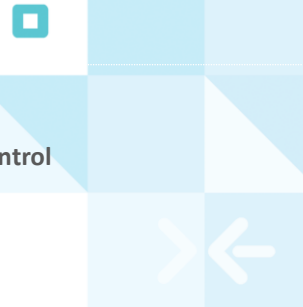


Fiscalization Service

"Improvement of the Management Module of the Billing Control"

Technical Specification

Fiscalization Service
Improvement of the Management Module of the Billing Control
Technical Specification (DRAFT)



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	9
2.3.2 Security preconditions.....	9
2.3.3 Application preconditions.....	9
3. INTERFACE	10
3.1 INTERFACE VERSIONING.....	10
3.2 DATA MESSAGE CODING	11
3.3 DATA MESSAGE STRUCTURE	11
3.4 REGISTER NEW TCR.....	11
3.4.1 Register TCRRequest Data Message	12
3.4.2 Register TCR ResponseData Message.....	15
3.4.3 Error Message	16
3.4.4 Example XML.....	16
3.5 REGISTER TCR CASH BALANCE	18
3.5.1 Register TCR Cash Balance RequestData Message.....	20
3.5.2 Register TCR Cash BalanceResponse Data Message.....	22
3.5.3 Error Message	23
3.5.4 Example XML.....	23
3.6 REGISTER INVOICE.....	24
3.6.1 UML process diagram	26
3.6.2 Register Invoice Request Data Message	27
3.6.3 Register Invoice Response Data Message	41
3.6.4 Error Message	43
3.6.5 Mandatory Controls.....	43
3.6.6 Optional Controls.....	44
3.6.7 Example XML.....	45
3.7 WAREHOUSE TRANSFER NOTE	46
3.7.1 Warehouse transfer note request message	46
3.7.2 Warehouse transfer note response message	53

3.7.3	Error Message	54
3.7.4	Example XML.....	54
3.8	ERROR MESSAGES.....	56
3.8.1	XML Format.....	56
3.8.2	Error Codes.....	58
3.8.3	Example XML.....	59
4.	DATA EXPORT OF THE NON-FISCALIZED INVOICES.....	60
5.	SECURITY.....	61
5.1	PKI	61
5.2	TRANSPORT SECURITY.....	61
5.3	MESSAGE SECURITY	61
5.3.1	Request And Response Data Message Signing.....	62
5.3.2	IIC Data Element.....	62
6.	CODE EXAMPLES	65
6.1	IIC CALCULATION CODE	65
6.1.1	Java example.....	65
6.1.2	C# example.....	66

1. Introduction

This document provides a description of the data