATRate	Attribute	[1, 1]	VAT rate.
VATAmt	Attribute	[1, 1]	VAT amount.
ConsTaxItems	Element	[0, 1]	Special, consumption taxes (alcohol, beverages).
Item	Element	[1, 20]	XML element representing one cons tax item.

				NumOfItems	Attribute	[1, 1]	Number of items under consumption tax.
				PriceBefConsTax	Attribute	[1, 1]	Price before adding consumption tax.
				ConsTaxRate	Attribute	[1, 1]	Rate of the consumption tax.
				ConsTaxAmt	Attribute	[1, 1]	Amount of consumption tax.
	Signature				Element	[1, 1]	XML element representing signature for the invoice.

Table 32

3.6.2.1 Header

XML element representing header of the request data message.

3.6.2.2 Header UUID

Element generated by the TCR. It uniquely identifies the request message sent from TCR to CIS. UUID should be constructed according to the RFC4122, version 4.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

3.6.2.3 Header SendDateTime

Element represents date and time of sending the request message to the CIS. Date and time should be in ISO 8601 format.

Data type	dateTime
Length	23 characters
Pattern	[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}[+-][0-9]{2}:[0-9]{2}
Example	2019-01-24T22:00:58+01:00 2019-01-24T22:00:58-01:00

Table 33

3.6.2.4 Invoice

XML element representing a single invoice.

3.6.2.5 Invoice TypeOfInv

Type of the invoice represents the type of invoice cash or non-cash.

Data type	string
Values	Enumeration, described in the table below.
Example	C

Table 34

Following table shows the list of allowed values inside TypeOfInv attribute.

Value	Description
C	Cash

N	Non-cash

Table 35

3.6.2.6 Invoice SelfIssuing

The invoice can be issued by buyer himself.

Data type	boolean
Values	true, false
Example	true

Table 36

3.6.2.7 Invoice TypeOfSelfIss

This element shows the type of self-issuing.

Data type	string
Values	Enumeration, described in the table below.
Example	S

Table 37

Following table shows the list of allowed values inside TypeOfSelfIss attribute.

Value	Description
S	The previous agreement between the parties.
P	Purchase from domestic farmers.
U	Purchase of services from abroad.
O	Other

Table 38

3.6.2.8 Invoice DateTimeCreated

Date and Time of the invoice creation.

Data type	string
Length	23 characters
Pattern	[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}[+-][0-9]{2}:[0-9]{2}
Example	2019-01-24T22:00:58+01:00 2019-01-24T22:00:58-01:00

Table 39

3.6.2.9 Invoice InvNum

Invoice number printed on the receipt.

Data type	string
Max length	100
Example	1/2019/36

Table 40

3.6.2.10 Invoice InvOrdNum

Number in a sequence is assigned to each new invoice so that the invoices can be counted. The sequence is reset at the beginning of each year.

Data type	integer
Pattern	([1-9][0-9]*)
Example	9934

Table 41

3.6.2.11 Invoice CashRegister

Code (ID) of the cash register on which the invoice is issued.

Data type	string
Length	10 characters
Pattern	[a-z]{2}[0-9]{3}[a-z]{2}[0-9]{3}
Example	ab123ab123

Table 42

3.6.2.12 Invoice IssuerInVAT

Is taxpayer in the VAT system.

Data type	boolean
Values	true, false
Example	true

Table 43

3.6.2.13 Invoice TaxFreeAmt

Amount on items that are tax free.

Data type	decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	17.24

Table 44

3.6.2.14 Invoice MarkUpAmt

Amount of the mark-up on the invoice.

Data type	decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	23.10

Table 45

3.6.2.15 Invoice GoodsExport

Total price of delivery of exported goods. There is no VAT on the invoice.

Data type	decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	246.00

Table 46

3.6.2.16 Invoice TotPriceWoVAT

Total price without any taxes.

Data type	decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	212.12

Table 47

3.6.2.17 Invoice TotVATAmt

Total amount of VAT which needs to be paid for all groups of items listed in this invoice.

Data type	decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	242.23

Table 48

3.6.2.18 Invoice TotPrice

Total price which needs to be paid by the customer for all groups of items listed in this invoice.

Data type	decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	212.12

Table 49

3.6.2.19 Invoice PaymentMeth

Method of payment.

Data type	string
Constraint	Enumeration, described in the tablebelow.
Example	N

Table 50

Enumeration values for the method of payment are listed in table below.

Value	Description
N	Cash. Bills and coins

K	Card
C	Cheque, bank check
T	Bank transaction using the transaction account
O	Other non-cash payments

Table 51

3.6.2.20 Invoice OperatorCode

Reference to the operator who is operating on TCR. Value represents code of the operator.

Data type	string
Length	10 characters
Pattern	[a-z]{2}[0-9]{3}[a-z]{2}[0-9]{3}
Example	ab123ab123

Table 52

3.6.2.21 Invoice BusinUnit

Code (ID) of the business unit in which the invoice is issued.

Data type	string
Length	10 characters
Pattern	[a-z]{2}[0-9]{3}[a-z]{2}[0-9]{3}
Example	ab123ab123

Table 53

3.6.2.22 Invoice SoftNum

Number of the software used for invoice issuing.

Data type	string
Length	10 characters
Pattern	[a-z]{2}[0-9]{3}[a-z]{2}[0-9]{3}
Example	ab123ab123

Table 54

3.6.2.23 Invoice IIC

Issuers invoice code which is generated by the cash register of the issuer of the invoice. This is a unique code for every invoice. The code is formed by concatenating the fields, signing with issuer's private key and calculating MD5 hash. Further description can be found in the chapter5.3.2.3.

Data type	string
Max length	32
Pattern	[0-9a-fA-F]{32}
Example	C701FB4839E7D2C3D8DBC81BBAC06164 c701fb4839e7d2c3d8dbc81bbac06164

Table 55

3.6.2.24 Invoice IICReference

Reference to the invoice IIC of the original invoice. It is entered only if original invoice was changed.

Data type	string
Max length	32
Pattern	[0-9a-fA-F]{32}
Example	C701FB4839E7D2C3D8DBC81BBAC06164 c701fb4839e7d2c3d8dbc81bbac06164

Table 56

3.6.2.25 Invoice IICSignature

Signed issuer's invoice code concatenated parameters. Further description can be found in the chapter 5.3.2.2.

Data type	string
Max length	512
Pattern	[0-9a-fA-F]{512}
Example	B2C218486302EC553EE1AB9124E1A14705742E870E8872EF34E63617AB252E189ACDF7A3E3F5C82061FFFF8AC2826A5588596A8807F648410899B6193F77F4BDCDFA87553A62079A2EF9E6E6F0B8DA1038968D2FCB920B580EBF33ACEEDFEA0DAA78067F916ADC5D278CC237EFD53A6156EABAFBE98A8F3CE99E854818822FA20C0FF46E5B3805264BBCD085F0A8A9BD503A1304E9202D7304FF93541FB7FAA4629EE0BD7ED566F610DCD047721AEAA828DFECA651087CDE5AF95C125793D4CD8E83B801DE171335A866D7E31F1473BF0C93EBFD994326C0FE97ACB8DA722F788EA27B8D9E15E8E7B6EF772AB7534060F2BCAF1C3E82645235C9D1857B0790C2

Table 57

3.6.2.26 Invoice IsSubseqDeliv

Element that says if the invoice will be delivered subsequently.

Data type	boolean
Values	true, false
Example	true

Table 58

3.6.2.27 Invoice ReverseCharge

Buyer is obliged to pay taxes by himself rather than issuer does it for him.

Data type	boolean
Values	true, false
Example	true

Table 59

3.6.2.28 Invoice BadDebt

If the invoice is marked as unpayable, it gets "bad debt" note.

Data type	boolean
Values	true, false
Example	true

Table 60

3.6.2.29 Invoice Issuer

XML element representing an issuer of the invoice.

3.6.2.30 Invoice Issuer NUIS

Issuer's NUIS.

Data type	string
Length	10
Pattern	[a-zA-Z]{1}[0-9]{8}[a-zA-Z]{1}
Example	K72001008V

Table 61

3.6.2.31 Invoice Issuer Name

Issuer's name.

Data type	string
Length	100 characters
Example	Name Surname

Table 62

3.6.2.32 Invoice Issuer Address

Issuer's address.

Data type	string
Length	100 characters
Example	Plaza Tirana 1

Table 63

3.6.2.33 Invoice Issuer Town

Issuer's town.

Data type	string
Length	100 characters
Example	Tirana

Table 64

3.6.2.34 Invoice Issuer Country

Issuer's country.

Data type	string
Length	100 characters
Example	Albania

Table 65

3.6.2.35 Invoice Buyer

XML element representing a buyer that purchase goods.

3.6.2.36 Invoice Buyer NUIS

Buyer's NUIS.

Data type	string
Max length	10
Pattern	[a-zA-Z]{1}[0-9]{8}[a-zA-Z]{1}
Example	K72001008V

Table 66

3.6.2.37 Invoice Buyer Name

Buyer's name.

Data type	string
Length	100 characters
Example	Name Surname

Table 67

3.6.2.38 Invoice Buyer Address

Buyer's address.

Data type	string
Length	100 characters
Example	Street Name 888

3.6.2.39 Invoice Buyer Town

Buyer's town.

Data type	string
Length	100 characters
Example	Tirana

Table 68

3.6.2.40 Invoice Buyer Country

Buyer's country.

Data type	string
Length	100 characters
Example	Albania

Table 69

3.6.2.41 Invoice Items

XML element representing list of invoice items (goods or services). Items which are the same should be grouped as one item (one XML element called “Item”) with the appropriate amount (sum of grouped items).

3.6.2.42 Invoice Items I (Item)

XML element representing a single item on the list of items.

3.6.2.43 Invoice Items I N (Item Name)

Name of the item.

Data type	String
Max length	50
Example	Coca-cola 1.5L

Table 70

3.6.2.44 Invoice Items I C (Item Code)

Code of the item from the barcode or similar representation. It helps in identification of the product (item).

Data type	String
Max length	50
Example	978020137962

Table 71

3.6.2.45 Invoice Items I U (Item Unit of measure)

Unit of measure for specific item – piece, weight, length...

Data type	String
Max length	50
Example	Kg

Table 72

3.6.2.46 Invoice Items I Q (Item Quantity)

Amount or number (quantity) of items.

Data type	Double
Constraints	Must be positive number, at least 0.001.
Example	3.500 0.375

Table 73

3.6.2.47 Invoice Items I UP (Item Unit Price)

Price of the one item (unit price).

Data type	Decimal

Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	3.50

Table 74

3.6.2.48 Invoice Items I R (Item Rebate)

Rebate percentage.

Data type	Decimal
Example	12 33.17

Table 75

3.6.2.49 Invoice Items I RR (Item Rebate Reducing base price)

Is rebate reducing base price?

Data type	Boolean
Values	true, false
Example	True

Table 76

3.6.2.50 Invoice Items I PB (Item Price Before VAT)

Price before VAT for the items in this group of items. This is not the unit price of the item. It is the unit price multiplied by the quantity of items.

Data type	Decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	134.34

Table 77

3.6.2.51 Invoice Items I VR (Item VAT Rate)

Rate of value added tax expressed as percentage.

Data type	Decimal
Pattern	0|(0|[1-9][0-9]*)\.[0-9]{2}
Example	3.50

Table 78

3.6.2.52 Invoice Items I VA (Item VAT Amount)

Amount of value added tax for the items in this group of items.

Data type	Decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	3.50

Table 79

3.6.2.53 Invoice Items I PA (Item Price After applying VAT)

Price after applying VAT for the items in this group of items.

Data type	Decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	3.50

Table 80

3.6.2.54 Invoice SameTaxItems

XML element representing list of invoice items (goods or services) that are under same VAT rate. All items of same VAT rate are grouped together.

3.6.2.55 Invoice SameTaxItems Item NumOfItems

Number of items of same tax rate.

Data type	Integer
Pattern	([1-9][0-9]*)
Example	2

Table 81

3.6.2.56 Invoice SameTaxItems Item PriceBefVAT

Price of the item before VAT

Data type	decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	12.20

Table 82

3.6.2.57 Invoice SameTaxItems Item VATRate

VAT rate applied on items from one group, expressed as percentage.

Data type	String
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	10.00

Table 83

3.6.2.58 Invoice SameTaxItems Item VATAmt

VAT amount for items from same tax group.

Data type	decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	246.00

Table 84

3.6.2.59 Invoice ConsTaxItems

XML element representing list of invoice items (goods or services) that are under consumption tax, like alcohol, beverages, etc).

3.6.2.60 Invoice ConsTaxItems Item NumOfItems

Number of items under the consumption tax.

Data type	integer
Pattern	([1-9][0-9]*)
Example	2

Table 85

3.6.2.61 Invoice ConsTaxItems Item PriceBefConsTax

Price of the item before consumption tax.

Data type	decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	12.20

Table 86

3.6.2.62 Invoice ConsTaxItems Item ConsTaxRate

Consumption tax rate.

Data type	String
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	10.00

Table 87

3.6.2.63 Invoice ConsTaxItems Item ConsTaxAmt

Consumption tax amount.

Data type	decimal
Pattern	0|(0|-?[1-9][0-9]*)\.[0-9]{2}
Example	246.00

Table 88

3.6.2.64 Signature

XML element stores enveloped digital signature described in the chapter 5.3.1.

3.6.3 REGISTER INVOICE RESPONSE DATA MESSAGE

Name	Field type	Occurrence [Min, Max]	Description
RegisterInvoiceResponse	Element	[1, 1]	Root XML element representing register invoice response message.
Header	Element	[1, 1]	XML element representing generic message data about the response sent.
UUID	Attribute	[1, 1]	UUID generated by a CIS for every register invoice response data message send to the TCR.
RequestUUID	Attribute	[1, 1]	UUID of the request message for which this response message was sent.
SendDateTime	Attribute	[1, 1]	Date and time of sending the register invoice response data message from a CIS to the TCR.
FIC	Element	[1, 1]	CIS generated verification code that can be used to uniquely identify registered invoice.
Signature	Element	[1, 1]	XML element with signature.

Table 89

3.6.3.1 Header

XML element representing header of the response data message.

3.6.3.2 Header UUID

Element generated by the CIS for every message sent to the TCR. It uniquely identifies the message sent to the TCR. UUID should be constructed according to the RFC4122, version 4.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 90

3.6.3.3 Header RequestUUID

Element generated by the TCR and referenced by the CIS. It uniquely identifies the request message for which response message was sent to the TCR. UUID should be constructed according to the RFC4122, version 4.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 91

3.6.3.4 Header SendDateTime

Element represents date and time of sending the response message to the TCR. Date and time should be in ISO 8601 format.

Data type	dateTime
Length	23 characters

Pattern	[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}[+-][0-9]{2}:[0-9]{2}
Example	2019-01-24T22:00:58+01:00 2019-01-24T22:00:58-01:00

Table 92

3.6.3.5 FIC

Element represents unique number of invoice generated by the CIS, under which the requested invoice is registered.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 93

3.6.3.6 Signature

XML element stores enveloped digital signature described in the chapter 5.3.1.

3.6.4 ERROR MESSAGE

Error message is defined in chapter 0.

3.6.5 MANDATORY CONTROLS

Mandatory (critical) controls shall be performed on received registered invoices data messages in the CIP system.

When any of the critical controls return a failure, the registered invoice data message shall not be accepted, and FIC shall not be issued. Upon identifying a critical error, CIS will return an error data message containing the error's numeric code and its text description (see chapter 3.6.4). When errors which the system can interpret as a cyber-attack are identified, the system does not send any response to the client (the taxpayer's cash register).

The critical controls include the following:

Control Name	Control Description
Check size of the data	Size should not exceed 150kB
XML formatting	UTF-8 is required
XML structure validation	Check of the individual registered invoice data message's in XML against the XSD schema (*.xsd). XSD schema contains an exact definition of the data and format structure for the individual data items and a check of presence of individual items
Certificate validation	Check that certificate is not expired. Check that the certificate is issued by the trusting CA. Check that the identification number in the certificate corresponds to the invoice issuer identification number (tax number) in the XML message. Check that certificate is not listed in CRL.
Electronic signature check	Check that the hash of the message calculated by CIS corresponds to the hash listed in the message.

	Check that the signature corresponds to the hash of the message and to the public key of the certificate.
Check date and time sent	Check if the element SendDateTime is within 90 minutes of the current moment. If message was sent in a period more than 90 minutes, e.g. because of the internet connection loss, that message has to be marked as described in invoice registration message structure (IsSubseqDeliv).
Self-invoicing	If it is true, then buyer fields need to be entered because it is the one who is issuing the invoice.
Pay in cash invoice limit	Check the amount of invoice when payed in cash. It can only be payed by cash if its amount is at most ALL 150,000. The amount is subject to change.

Table 94

3.6.6 OPTIONAL CONTROLS

Optional controls are not performed at the moment of registration the invoice but are instead postponed for later processing of the invoices. Errors detected here will be available to taxpayer over "self-care portal" and to tax officials over CPCM. There will be other controls implemented as part of anomaly management which are not listed here.

Description should be available in multiple languages.

Control Name	Control Description
Valid TIN of invoice issuer	TIN is valid and in the register of active taxpayers
Valid TCR	Code of TCR should be valid and TCR should be active.
Valid business unit	Code of the business unit should be valid, and this business unit should be active. Business unit needs to belong to the listed business units of the invoice issuer. TCR should belong to listed business unit of the invoice issuer.
Valid operator code	Operator code is valid in the moment of invoice issuing and the operator is assigned to the invoice issuer
Valid software code	Software code is valid at the time of invoice issuing. If TCR code is present, then software code of the invoice needs to be the same as the software code assigned to TCR.
Calculation of price	Recalculate the total prices and group of prices for each tax rate. In the error description, correct price and wrong price should be stated.
Corrective invoice reference exists	In case that the invoice is corrective invoice, the referenced invoice exists
Corrective invoice reference of the same taxpayer	In case that the invoice is corrective invoice, both referenced (original) invoice and corrective invoiceare issued by the same taxpayer
Check IIC	Issuer's invoice code is checked against the defined format and procedure of generating IIC. CIS checks if fields which are listed in the request message correspond to the values used for creation of IIC
TCR of Self-care portal	TCR of Self-care portal can only be used with certificate of Self-care portal
Software code of Self-care portal	Software "Self-care portal" can only be used with certificate of Self-care portal

Table 95

3.6.7 EXAMPLE XML

3.6.7.1 Request XML

```
<SOAP-ENV:Envelopexmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<RegisterInvoiceRequestxmlns="https://eFiskalizimi.tatime.gov.al/FiscalizationService/schema"xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"Id="Req
uest">
<HeaderSendDateTime="2019-09-03T14:06:34+02:00"UUID="2e2147bc-a147-442c-a1cc-7949fc469c4f"/>
<InvoiceBadDebt="false"BusinUnit="bb123bb123"CashRegister="cc123cc123"DateTimeCreated="2019-09-
03T14:06:34+02:00"IIC="A3113F3CB0C2AFE6F6CE200E8910B17B"IICSignature="6EB69E96C6420150A0335F779C5C487BF3B845B65FC4025BF4C05B89519174B593D39C39FF3F
CC28F4A740319415490E0A56EA8EF669238A06F369FB75FD28DE607525ED2992D3D8F6F4B7BBD7BB024229A3F25E5E2D8F120C391DDAB58960FAA88CFA16C5DFF7F44A2B502665509E
A49E2D76DFBB0B073594768E4292F4CC51DE4780E86205879FDB3A53F32E6DDDCC3647339D86A055E1FD668119992B377FDE50BBF36A1777449A7CC8A857700700A8D318F4A802E770
11E4693493E19F1F38DF904AF046DE61EB4D76E5308CBBE0EB74CF58D50CF794692144CF2F5DDCBF3EBB5DABD9CFCBABE164717DAECB194CC26940B97AC806815E91E52AE44695A2"I
nvNum="31/2019/cc123cc123"InvOrdNum="31"IsSubseqDeliv="false"IssuerInVAT="true"OperatorCode="oo123oo123"PaymentMeth="N"ReverseCharge="false"SelfIs
suing="false"SoftNum="ss123ss123"TotPrice="20.00"TotPriceWoVAT="16.00"TotVATAmt="4.00"TypeOfInv="C">
```

```
<IssuerAddress="Issueraddress"Country="Issuercountry"NUIS="I12345678I"Name="Issuername"Town="Issuertown"/>
<Items>
<IC="501234567890"N="Item name"PA="20.00"PB="16.00"Q="1.0"R="0"RR="true"U="piece"UP="20.00"VA="4.00"VR="25.00"/>
</Items>
<SameTaxItems>
<ItemNumOfItems="1"PriceBefVAT="16.00"VATAmt="4.00"VATRate="25.00"/>
</SameTaxItems>
</Invoice>
<Signaturexmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethodAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<SignatureMethodAlgorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
<ReferenceURI="#Request">
<Transforms>
<TransformAlgorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<TransformAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</Transforms>
<DigestMethodAlgorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<DigestValue>pF7eLrR8kyp7n3y6...g0LeKahXKKPb6R68fZWeM=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>xpkKInX2sUJOZfX9BZeCo+fkwScqV5XgAQ....ZPDWxl6JYwmJUyQxf4vct1XU+eATWJ</SignatureValue>
<KeyInfo>
<X509Data>
<X509Certificate>MIIE6TCCAtGgAwIBAgICEA8wDQYJKoZIhvcNAQELBQ......uoqWsSuLmA==</X509Certificate>
</X509Data>
</KeyInfo>
</Signature>
</RegisterInvoiceRequest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.6.7.2 Response XML

```
<env:Envelopexmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header/>
<env:Body>
<RegisterInvoiceResponseId="Response"xmlns="https://eFiskalizimi.tatime.gov.al/FiscalizationService/schema"xmlns:ns0="http://www.w3.org/2000/09/xmldsig#">
<FIC>0f757c75-24ee-45b8-8a9e-f27538d46eb3</FIC>
<Signaturexmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethodAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<SignatureMethodAlgorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
<ReferenceURI="#Response">
<Transforms>
<TransformAlgorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<TransformAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</Transforms>
<DigestMethodAlgorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<DigestValue>grEWs31wPjI48DheEBWcD4m+8kjix01n027Ai2Yon9E=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>F0cKff2swcpZBgKmtlNtbXhoLITyq.............5JkwnWWqqjYBIlaStA3sFwd2Fx4kVXaOnkbD0w==</SignatureValue>
<KeyInfo>
<X509Data>
<X509Certificate>MIIFOTCCAyGgAwIBAgICEAQAwW.........FbRi7FZKdQZc/WdwruyYIHYKTCznxUZu2SQ/hsah</X509Certificate>
</X509Data>
</KeyInfo>
</Signature>
</RegisterInvoiceResponse>
</env:Body>
</env:Envelope>
```

3.6.8 BULK UPLOAD OF INVOICES

When the taxpayer is operating in the area where there is no internet connection, he can use the alternative way of invoice registration. Invoice registration requests should be exported into files in a special format. The files can then be uploaded on Self-care portal which will register the invoices. Responses from the registration service will be generated and the taxpayer will download the responses in files and bring them back to the TCR. Instead of using the Self-care portal web application, the taxpayer can bring the files on USB flash drive to the local Tax administration office where the officer will do the same process as in the Self-care portal.

What taxpayer does in that case is that he uses TCR to creates a file for each of those invoices, and files should be named in `<yyyyMMddHHmmSS>_<IIC>_request.xml` format.

When those .xml files (WS messages) are created, they will be created in the way that only main part of the XML, the one that contains data, will be preserved. It will omit WS header and WS envelope from the message. The file should contain only content of envelope body element.

Then each of those files should be saved in a zipped (archived) folder, named `<randomNumbersAndLetters>_request.zip`. It is recommended to put a prefix on a file which corresponds to a code of the electronic billing device in order to easily identify it later. Size of the ZIP archive should not exceed 15 MB. Therefore it is recommended that at most 100 invoices is inside one ZIP archive since one invoice can be up to 150kB.

After that the taxpayer transfers that archive to the USB flash drive or other transferable media and takes it to the place where there is an active internet connection. After logging in to Self-care portal using the certificate, taxpayer imports ZIP archive with invoices represented as XML files. Self-care portal should extract XML files from the ZIP archive, form a web service message request and send it to fiscalization service. The fiscalization service will make a response which Self-care portal will transform into XML file by extracting the content of SOAP envelope body element. XML file representing the response should be named `<yyyyMMddHHmmSS>_<IIC>_response.xml`. XML files will be compressed in a ZIP archive with corresponding name `<randomNumbersAndLetters>_response.zip`where `<randomNumbersAndLetters>`is the same as in the ZIP archive with request XML files. That means that each request in XML file named `<yyyyMMddHHmmSS>_<IIC>_request.xml` shall be paired with response in XML file `<yyyyMMddHHmmSS>_<IIC>_response.xml`,and each archive containing XML request named `<randomNumbersAndLetters>_request.zip` will be paired with archive named `<randomNumbersAndLetters>_response.zip` containing XML responses.

If there are errors, the Self Care Portal will notify the user about it.

Application should log to application log name of the incoming ZIP archive, names of the extracted request files, name of the generated response files, name of the generated ZIP archive, TIN of the user.

3.6.8.1 XML example

As mentioned in previous chapter, when there is bulk upload of the invoices, the XML file will be created in a way that only main part of it – the content of the body (without body element) - the one containing the message – will be preserved, while other parts will be omitted. Here is the example of it (SOAP envelope elements are removed and in the example they are strikethrough):


```
<soapenv:Envelopexmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"xmlns:sch="http://alimc.neos.hr/FiscalizationService/schema"xmlns:xd="ht
tp://www.w3.org/2000/09/xmldsig#">
<soapenv:Header/>
<soapenv:Body>
<RegisterInvoiceRequestxmlns="http://alimc.neos.hr/FiscalizationService/schema"xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"Id="Request">
<HeaderSendDateTime="2019-06-14T11:00:42+02:00"UUID="9f4459da-e558-4cbc-bbff-217667096451"/>
<InvoiceBadDebt="false"BusinUnit="07790411-b967-454e-8e7c-a8154d719020"CashRegister="f5e480b0-ccc6-4363-a8eb-ae4079b7309b"DateTimeCreated="2019-
06-
14T11:00:42+02:00"GoodsExport="9.91"IIC="093FB373E719C5CDFEF23E5C3D415B7C"IICSignature="24C7727E95E2D4398FA9715BC9FE481A940C470EFBD8ED11D464E713CA
D26696585FA263C1405C572D6AA100434676C1C5B4848B16EBCFBAA0D9C50CC5B8CB728524160636DBDB5BFBE1CF85447A8584A207F7832131D513DE154DE4B56CC5FBDCEBD5CDA447
6A2AD1AFB1B80EF176E29C1D51EB784DEFE23C3004679FA1BFA207431C524273F91C65550F161FB8B3E0FFB41C34462B830BF748BFF3707ECF8FB6FA9205532596249ADA950066A0AC
A523AA4C4BABCCF328806CC40D98ECE4068205C281EFB6C8666A5513171081476B1E1836833603A032397181BEE37009749D77C6FFDD8442F1E85CC4354B9AD26E52237D2113F78022
11E464B216C151D0"InvNum="InvNum"InvOrdNum="9952"IsSubseqDeliv="false"IssuerInVAT="true"MarkUpAmt="10.99"OperatorCode="K72001008V"PaymentMeth="C"Re
verseCharge="false"SelfIssuing="true"SoftNum="2191a92e-b3b1-44f5-a2eb-
84abc817264f"TaxFreeAmt="10.91"TotPrice="99.01"TotPriceWoVAT="10.01"TotVATAmt="1.91"TypeOfInv="N"TypeOfSelfIss="O">
<IssuerAddress="Address"Country="Country"NUIS="K72001009V"Name="Name"Town="Town"/>
<BuyerAddress="Address"Country="Country"NUIS="K72001007V"Name="Name"Town="Town"/>
<Items>
<ItemCode="123451"IsRbtRed="true"Name="Kola1"PriceAftVAT="10.91"PriceBefVAT="10.91"Quantity="1.0"Rbt="1.0"Unit="kg"UnitPrice="10.91"VATAmt="10.91"
VatRATE="10.91"/>
<ItemCode="123452"IsRbtRed="true"Name="Kola2"PriceAftVAT="10.91"PriceBefVAT="10.91"Quantity="1.0"Rbt="1.0"Unit="kg"UnitPrice="10.91"VATAmt="10.91"
VatRATE="10.91"/>
<ItemCode="123453"IsRbtRed="true"Name="Kola3"PriceAftVAT="10.91"PriceBefVAT="10.91"Quantity="1.0"Rbt="1.0"Unit="kg"UnitPrice="10.91"VATAmt="10.91"
VatRATE="10.91"/>
<ItemCode="123454"IsRbtRed="true"Name="Kola4"PriceAftVAT="10.91"PriceBefVAT="10.91"Quantity="1.0"Rbt="1.0"Unit="kg"UnitPrice="10.91"VATAmt="10.91"
VatRATE="10.91"/>
<ItemCode="123455"IsRbtRed="true"Name="Kola5"PriceAftVAT="10.91"PriceBefVAT="10.91"Quantity="1.0"Rbt="1.0"Unit="kg"UnitPrice="10.91"VATAmt="10.91"
VatRATE="10.91"/>
<ItemCode="123456"IsRbtRed="true"Name="Kola6"PriceAftVAT="10.91"PriceBefVAT="10.91"Quantity="1.0"Rbt="1.0"Unit="kg"UnitPrice="10.91"VATAmt="10.91"
VatRATE="10.91"/>
<ItemCode="123457"IsRbtRed="true"Name="Kola7"PriceAftVAT="10.91"PriceBefVAT="10.91"Quantity="1.0"Rbt="1.0"Unit="kg"UnitPrice="10.91"VATAmt="10.91"
VatRATE="10.91"/>
<ItemCode="123458"IsRbtRed="true"Name="Kola8"PriceAftVAT="10.91"PriceBefVAT="10.91"Quantity="1.0"Rbt="1.0"Unit="kg"UnitPrice="10.91"VATAmt="10.91"
VatRATE="10.91"/>
<ItemCode="123459"IsRbtRed="true"Name="Kola9"PriceAftVAT="10.91"PriceBefVAT="10.91"Quantity="1.0"Rbt="1.0"Unit="kg"UnitPrice="10.91"VATAmt="10.91"
VatRATE="10.91"/>
<ItemCode="1234510"IsRbtRed="true"Name="Kola10"PriceAftVAT="10.91"PriceBefVAT="10.91"Quantity="1.0"Rbt="1.0"Unit="kg"UnitPrice="10.91"VATAmt="10.9
1"VatRATE="10.91"/>
</Items>
<SameTaxItemsNumOfItems="1"PriceBefVAT="10.91"VATAmt="10.91"VATRate="10.91"/>
```

```
<ConsTaxItemsConsTaxAmt="10.91"ConsTaxRate="10.91"NumOfItems="1"PriceBefConsTax="10.91"/>
</Invoice>
<Signaturexmlns="http://www.w3.org/2000/09/xmldsig#"><SignedInfo><CanonicalizationMethodAlgorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/><SignatureMethodAlgorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-
sha256"/><ReferenceURI="#Request"><Transforms><TransformAlgorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/><TransformAlgorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/></Transforms><DigestMethodAlgorithm="http://www.w3.org/2001/04/xmlenc#sha256"/><DigestValue>EIWEiHIM+ppzdWXr5tbh1TbyMGQZ7cEIzGOeJuXBuTQ=</
DigestValue></Reference></SignedInfo><SignatureValue>cvUsypF7Ui0J86sul4VMwNH+OEKEzt0TcqJmL30cV2/uaDeqai1AH2IGW+NaHgZcs2mvRnbtep7y
xpVF/B4YgaO2V8fIJW5UBMbMzNreS4bpfU34boPfz1ZZXHU9ujXwdkQJqOsBD9LuTPn8ye7MM9js
wkNb3rdREuqmakaNZMHyIGfNgOpQPpNhZhA++LoQBMUTsMYsQiuS2ci833dQcIEHPaf9MT8pMVZz
elhOEQxCep86pczlT0N+LkihdmzDCjUV1TR2xywGjNqGUYGMjZk0cBJF7fzh9Vl9soPXeCqVt5PF
9kJs1Yy1Zfcs1/0PsVieFjiEQVx1E6GK1F2NVQ==</SignatureValue><KeyInfo><X509Data><X509Certificate>MIIE6TCCAtGgAwIBAgICEA8wDQYJKoZIhvcNAQELBQAwWDELMAkGA
1UEBhMCSFIxEDAOBgNVBAgM
B0Nyb2F0aWExFDASBgNVBAoMC05FT1MgZC5vLm8uMSEwHwYDVQQDDBhORU9TIGQuby5vLiBJbnRl
cm1lZGlhdGUwHhcNMTkwMjIyMTMxMTE1WhcNMjcwMjIwMTMxMTE1WjBPMQswCQYDVQQGEwJIUjEQ
MA4GA1UECAwHQ3JvYXRpYTEUMBIGA1UECgwLRGVtbyBmaXNrYWwxGDAWBgNVBAMMD2Zpc2thbDEu
bmVvcy5ocjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAOszkeSLTWByTVTMZY2bXfyk
FMhiKM25yCfCFqnBHrJLUVeRRR74kGbMdg6cnlUE7vynpfV4fmejmerebAA832J5MAq7G9BYU/tN
ZtjyB2iFGYVbXKaZ9IMTVn4M1uLE708q2JOjtSscprMf63JP1kUAdZqevGrvJoBjMxrpB3y+1pOI
zlOVQ9x5UrQcdTI1VvrKDeRlC/61hmmLp+mbJ/WmGBLo2zoSryg8vJzrTSEuvXanXaYmQJL3YfbG
HRdRJYF2YXZmbBlETJujnHj1D2TvxxSqu0ZSuryMV1SOpfTurVrfhrobpfHtMdlU8G03acBuBaXX
1E6lAyunLMBzoU0CAwEAAaOBxTCBwjAJBgNVHRMEAjAAMBEGCWCGSAGG+EIBAQQEAwIFoDAzBglg
hkgBhvhCAQ0EJhYkT3BlblNTTCBHZW5lcmF0ZWQgQ2xpZW50IENlcnRpZmljYXRlMB0GA1UdDgQW
BBQP5MUP/iZc6yxbyFYV9quMDLq5uzAfBgNVHSMEGDAWgBS7a9gudUvZTTp3chYtQqKDNcDqOzAO
BgNVHQ8BAf8EBAMCBeAwHQYDVR0lBBYwFAYIKwYBBQUHAwIGCCsGAQUFBwMEMA0GCSqGSIb3DQEB
CwUAA4ICAQAJAQ+fa3FtbxgHpKEPagG9+hq8oCzdb5M/WfEMtLKUQuwm1y4nVf9C+ohnOBtAO9A1
UBDp4FNPGuPRjlSB7jy1uDyXj50aCLMnvfBnypcuANdxMO7LlYX38wLiTiSdVyjMi2w6xNj49aXA
PWexjwUFecCK270FB9yJ7yF8wc+zcZ9m1UBbmKvDQgPqHHhdsCBbYmpTtF5HfvBXqZBrqzLXfU/P
PdKHxzwi8337eUQGaIUznlS87oIDJYEmAVBLG90VU52B/eGlPBXKqR3tgrhRHiJ5V/ANPP76n3Uz
lSXqTR8ujK3dUxQPszylYqYXcWH5gBwIAmikbZEoVXGuzd3M+NYPATQdkOLKYuPZKCRKLNkCZ8Xi
Yz7DHLEjYTCvxdT7HLAw4LNctgJhIFBUa22DhlY2At2rznXoFyeDtndQG6KR6XwL+EHPbD6FZPRE
mBcJpTAhqKH67z4dVTNFjfAGYZkBb7C7g4LDGpdFBzIj7GZ2HS1GJuTUZ2uZ6T31hmEZDESG6v70
xHZFn8NElrZ/ZGYNWDh7XOYB7CmutL+ldydbOmpXnvlv11mkX84mC5o19s8XqFHKd9YLVOCnZlhh
U7AUm6XBIztPtlt0O4KpCZaubOeyrCWjM+1XKiigNagaXUjrriif/j5a1LAq5sMhZeH0MWEUfLuL
uoqWsSuLmA==</X509Certificate></X509Data></KeyInfo></Signature></RegisterInvoiceRequest>
</soapenv:Body>
</soapenv:Envelope>
```


3.6.9 CHECKING INVOICE REGISTRATION IN WEB APPLICATION “INVOICE CHECK”

Each invoice should contain a QR-code which has a link. The link leads to the web application “Invoice check” and displays information about the invoice if the invoice is successfully registered or instructsuser to report an issue if it is not registered within the required timeframe.

QR-code contains a URL with special query string which identifies the invoice. First part of the URL is fixed and contains the protocol, hostname and path (described in document Verify invoice user manual https://efiskalizimi-app-test.tatime.gov.al/invoice-check/#/verify) followed by the query parameters which are described in the following table:

Query parameter	Description	Sample value
iic	Invoice IIC (issuer’s identification number)	EA26D5BE7F45827026108F825A8A512B
tin	Taxpayer identification number (issuer’s TIN)	L91806031N
crtd	Date and time when the invoice was created. Value is displayed in special format “yyyy-mm-ddThh:mi:ss” where *yyyy*, *mm* is month, *dd* is day of the month, *T* is fixed value, *hh*are hours in 24-hours format, *mi* are minutes, *ss* are seconds.	2019-09-26T13:50:13
ord	Invoice order number	6
bu	Code of the business unit	bg517kw842
cr	Code of the TCR	xb131ap287
sw	Code of the software installed on the TCR	gz434bv927
prc	Total price of the invoice	2354.84

Example URL encoded in a QR-code with sample values from the table above is:

https://efiskalizimi-app-test.tatime.gov.al/invoice-check/#/verify?iic=EA26D5BE7F45827026108F825A8A512B&tin=L91806031N&crtd=2019-09-26T13:50:13&ord=6&bu=bg517kw842&cr=xb131ap287&sw=gz434bv927&prc=199.00

Example QR-code which is formed from the URL above is:

Figure 7 - Sample QR code

3.7 WAREHOUSE TRANSFER NOTE

Warehouse transfer note is a document that is sent when goods are transferred between warehouses of the same owner (company) or between warehouse and business unit where the goods are sold. That means that every movement of goods on the territory of the Republic of Albania will be recorded and the Tax administration will know about it.

Process itself starts with the request for the new note. Fiscalization is initiated and an XML message is created on the taxpayer's electronic billing device and then signed with a digital certificate. Message is then sent to the CIS. If it is valid, it is saved in the database and its IIC is generated. XML message with data is then created and signed and then sent back to the electronic billing device where it is printed out.

If the message is not valid when CIS receives it, it will be sent back and refiscalized.

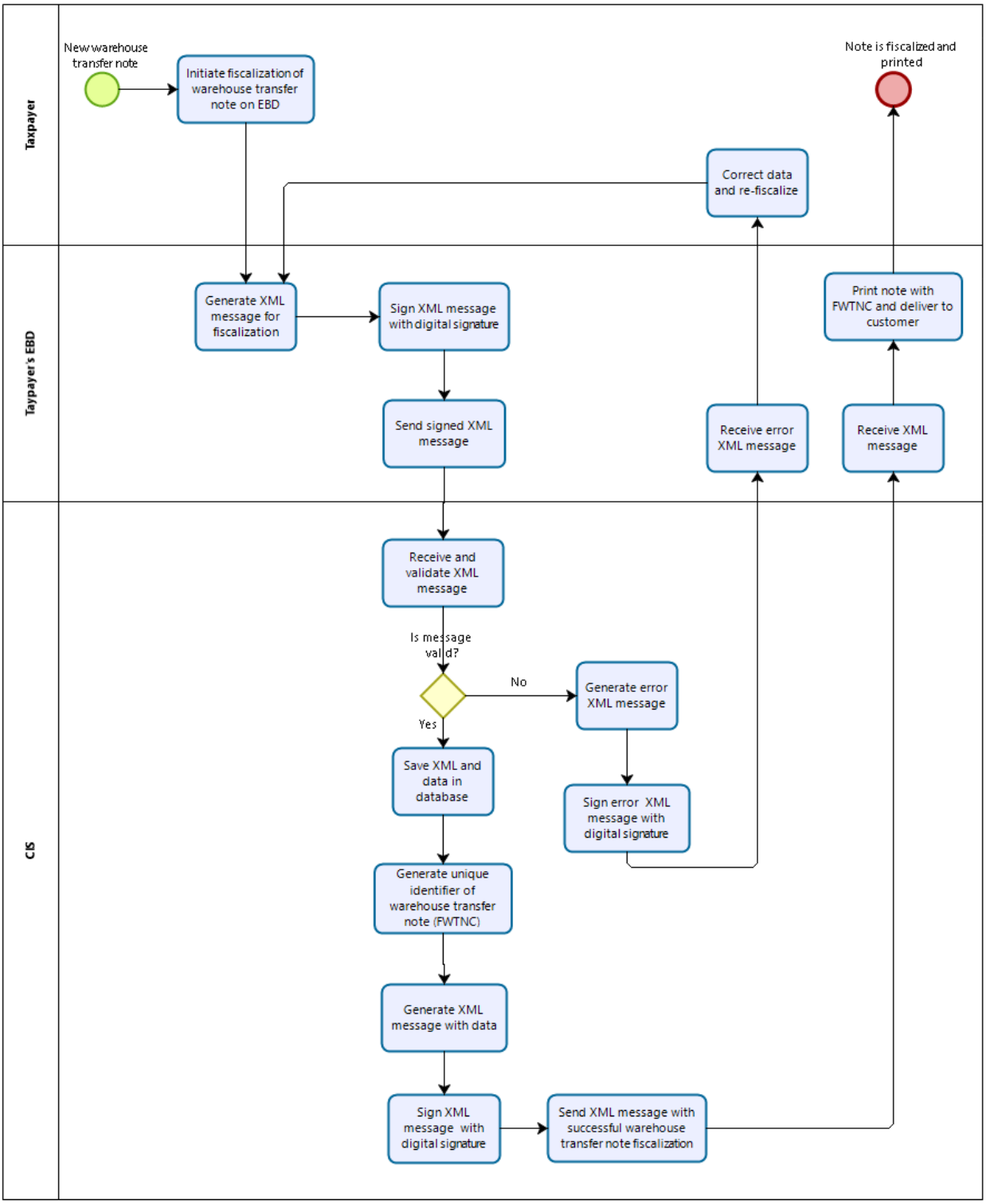

Figure 8 – Fiscalization of warehouse transfer note

3.7.1 WAREHOUSE TRANSFER NOTE REQUEST MESSAGE

Name	Field type	Occurrence [Min, Max]	Description
WTNRequest	Root	[1, 1]	Root XML element representing warehouse transfer note request message.
Header	Element	[1, 1]	XML element representing header of the note containing data about the messagesent.
UUID	Attribute	[1, 1]	UUID generated by the TCR.
SendDateTime	Attribute	[1, 1]	Date and time of sending the message from a TCR to the CIS.
Note	Element	[1, 1]	XML element representing header of the note containing data about the message sent.
DateTimeCreated	Attribute	[1,1]	Date and time of creation of the note.
WTNNum	Attribute	[1,1]	Number of the warehouse transfer note.
OperatorCode	Attribute	[1,1]	Operator code.
BusinUnit	Attribute	[1,1]	Business unit number.
SoftNum	Attribute	[1,1]	Software number.
StartAddr	Attribute	[1,1]	Address of the starting point of transportation.
StartCity	Attribute	[1,1]	City of the starting point of transportation.
DestinAddr	Attribute	[1,1]	Address of destination.
DestinCity	Attribute	[1,1]	City of destination.
WTNIC	Attribute	[1,1]	Warehouse transfer note identification code. Protection code of the issuer of the note (NSLFSH)
IsAfterDel	Attribute	[1,1]	Record of afterwards note delivery.
TransDate	Attribute	[1,1]	Date of transportation.
CarrierId	Attribute	[0,1]	Unique ID of the carrier.
VehPlates	Attribute	[1,1]	Carrier's vehicle plates number.
Issuer	Element	[1,1]	XML element representing the issuer of the note.
NUIS	Attribute	[1,1]	Taxpayer's NUIS.
Name	Attribute	[1,1]	Taxpayer's name.
Items	Element	[1,1]	XML element representing list of items.
I	Attribute	[1,1000]	Name of the item (goods or services).
N	Attribute	[1,1]	Name of the item (goods or services).
C	Attribute	[0,1]	Code of the item from the barcode or similar representation.
U	Attribute	[1,1]	What is the item's unit of measure (piece, weight measure, length measure…).
Q	Attribute	[1,1]	Amount or number (quantity) of items

Table 96

3.7.1.1 Header

XML element representing the header of the message.

3.7.1.2 Header UUID

Element generated by the CIS for every message sent to the TCR. It uniquely identifies the message sent to the TCR. UUID should be constructed according to the RFC4122, version 4.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 97

3.7.1.3 Header SendDateTime

Element represents date and time of sending the response message to the TCR. Date and time should be in ISO 8601 format.

Data type	dateTime
Length	23 characters
Pattern	[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}[+-][0-9]{2}:[0-9]{2}
Example	2019-01-24T22:00:58+01:00 2019-01-24T22:00:58-01:00

Table 98

3.7.1.4 Note

XML root element representing the note.

3.7.1.5 Note Issuer

XML element representing the issuer of the note.

3.7.1.6 Note Issuer NUIS

NUIS of the note issuer.

Data type	String
Length	10 characters
Pattern	[a-zA-Z]{1}[0-9]{8}[a-zA-Z]{1}
Example	K72001008V

Table 99

3.7.1.7 Note Issuer Name

Name of the note issuer.

Data type	String
Length	100 characters
Example	Name Surname

Table 100

3.7.1.8 Note DateTimeCreated

Date and time of the note creation.

Data type	dateTime
Length	23 characters
Pattern	[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}[+-][0-9]{2}:[0-9]{2}
Example	2019-01-24T22:00:58+01:00 2019-01-24T22:00:58-01:00

Table 101

3.7.1.9 Note WTNNum

Unique identifying number of the warehouse transfer note. It consists of ordinal number of the note and calendar year, without leading zero.

Data type	String
Max length	20 characters
Pattern	[1-9][0-9]{1,11}
Example	322019 90292019

Table 102

3.7.1.10 Note OperatorCode

Reference to the operator who is operating on TCR. Value represents code of the operator.

Data type	string
Length	10 characters
Pattern	[a-z]{2}[0-9]{3}[a-z]{2}[0-9]{3}
Example	ab123ab123

Table 103

3.7.1.11 Note BusinUnit

Code (ID) of the business unit in which the note is issued.

Data type	string
Length	10 characters
Pattern	[a-z]{2}[0-9]{3}[a-z]{2}[0-9]{3}
Example	ab123ab123

Table 104

3.7.1.12 Note SoftNum

Number of the software used for invoice issuing.

Data type	string
Length	10 characters
Pattern	[a-z]{2}[0-9]{3}[a-z]{2}[0-9]{3}
Example	ab123ab123

Table 105

3.7.1.13 Note StartAddr

Starting address of the transportation.

Data type	string
Length	100 characters
Example	Street Name 888

Table 106

3.7.1.14 Note StartCity

City where the transportation started in.

Data type	string
Length	100 characters
Example	Tirana

Table 107

3.7.1.15 Note DestinAddr

Destination address of the transportation.

Data type	string
Length	100 characters
Example	Street Name 888

Table 108

3.7.1.16 Note DestinCity

City where the transportation will end.

Data type	string
Length	100 characters
Example	Tirana

Table 109

3.7.1.17 Note WTNIC

Warehouse transfer note identification code. Protection code of the issuer of the warehouse transfer note.

Data type	string
MaxLength	32 characters
Pattern	[0-9a-fA-F]{32}
Example	C701FB4839E7D2C3D8DBC81BBAC06164

Table 110

3.7.1.18 Note IsAfterDel

Record of afterwards transfer note delivery. If the note is delivered afterwards, the value is true, if it is not, the value is false.

Data type	boolean
Values	true, false
Example	true

Table 111

3.7.1.19 Note TransDate

Date of transporation of the note.

Data type	dateTime
Length	23 characters
Pattern	[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}[+-][0-9]{2}:[0-9]{2}
Example	2019-01-24T22:00:58+01:00 2019-01-24T22:00:58-01:00

Table 112

3.7.1.20 Note CarrierId

ID of the carrier of the note.

Data type	String
Length	50 characters
Pattern	[a-zA-Z0-9]+
Example	???

Table 113

3.7.1.21 Note VehPlates

Plates of the vehicle that will carry the note.

Data type	String
Length	30 characters
Pattern	[a-zA-Z0-9]+
Example	AA 000 AA

Table 114

3.7.1.22 Note Items

XML element representing a list of items of the note.

3.7.1.23 Note Items I (Item)

XML element representing a single item on the list of items.

3.7.1.24 Note Items I N (Item Name)

Name of the item.

Data type	String
Max length	50 characters
Example	Coca-cola 1.5L

Table 115

3.7.1.25 Note Items I C (Item Code)

Code of the item from the barcode or similar representation. It helps in identification of the product (item).

Data type	String
Max length	50 characters
Example	978020137962

Table 116

3.7.1.26 Note Items I U (Item Unit of measure)

Unit of measure for specific item – piece, weight, length...

Data type	String
Max length	50 characters
Example	Kg

Table 117

3.7.1.27 Note Items I Q (Item Quantity)

Amount or number (quantity) of item.

Data type	double
Constraints	Must be positive number, at least 0.001.
Example	3.500 0.375

Table 118

3.7.2 WAREHOUSE TRANSFER NOTE RESPONSE MESSAGE

Name	Field type	Occurrence [Min, Max]	Description
WtnResponse	Element	[1, 1]	Root XML element representing warehouse transfer note response message.
Header	Element	[1, 1]	XML element representing generic message data about the response sent.
UUID	Attribute	[1, 1]	UUID generated by a CIS for every note response data message send to the TCR.
RequestUUID	Attribute	[1, 1]	UUID of the request message for which this response message was sent.
SendDateTime	Attribute	[1, 1]	Date and time of sending the note response data message from a CIS to the TCR.
FWTNC	Element	[1, 1]	Fiscal warehouse transfer note code. CIS generated verification code that can be used to uniquely identify registered note.
Signature	Element	[1, 1]	XML element with signature.

Table 119

3.7.2.1 Header

XML element representing header of the response data message.

3.7.2.2 Header UUID

Element generated by the CIS for every message sent to the TCR. It uniquely identifies the message sent to the TCR. UUID should be constructed according to the RFC4122, version 4.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 120

3.7.2.3 Header RequestUUID

Element generated by the TCR and referenced by the CIS. It uniquely identifies the request message for which response message was sent to the TCR. UUID should be constructed according to the RFC4122, version 4.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 121

3.7.2.4 Header SendDateTime

Element represents date and time of sending the response message to the TCR. Date and time should be in ISO 8601 format.

Data type	dateTime
Length	23 characters

Pattern	[0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}[+-][0-9]{2}:[0-9]{2}
Example	2019-01-24T22:00:58+01:00 2019-01-24T22:00:58-01:00

Table 122

3.7.2.5 FWTNC

Fiscal warehouse transfer note code. Element represents unique number generated by the CIS.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 123

3.7.2.6 Signature

XML element stores enveloped digital signature described in the chapter 5.3.1

3.7.3 ERROR MESSAGE

Error message is defined in chapter 0.

3.7.4 EXAMPLE XML

3.7.4.1 Request XML

```
<SOAP-ENV:Envelopexmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<RegisterWTNRequestxmlns="https://eFiskalizimi.tatime.gov.al/FiscalizationService/schema"xmlns:ns2="http://www.w3.org/2000/09/xmldsig#"Id="Request
">
<HeaderSendDateTime="2019-09-03T14:19:01+02:00"UUID="42db3af5-0d9f-4dea-95b4-4b947ab8d8e7"/>
<WTNBusinUnit="bb123bb123"DateTimeCreated="2019-09-03T14:19:01+02:00"DestinAddr="Destination address"DestinCity="Destination
city"IsAfterDel="false"OperatorCode="oo123oo123"SoftNum="ss123ss123"StartAddr="Start address"StartCity="Start city"TransDate="2019-09-
03T14:19:01+02:00"VehPlates="AA0000AA"WTNIC="5C5E58580D0A24E1F7A5E5E011929511"WTNICSignature="82D69C38206D133D9DCF59C02B0F1BC66898DCB9C0501ED3253B
C57893BDC1367278935A8E2C8A8155E039D1C65B82109CF8D6C448F5A0369AAA0C398DA07903581 2D3954B549A05CDC9397AA24792280272DD1DC73B5D27BB4E4E795B611E63500EEA
D9F7E5A45DF7CBBE1CAD0569FB396540E7D1DC42BD005503F0F68786B81B01FAD69160D1E7E75B15F64ED4FA58B776E0F61B74619845681BEC8F8D433945CF6387B311EB03AB42730C
F75FF4277A3762DFEF97FA441C5EE00BFD1403E41A59D94F057BB2DA48074A498F3B935567D6DC4EC34E36242C979ED37FEAE8EBD4E3B02C98A4619E2319126CEB55D8731CC031177A
9B07B1E9DCF414DF5FCA48"WTNNum="12345678901">
<IssuerNUIS="I12345678I"Name="Issuername"/>
<Items>
<IC="501234567890"N="Itemname"Q="1.0"U="piece"/>
</Items>
</WTN>
<Signaturexmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethodAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<SignatureMethodAlgorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
<ReferenceURI="#Request">
<Transforms>
<TransformAlgorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<TransformAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</Transforms>
<DigestMethodAlgorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<DigestValue>aNa0r4RW04BlsAciTUrWHGW9PADskl2op315BAQ0hTg=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>4/j/d4j/5xCJ2YUP+XTC6li0B94.....KESd38NT5+puArBcNgLYIjLx/dh6Q==</SignatureValue>
<KeyInfo>
<X509Data>
<X509Certificate>MIIE6TCCAtGgAwIBAgICEA8wDQYJKoZIhvcNA.....uoqWsSuLmA==</X509Certificate>
</X509Data>
</KeyInfo>
</Signature>
</RegisterWTNRequest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.7.4.2 Response XML

```
<env:Envelopexmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header/>
<env:Body>
<RegisterWTNResponseId="Response"xmlns="https://eFiskalizimi.tatime.gov.al/FiscalizationService/schema"xmlns:ns0="http://www.w3.org/2000/09/xmldsig#">
<FWTNC>32d4e940-d118-48d2-9c0a-03f2c60c6890</FWTNC>
<Signaturexmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethodAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
<SignatureMethodAlgorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
<ReferenceURI="#Response">
<Transforms>
<TransformAlgorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
<TransformAlgorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
</Transforms>
<DigestMethodAlgorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
<DigestValue>i3ppi6ppQBd1Rju0rVwdOm/u3d1wDHVjOqcF1BtksTw=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>Ni4cix4j7KjYpqASQJrTe24GhmlDB225GKqRm.....................3tmCyP5AUi2y2PfMBpZ9kH65e/4VBc7JQ==</SignatureValue>
<KeyInfo>
<X509Data>
<X509Certificate>MIIFOTCCAyGgAwIBAgICEAwwDQYJKoZIhvc.................FbRi7FZKdQZc/WdwruyYIHYKQ/hsah</X509Certificate>
</X509Data>
</KeyInfo>
</Signature>
</RegisterWTNResponse>
</env:Body>
</env:Envelope>
```

3.8 ERROR MESSAGES

In case of an error in the process of the request message, an error message is sent as a reply by CIS. Error messages share the same general format which is based on SOAP fault message version 1.1 and extended with the code XML element which represents numeric error code.

3.8.1 XML FORMAT

Name	Occurrence [Min, Max]	Description
fault	[1, 1]	XML element representing error message.
faultCode	[1, 1]	XML element representing class of errors.
faultString	[1, 1]	XML element where the error explanation is written.
detail	[1, 1]	XML element that carries error messages. It can contain multiple child elements.
requestUUID	[1, 1]	XML element that specifies UUID of the request for which error occurred.
code	[1, 1]	XML element that describes the error with a numeric code. List of codes can be found in the chapter 3.9.3.

Table 124

3.8.1.1 Header

This is an XML root element representing the header of the error message.

3.8.1.2 Header UUID

This is an attribute that uniquely describes the message and gives it the unique identification.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 125

3.8.1.3 Fault

This is an XML element that will appear only if error happens.

3.8.1.4 FaultCode

This is an XML element that represents origin of error that occurred.

Data type	string
Constraint	Enumeration, described in the table below.
Example	Client

Table 126

Enumeration values for the method of payment are listed in table below.

Value	Description
Client	Received message was incorrectly formed or contained incorrect information.
Server	There was a problem with the server, so the message could not proceed.

Table 127

3.8.1.5 FaultString

This is an XML element that contains textual explanation for error that occurred.

Data type	string
Length	Undefined
Example	Validation failed with digest wrong.

Table 128

3.8.1.6 Detail

This is an XML element that carries numeric error code.

3.8.1.7 Code

This is a Detail's child element, that describes the numeric error code. Numeric error codes are listed in the chapter3.8.2.

Data type	int
Length	3
Pattern	[1-9][0-9]{0,2}
Example	21

Table 129

3.8.1.8 RequestUUID

This is a Detail's child element, that specifies UUID of the request message that generated an error.

Data type	string
Length	36 characters
Pattern	[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[1-5][0-9a-fA-F]{3}-[89abAB][0-9a-fA-F]{3}-[0-9a-fA-F]{12}
Example	58e0a7d7-eebc-41d8-9669-0800200c9a66 58E0A7D7-EEBC-41D8-9669-0800200C9A66

Table 130

3.8.2 ERROR CODES

Following table lists all the error code that a fiscalization service can return to the TCR.

Error number	Error origin	Error description
0	Client	Exception occurred while extracting received XML message during size check.
1	Client	Received XML message exceed allowed size.
2	Client	Client and server time difference more than 60 minutes
10	Client	Exception occurred while extracting received XML message during XML validation against the XSD.
11	Client	Received XML message failed XSD validation.
20	Client	Exception occurred while extracting received XML message during signature check.
21	Client	Received XML message missing Signature XML element.
22	Client	Received XML message missing RegisterInvoiceRequest XML element.
23	Client	Exception occurred while extracting Signature XML element during signature check.
24	Client	Provided more than one Signature XML element.
25	Client	Signed wrong XML element.
26	Client	Wrong signature method specified.
27	Client	Wrong canonicalization method specified.
28	Client	Wrong digest method specified.
29	Client	Cryptographic signature wrong.
30	Client	Digest calculation wrong.
31	Client	Overall signature wrong.
32	Client	There are more key Info elements than needed.
33	Client	Certificate provided is not of X509 type of certificate.
34	Client	Certificate provided is not valid.
35	Client	Certificate is not issued by AKSHI.
36	Client	Certificate has expired.
37	Client	Compare the NUIS in XML with the NUIS in the certificate
40	Client	Invoice amount too large to be paid by cash.
41	Client	Issuer NUIS or business unit number are missing
42	Client	Software number is missing.
43	Client	Maintainer number is missing.
9xx	Server	Internal server exceptions

Table 131

3.8.3 EXAMPLE XML

```
<SOAP-ENV:Envelopexmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<SOAP-ENV:Fault>
<faultcode>SOAP-ENV:CLIENT</faultcode>
<faultstringxml:lang="en">Validation failed with digest wrong.</faultstring>
<detail>
<code>30</code>
<requestuuid>78dde160-2b33-40e4-98fa-f6a2c34475a3</requestuuid>
</detail>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

4. Data export of the non-fiscalized invoices

There are cases where invoice fiscalization will not be possible at the moment of invoice issuing. Reason for that is lack of the internet connection because of issuer being in remote areas or because of giving its services in places where having internet connection is not optional or it is difficult.

In these situations, issuer will issue the invoice to the buyer, along with all of its features, but fiscalization itself will not be made at the moment of invoice issuing. Issuers will store the invoices they issued in the device they used (cash register or similar). Each message, containing FIC and being in XML format, for example `<RegisterInvoiceRequest>…</RegisterInvoiceRequest>`, will go into its own folder.

Folder naming will follow this convention:

`RegisterInvoiceRequest_<NUIS-number>_<FIC>_<yyyyMMddHHmmss>.xml`

Once the issuer gets to the internet connection, he will do the fiscalization of his invoices through the Self-care portal. Other option for the issuer is to give his messages to the Tax administration where they will do the fiscalization on his behalf. After user will fiscalize the invoice by use of Self Care portal, fiscalized invoices shall be uploaded back to the cash register on the principle to replace original invoices which were not fiscalized with fiscalized ones.

5. Security

Following chapter describes security principles used by fiscalization service and parties that communicate with it.

5.1 CERTIFICATES

For the purpose of fiscalization there are three types of certificates that shall be used:

- Server certificate for transport security with common name *.tatime.gov.al issued to the service.
  Used to secure transport between the service and an issuer.
  It is issued by third party CA.
- Certificate for message security issued to the service.
  Used to digitally sign response data messages.
  It is issued by AKSHI CA.
- Certificate for message security issued to an issuer.
  Used to digitally sign request data messages and IIC data elements.
  It is issued by AKSHI CA.

AKSHI certificates are issued by the following certificate authorities:

- AKSHI Root Certification Authority (Self signed certificate)
  - AKSHI Certification Authority
    - AKSHI Class 1 Certification Authority

A digital certificate for fiscalization purposes is issued by the competent authority for issuing digital certificates (CAs). In the case of the Republic of Albania it is AKSHI. A detailed description of the submission of a request for issuing a digital certificate and obtaining a certificate is (or will have to be) defined in the instructions of AKSHI and is not the subject of this documentation.

5.2 TRANSPORT SECURITY

To ensure data security and integrity of the communication between the issuer and the service, service is using One-way TLS, specifically protocol version TLS V1.2. Service presents a certificate to client issued by the AKSHI CA, and with common name *.tatime.gov.al.

5.3 MESSAGE SECURITY

To ensure unambiguous identification of the taxpayer and to provide unchanged content, each request data message and IIC or WTNIC data element is digitally signed with a private key that is unique pair with the valid taxpayer certificate. Response data messages from the CIS are digitally signed with a private key that is unique pair with the valid CIS certificate.

In most cases, the private key used to digitally sign request data message and IIC or WTNIC data is the same. An exception under this rule is possible if the certificate used at the time of initial creation of the request is no longer valid at the time of resending the request. In that case, a private key from valid corresponding certificate must be used to digitally sign request data message, but not the IIC or WTNIC data element, which remains the same.

Request and response data messages are digitally signed according to the XML Signature Syntax and Processing standard edition 1.1 available at https://www.w3.org/TR/xmldsig-core/. Additional description is available in the chapter 5.3.1.

IIC or WTNIC data element is created and digitally signed according to the custom cryptographic algorithm described in the chapter5.3.2 or 5.3.3.

5.3.1 REQUEST AND RESPONSE DATA MESSASGE SIGNING

Every request and response data message described in the chapter 3, must contain signature XML element. That element is generated according to XML Signature Syntax and Processing standard edition 1.1 available at https://www.w3.org/TR/xmldsig-core/.

Element to signed is a first and only element inside soap envelope body XML element, with Id equals to Request or Response, depending on the message direction.

XML digital signature element is created with following options:

- Signature type: Enveloped, http://www.w3.org/2000/09/xmldsig#enveloped-signature
- Canonicalization method: C14 Exclusive, http://www.w3.org/2001/10/xml-exc-c14n#
- Digest method: SHA256, http://www.w3.org/2001/04/xmlenc#sha256
- Signing method: RSA SHA256, http://www.w3.org/2001/04/xmldsig-more#rsa-sha256

5.3.2 IIC DATA ELEMENT

IIC, Issuer's Invoice Code, is an alphanumeric security code generated by the issuer which uniquely matches issued invoice with an issuer. It is generated by concatenating specific parameters of the invoice and signed with a private key of the issuer.

IIC has two purposes:

1. To protect issuer from malicious third party because only the issuer that generated IIC can regenerate it by supplying the algorithm with the same parameters and using the same private key.
2. To verify that issued invoice is registered in the CIS.

At the tax administration's request, the taxpayer, based on the same input parameters, must create an IIC equal to that of the invoice.

IIC is generated using the following algorithm steps:

1. Concatenate parameters
2. Calculate digital signature with SHA256, RSA and RSASSA-PKCS-v1_5 padding
3. Calculate digest

5.3.2.1 Concatenate parameters

IIC is generated by concatenating following parameters of the invoice:

- Issuer NUIS (Chapter 3.6.2.30)
- Date and time created (Chapter 3.6.2.8)
- Invoice number(Chapter3.6.2.10)
- Business unit number(Chapter3.6.2.21)
- Cash register number(Chapter 3.6.2.11)
- Software number(Chapter 3.6.2.22)
- Total price(Chapter 3.6.2.18)

Before concatenation, all parameters must be converted into UTF-8 encoding. Parameters are concatenated with pipe character UTF-8 with decimal code 124.

For example, for parameters:

- Issuer NUIS: I12345678I
- Date and time created: 2019-06-12T17:05:43+02:00
- Invoice number: 9952
- Business unit number: bb123bb123
- Cash register number: cc123cc123
- Software number: ss123ss123
- Total price: 99.01

Resulted concatenated value is:
I12345678I|2019-06-12T17:05:43+02:00|9952|bb123bb123|cc123cc123|ss123ss123|99.01

5.3.2.2 Calculate digital signature

After the concatenation, resulting value is hashed with SHA256 algorithm and then signed with RSA algorithm and issuer's private key.

For example, for values:

- Concatenated value:
  I12345678I|2019-06-12T17:05:43+02:00|9952|bb123bb123|cc123cc123|ss123ss123|99.01
- PEM encoded private key:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA6zOR5ItNYHJNVMxljZtd/KQUyGIozbnIJ8IWqcEesktRV5FF
HviQZsx2DpyeVQTu/Kel9Xh+Z6OZ6t5sADzfYnkwCrsb0FhT+01m2PIHaIUZhVtc
ppn0gxNWfgzW4sTvTyrYk6O1Kxymsx/rck/WRQB1mp68au8mgGMzGukHfL7Wk4jO
U5VD3HlStBx1MjVW+soN5GUL/rWGaYun6Zsn9aYYEujbOhKvKDy8nOtNIS69dqdd
piZAkvdh9sYdF1ElgXZhdmZsGURMm6OcePUPZO/HFKq7RlK6vIxXVI6l9O6tWt+G
uhul8e0x2VTwbTdpwG4FpdfUTqUDK6cswHOhTQIDAQABAoIBAQCqBWJuUqDBmn76
ULMMlYZwjfAUFpkmdikRTIVzew4EltubMIFF7Sr9lMm2sFLoZKOZ8lrOwqalpqcq
GFT8KwTUO4SWDUIC7wbuf7pcE0F1tdmIBE5KhLozUnRQtFlWHkRb9z4OI+Zf3ttG
W0mpHbtnr/hTqHHN30j2wD7+MfvemPbcAvu9JLCYUzUZ06qxUwAjyFgsW7YyLa0a
qFB0QOYc6RsLvoSFXW0M5ghdtgoZvl+ayt4fgz1L3FjAMuXoLEX/778VA92/NZ0Q
mzQdKTT6B4Pm5s8XrY9OhLlsYqKuyR/aoSHC/anSLw0yJ/5Gis2gmCwo3a7+PEYy
LUN7C0yFAoGBAPhgYufTkdod5PqG/SCEE2i6pjk0ZnuIUu9f2cmhxnvyChlig2wk
oDWUSGuXwItNF+X7j3XoZz8FNJcriK7KP2UPDOWP0ZvxZgZEcmwut27x1vVjzjCG
sl0w5fFO363hhtX35Jq2lVZGbN1LpIoEZgCeS/nBs+9DcRjDoXliKWfHAoGBAPJr
qSWLVO3gIG1wikXBWCYZUTSzsO6NWfxcWPHKTnKVrOifBTK23zuZ6ggluNqLz/Ae
64ZwssMoIVIyXE01XMPP8io4QidyVEd2n70pjrVcUVYyr9IwKmchmNBfKFMof05f
NV29P1Am1Jqv2EQi5jE/BbBu9kLifs2YyGBAn/ZLAoGAVsLsqciZAVVCAFWZJHue
gA37NK5eQja7qcyUuj9dozxIVNe5ytP8dtrmdVccNkzm1TqLwYc+UaBS35+gblZN
0NJyEdqsQMoRdo0AX1PuVb369ds4UnEq6yzClgmUTxwhyqp+W6D+B5YwPxlGT8P7
kam6JnOIlEK9xgXIaStmBU8CgYB6RwXVszcOmYuhyC9mygSNix2j6LNpUJFAMtCG
fZYeRBMobvWvRADLznH21Bgu3HDxXJdOg9AXkklkbZSTOURmXKB43VG5Ffke5t3i
C3E5V6yLPxvieHsa9B5hlG4BrB6yyGFhvBCQfFWnBOWgUL4tvu0+tmmvCRIO4G7J
5i8JiwKBgQCQHTfRrGaEsq1BG7zPOQSqo9q5cxL8WzYd0sTs3FDcwCtHqxBEQ3rr
O/l+HvRa+y6ZEH6q4pREewTIymfv9tmGxVe3f8zrKGR5litvN6OnZuWJdq57Y1lN
J1sdpMxTtxQQmexsADif+QByCvdeFKE5C3veMLdgS5I6HTMN9k5laA==
-----END RSA PRIVATE KEY-----
```

Resulting signature value is:
404ADDB017B2DE49B0A51340A991130E670F08BC2BE854EEAAE9C3F41A2C98E1D70545690F0EFBD13511A38DB1E3
6E086DC253C3519E7DAF896A418BFAFCCE9836B0759B2E84713B25C39C040E35608AC85141A65D623454BAF4D0E04
D69A8D77505879C1DB9552542309A110B8CB2B9885C2236C3C6D65E695DFA4CA7D6258BD9EB0749A9EE09DA237C
4E1B8EE39C3CAD3E32A21F807DA0908192DADA3F9D55C4FEB3C100F97D5AA81CFE157E1A90059111E6DCD2F2AD3D

B9AAA202D084144E60ADED38988C384012967EF47B548135804EF2F4542DD0971E11AA392F048836D1C7DF9014F50
7B79258FA9B43AA14E32196D6127FD8154C24CE0CB374677D20

5.3.2.3 Calculate digest

After the signing, resulting value is hashed with a MD5 algorithm.

For example, for a value:

- Signature value:
  404ADDB017B2DE49B0A51340A991130E670F08BC2BE854EEAAE9C3F41A2C98E1D70545690F0EFBD13511A
  38DB1E36E086DC253C3519E7DAF896A418BFAFCCE9836B0759B2E84713B25C39C040E35608AC85141A65D6
  23454BAF4D0E04D69A8D77505879C1DB9552542309A110B8CB2B9885C2236C3C6D65E695DFA4CA7D6258B
  D9EB0749A9EE09DA237C4E1B8EE39C3CAD3E32A21F807DA0908192DADA3F9D55C4FEB3C100F97D5AA81CF
  E157E1A90059111E6DCD2F2AD3DB9AAA202D084144E60ADED38988C384012967EF47B548135804EF2F4542
  DD0971E11AA392F048836D1C7DF9014F507B79258FA9B43AA14E32196D6127FD8154C24CE0CB374677D20

MD5 digest value is: D04C13B4063D63A13B5D822A90178A7C

5.3.3 WTNIC DATA ELEMENT

WTNIC, Warehouse Transfer Node Identification Code, is an alphanumeric security code generated by the issuer which uniquely matches issued WTN with an issuer. It is generated by concatenating specific parameters of the WTN and signed with a private key of the issuer.

WTNIC has two purposes:

1. To protect issuer from malicious third party because only the issuer that generated WTNIC can regenerate it by supplying the algorithm with the same parameters and using the same private key.
2. To verify that issued WTN is registered in the CIS.

At the tax administration's request, the taxpayer, based on the same input parameters, must create an WTNIC equal to that of the WTN.

WTNIC is generated using the following algorithm steps:

1. Concatenate parameters
2. Calculate digital signature with SHA256, RSA and RSASSA-PKCS-v1_5 padding
3. Calculate digest

5.3.3.1 Concatenate parameters

WTNIC is generated by concatenating following parameters of the invoice:

- Issuer NUIS (Chapter3.7.1.6)
- Date and time created (Chapter3.7.1.8)
- WTN number (Chapter3.7.1.9)
- Business unit number (Chapter3.7.1.11)
- Software number (Chapter3.7.1.12)

Before concatenation, all parameters must be converted into UTF-8 encoding. Parameters are concatenated with pipe character UTF-8 with decimal code 124.

For example, for parameters:

- Issuer NUIS: I12345678I
- Date and time created: 2019-06-12T17:05:43+02:00

- WTN number: 9952
- Business unit number: bb123bb123
- Software number: ss123ss123

Resulted concatenated value is:
I12345678I|2019-06-12T17:05:43+02:00|9952|bb123bb123|ss123ss123

5.3.3.2 Calculate digital signature

After the concatenation, resulting value is hashed with SHA256 algorithm and then signed with RSA algorithm and issuer's private key.

For example, for values:

- Concatenated value:
  I12345678I|2019-06-12T17:05:43+02:00|9952|bb123bb123|ss123ss123
- PEM encoded private key:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEA6zOR5ItNYHJNVMxljZtd/KQUyGIozbnIJ8IWqcEesktRV5FF
HviQZsx2DpyeVQTu/Kel9Xh+Z6OZ6t5sADzfYnkwCrsb0FhT+01m2PIHaIUZhVtc
ppn0gxNWfgzW4sTvTyrYk6O1Kxymsx/rck/WRQB1mp68au8mgGMzGukHfL7Wk4jO
U5VD3HlStBx1MjVW+soN5GUL/rWGaYun6Zsn9aYYEujbOhKvKDy8nOtNIS69dqdd
piZAkvdh9sYdF1ElgXZhdmZsGURMm6OcePUPZO/HFKq7RlK6vIxXVI6l9O6tWt+G
uhul8e0x2VTwbTdpwG4FpdfUTqUDK6cswHOhTQIDAQABAoIBAQCqBWJuUqDBmn76
ULMMlYZwjfAUFpkmdikRTIVzew4EltubMIFF7Sr9lMm2sFLoZKOZ8lrOwqalpqcq
GFT8KwTUO4SWDUIC7wbuf7pcE0F1tdmIBE5KhLozUnRQtFlWHkRb9z4OI+Zf3ttG
W0mpHbtnr/hTqHHN30j2wD7+MfvemPbcAvu9JLCYUzUZ06qxUwAjyFgsW7YyLa0a
qFB0QOYc6RsLvoSFXW0M5ghdtgoZvl+ayt4fgz1L3FjAMuXoLEX/778VA92/NZ0Q
mzQdKTT6B4Pm5s8XrY9OhLlsYqKuyR/aoSHC/anSLw0yJ/5Gis2gmCwo3a7+PEYy
LUN7C0yFAoGBAPhgYufTkdod5PqG/SCEE2i6pjk0ZnuIUu9f2cmhxnvyChlig2wk
oDWUSGuXwItNF+X7j3XoZz8FNJcriK7KP2UPDOWP0ZvxZgZEcmwut27x1vVjzjCG
sl0w5fFO363hhtX35Jq2lVZGbN1LpIoEZgCeS/nBs+9DcRjDoXliKWfHAoGBAPJr
qSWLVO3gIG1wikXBWCYZUTSzsO6NWfxcWPHKTnKVrOifBTK23zuZ6ggluNqLz/Ae
64ZwssMoIVIyXE01XMPP8io4QidyVEd2n70pjrVcUVYyr9IwKmchmNBfKFMof05f
NV29P1Am1Jqv2EQi5jE/BbBu9kLifs2YyGBAn/ZLAoGAVsLsqciZAVVCAFWZJHue
gA37NK5eQja7qcyUuj9dozxIVNe5ytP8dtrmdVccNkzm1TqLwYc+UaBS35+gblZN
0NJyEdqsQMoRdo0AX1PuVb369ds4UnEq6yzClgmUTxwhyqp+W6D+B5YwPxlGT8P7
kam6JnOIlEK9xgXIaStmBU8CgYB6RwXVszcOmYuhyC9mygSNix2j6LNpUJFAMtCG
fZYeRBMobvWvRADLznH21Bgu3HDxXJdOg9AXkklkbZSTOURmXKB43VG5Ffke5t3i
C3E5V6yLPxvieHsa9B5hlG4BrB6yyGFhvBCQfFWnBOWgUL4tvu0+tmmvCRIO4G7J
5i8JiwKBgQCQHTfRrGaEsq1BG7zPOQSqo9q5cxL8WzYd0sTs3FDcwCtHqxBEQ3rr
O/l+HvRa+y6ZEH6q4pREewTIymfv9tmGxVe3f8zrKGR5litvN6OnZuWJdq57Y1lN
J1sdpMxTtxQQmexsADif+QByCvdeFKE5C3veMLdgS5I6HTMN9k5laA==
-----END RSA PRIVATE KEY-----
```

Resulting signature value is:
7F3E538EFDEA77AE4DD7AE9E6E1C5EA4A6AF6951745C0BBAF67E032A461A3D7B6E213BC787C312343C282C335BDE
F0413543AD117CBEAC2EA61FE32554C6C87880AE6279970D12C3C77D5C8CB045CAE6CFA4C904A4B09DCDC3E166C0
246FEBE9E05E60B02A0AC05A7DCAAE40407FA7222CDB2BF2EB82F1917CAB4D1D6785B9A5A0D61825EFE9A778A04C
04D0C75C09C41B7A1458FD2175E681239D5921AB8C65E5C213893FBBECB4F97153407398DD451A870929C471D7C7
969AB1181E50B6CA3692C641994BF3AC1009C45B68F8879007DCBFD3576FD02B83745781A0AE1851EFC7416F3D3EA
0D1E8EB01835D1B64931C6504AFD00BEFD84019CF994E9AF9AE

5.3.3.3 Calculate digest

After the signing, resulting value is hashed with a MD5 algorithm.

For example, for a value:

- Signature value:
  7F3E538EFDEA77AE4DD7AE9E6E1C5EA4A6AF6951745C0BBAF67E032A461A3D7B6E213BC787C312343C282
  C335BDEF0413543AD117CBEAC2EA61FE32554C6C87880AE6279970D12C3C77D5C8CB045CAE6CFA4C904A4
  B09DCDC3E166C0246FEBE9E05E60B02A0AC05A7DCAAE40407FA7222CDB2BF2EB82F1917CAB4D1D6785B9A
  5A0D61825EFE9A778A04C04D0C75C09C41B7A1458FD2175E681239D5921AB8C65E5C213893FBBECB4F9715
  3407398DD451A870929C471D7C7969AB1181E50B6CA3692C641994BF3AC1009C45B68F8879007DCBFD357
  6FD02B83745781A0AE1851EFC7416F3D3EA0D1E8EB01835D1B64931C6504AFD00BEFD84019CF994E9AF9AE

MD5 digest value is: 363350A9E21BC289253AAC9DC8C5E7E5

6. Code examples

This chapter covers the code examples for specific actions.

6.1 IIC GENERATION CODE

6.1.1 JAVA EXAMPLE

This is the example for the generation of the IIC in Java language. Variables are hardcoded as this is just an example.

```
importjava.io.FileInputStream;
importjava.security.*;

importjavax.xml.bind.DatatypeConverter;

publicclassSampleGenerateIIC{

privatestaticfinalString KEYSTORE_LOCATION ="***.p12";
privatestaticfinalString KEYSTORE_TYPE ="PKCS12";
privatestaticfinalString KEYSTORE_PASS ="***";
privatestaticfinalString KEYSTORE_KEY_ALIAS ="***";

publicstaticvoidmain(String[]args){

StringiicInput="";

// issuerNuis
iicInput+="I12345678I";
// dateTimeCreated
iicInput+="|2019-06-12T17:05:43+02:00";
// invoiceNumber
iicInput+="|9952";
// busiUnit
iicInput+="|bb123bb123";
// cashRegister
iicInput+="|cc123cc123";
// softNum
iicInput+="|ss123ss123";
// totalPrice
iicInput+="|99.01";

try(FileInputStreamfileInputStream=newFileInputStream(KEYSTORE_LOCATION)){
// Load a privatefrom a keystore
KeyStorekeyStore=KeyStore.getInstance(KEYSTORE_TYPE);
keyStore.load(fileInputStream,KEYSTORE_PASS.toCharArray());
KeyprivateKey=keyStore.getKey(KEYSTORE_KEY_ALIAS,KEYSTORE_PASS.toCharArray());

// Create IIC signature according to RSASSA-PKCS-v1_5
            Signature signature=Signature.getInstance("SHA256withRSA");
signature.initSign((PrivateKey)privateKey);
signature.update(iicInput.getBytes());
byte[]iicSignature=signature.sign();
StringiicSignatureString=DatatypeConverter.printHexBinary(iicSignature).toUpperCase();
System.out.println("The IIC signature is: "+iicSignatureString);

// Hash IIC signature with MD5 to create IIC
MessageDigest md =MessageDigest.getInstance("MD5");
byte[]iic=md.digest(iicSignature);
StringiicString=DatatypeConverter.printHexBinary(iic).toUpperCase();
System.out.println("The IIC is: "+iicString);
}catch(Exception e){
e.printStackTrace();
}
}
}
```

6.1.2 C# EXAMPLE

This is the example for the generation of the IIC in C# language. Variables are hardcoded as this is just an example.

```
using System;
usingSystem.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
usingSystem.Text;

namespaceFiscalizationSigningUtilityDotNet
{
classSampleGenerateIIC
{
privateconstString KEYSTORE_LOCATION ="***.p12";
privateconstString KEYSTORE_PASS ="***";

publicstaticvoidMain(string[]args)
{
StringiicInput="";

// issuerNuis
iicInput+="I12345678I";
// dateTimeCreated
iicInput+="|2019-06-12T17:05:43+02:00";
// invoiceNumber
iicInput+="|9952";
// busiUnit
iicInput+="|bb123bb123";
// cashRegister
iicInput+="|cc123cc123";
// softNum
iicInput+="|ss123ss123";
// totalPrice
iicInput+="|99.01";

using(X509Certificate2 keyStore=new X509Certificate2(KEYSTORE_LOCATION, KEYSTORE_PASS))
{
try
{
// Load a privatefrom a keystore
                    RSA privateKey=keyStore.GetRSAPrivateKey();

// Create IIC signature according to RSASSA-PKCS-v1_5
byte[]iicSignature=privateKey.SignData(Encoding.ASCII.GetBytes(iicInput), HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1);
stringiicSignatureString=BitConverter.ToString(iicSignature).Replace("-",string.Empty);
Console.WriteLine("The IIC signature is: "+iicSignatureString);

// Hash IIC signature with MD5 to create IIC
byte[]iic=((HashAlgorithm)CryptoConfig.CreateFromName("MD5")).ComputeHash(iicSignature);
stringiicString=BitConverter.ToString(iic).Replace("-",string.Empty);
Console.WriteLine("The IIC is: "+iicString);
}
catch(Exception ex)
{
Console.WriteLine(ex.Message);
}
}
}
}
}
```

6.2 WTNIC GENERATION CODE

6.2.1 JAVA EXAMPLE

This is the example for the generation of the WTNIC in Java language. Variables are hardcoded as this is just an example.

```
importjava.io.FileInputStream;
importjava.security.*;

importjavax.xml.bind.DatatypeConverter;

publicclassSampleGenerateWTNIC{

privatestaticfinalString KEYSTORE_LOCATION ="***.p12";
privatestaticfinalString KEYSTORE_TYPE ="PKCS12";
privatestaticfinalString KEYSTORE_PASS ="***";
privatestaticfinalString KEYSTORE_KEY_ALIAS ="***";

publicstaticvoidmain(String[]args){

StringwtnicInput="";

// issuerNuis
wtnicInput+="I12345678I";
```

```
// dateTimeCreated
wtnicInput+="|2019-06-12T17:05:43+02:00";
// wtnNumber
wtnicInput+="|9952";
// busiUnit
wtnicInput+="|bb123bb123";
// softNum
wtnicInput+="|ss123ss123";

try(FileInputStreamfileInputStream=newFileInputStream(KEYSTORE_LOCATION)){
// Load a privatefrom a keystore
KeyStorekeyStore=KeyStore.getInstance(KEYSTORE_TYPE);
keyStore.load(fileInputStream,KEYSTORE_PASS.toCharArray());
KeyprivateKey=keyStore.getKey(KEYSTORE_KEY_ALIAS,KEYSTORE_PASS.toCharArray());

// Create WTNIC signature according to RSASSA-PKCS-v1_5
            Signature signature=Signature.getInstance("SHA256withRSA");
signature.initSign((PrivateKey)privateKey);
signature.update(wtnicInput.getBytes());
byte[]wtnicSignature=signature.sign();
StringwtnicSignatureString=DatatypeConverter.printHexBinary(wtnicSignature).toUpperCase();
System.out.println("The WTNIC signature is: "+wtnicSignatureString);

// Hash WTNIC signature with MD5 to create WTNIC
MessageDigest md =MessageDigest.getInstance("MD5");
byte[]wtnic=md.digest(wtnicSignature);
StringwtnicString=DatatypeConverter.printHexBinary(wtnic).toUpperCase();
System.out.println("The WTNIC is: "+wtnicString);
}catch(Exception e){
e.printStackTrace();
}
}
}
```

6.2.2 C# EXAMPLE

This is the example for the generation of the WTNIC in C# language. Variables are hardcoded as this is just an example.

```
using System;
usingSystem.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
usingSystem.Text;

namespaceFiscalizationSigningUtilityDotNet
{
classSampleGenerateWTNIC
{
privateconstString KEYSTORE_LOCATION ="***.p12";
privateconstString KEYSTORE_PASS ="***";

publicstaticvoidMain(string[]args)
{
StringwtnicInput="";

// issuerNuis
wtnicInput+="I12345678I";
// dateTimeCreated
wtnicInput+="|2019-06-12T17:05:43+02:00";
// wtnNumber
wtnicInput+="|9952";
// busiUnit
wtnicInput+="|bb123bb123";
// softNum
wtnicInput+="|ss123ss123";

using(X509Certificate2 keyStore=new X509Certificate2(KEYSTORE_LOCATION, KEYSTORE_PASS))
{
try
{
// Load a privatefrom a keystore
                    RSA privateKey=keyStore.GetRSAPrivateKey();

// Create WTNIC signature according to RSASSA-PKCS-v1_5
byte[]wtnicSignature=privateKey.SignData(Encoding.ASCII.GetBytes(wtnicInput), HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1);
stringwtnicSignatureString=BitConverter.ToString(wtnicSignature).Replace("-",string.Empty);
Console.WriteLine("The WTNIC signature is: "+wtnicSignatureString);

// Hash WTNIC signature with MD5 to create IIC
byte[]wtnic=((HashAlgorithm)CryptoConfig.CreateFromName("MD5")).ComputeHash(wtnicSignature);
stringwtnicString=BitConverter.ToString(wtnic).Replace("-",string.Empty);
Console.WriteLine("The WTNIC is: "+wtnicString);
}
catch(Exception ex)
{
Console.WriteLine(ex.Message);
}
}
}
}
}
```

6.3 SIGNATURE GENERATION CODE

6.3.1 JAVA EXAMPLE

This is the example for the generation of the signature in Java language. Variables are hardcoded as this is just an example.

```
import java.io.*;
importjava.security.*;
import java.security.cert.X509Certificate;
importjava.util.*;

importjavax.xml.crypto.dsig.*;
importjavax.xml.crypto.dsig.keyinfo.*;
importjavax.xml.crypto.dsig.spec.*;
importjavax.xml.crypto.dsig.dom.DOMSignContext;
importjavax.xml.parsers.DocumentBuilder;
importjavax.xml.parsers.DocumentBuilderFactory;
importjavax.xml.transform.*;
importjavax.xml.transform.dom.DOMSource;
importjavax.xml.transform.stream.StreamResult;

import org.w3c.dom.*;
importorg.xml.sax.InputSource;

publicclassSampleGenerateSignature{

privatestaticfinalXMLSignatureFactoryxmlSigFactory=XMLSignatureFactory.getInstance("DOM");

publicstaticfinal String XML_SCHEMA_NS ="https://eFiskalizimi.tatime.gov.al/FiscalizationService/schema";
publicstaticfinal String XML_REQUEST_ELEMENT ="RegisterInvoiceRequest";
publicstaticfinal String XML_REQUEST_ID ="Request";
publicstaticfinal String XML_SIG_METHOD ="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256";

privatestaticfinal String REQUEST_TO_SIGN =
"<RegisterInvoiceRequest "+
"       xmlns=\"https://eFiskalizimi.tatime.gov.al/FiscalizationService/schema\" "+
"       xmlns:ns2=\"http://www.w3.org/2000/09/xmldsig#\" "+
"       Id=\"Request\">\r\n"+
"    <Header>...</Header>\r\n"+
"    <Invoice>...</Invoice>\r\n"+
"</RegisterInvoiceRequest>";
privatestaticfinal String KEYSTORE_LOCATION ="***.p12";
privatestaticfinal String KEYSTORE_TYPE ="PKCS12";
privatestaticfinal String KEYSTORE_PASS ="***";
privatestaticfinal String KEYSTORE_KEY_ALIAS ="***";

publicstaticvoid main(String[]args){

try(FileInputStreamfileInputStream=newFileInputStream(KEYSTORE_LOCATION)){
// Load a private from a key store
            KeyStore keyStore=KeyStore.getInstance(KEYSTORE_TYPE);
keyStore.load(fileInputStream,KEYSTORE_PASS.toCharArray());
            Key privateKey=keyStore.getKey(KEYSTORE_KEY_ALIAS,KEYSTORE_PASS.toCharArray());
            X509Certificate certificate =(X509Certificate)keyStore.getCertificate(KEYSTORE_KEY_ALIAS);

// Load XML to DOC
DocumentBuilderFactorydocFactory=DocumentBuilderFactory.newInstance();
docFactory.setNamespaceAware(true);
DocumentBuilderdocBuilder=docFactory.newDocumentBuilder();
            Document doc =docBuilder.parse(newInputSource(newStringReader(REQUEST_TO_SIGN)));

// Find root request element
NodeListnodeToSignList=doc.getElementsByTagNameNS(XML_SCHEMA_NS, XML_REQUEST_ELEMENT);
if(nodeToSignList.getLength()==0){
           thrownew Exception(String.format("XML element %s not found", XML_REQUEST_ELEMENT));
}
            Node nodeToSign=nodeToSignList.item(0);

// Create transform list
            List<Transform>transformList=newArrayList<>();
            transformList.add(xmlSigFactory.newTransform(Transform.ENVELOPED,(TransformParameterSpec)null));
            transformList.add(xmlSigFactory.newTransform(CanonicalizationMethod.EXCLUSIVE,(C14NMethodParameterSpec)null));

// Create digest reference element
            Reference ref =xmlSigFactory.newReference(
"#"+ XML_REQUEST_ID,
xmlSigFactory.newDigestMethod(DigestMethod.SHA256,null),
transformList,
null,
null);

// Create signature method
SignatureMethodsignatureMethod= xmlSigFactory.newSignatureMethod(XML_SIG_METHOD,(SignatureMethodParameterSpec)null);

// Create signed info element
SignedInfosignedInfo=xmlSigFactory.newSignedInfo(
                    xmlSigFactory.newCanonicalizationMethod(CanonicalizationMethod.EXCLUSIVE,(C14NMethodParameterSpec)null),
signatureMethod,
Collections.singletonList(ref));

// Add certificate
            List<X509Certificate>certificateList=newArrayList<>();
certificateList.add(certificate);
```

```
// Create key info element
KeyInfoFactorykeyInfoFactory=xmlSigFactory.getKeyInfoFactory();
            X509Data x509Data= keyInfoFactory.newX509Data(certificateList);
KeyInfokeyInfo=keyInfoFactory.newKeyInfo(Collections.singletonList(x509Data));

// Create context for signing
DOMSignContextdsc=newDOMSignContext(privateKey,nodeToSign);
dsc.setIdAttributeNS((Element)nodeToSign,null,"Id");

// Sign document
XMLSignature signature =xmlSigFactory.newXMLSignature(signedInfo,keyInfo);
signature.sign(dsc);

// Output to string
TransformerFactorytransformFactory=TransformerFactory.newInstance();
            Transformer transformer=transformFactory.newTransformer();
            transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION,"yes");
StringWritersw=newStringWriter();
StreamResultstreamRes=newStreamResult(sw);
transformer.transform(newDOMSource(doc),streamRes);
System.out.println("Signed document is: "+sw.toString());
}catch(Exception e){
e.printStackTrace();
}
}
}
```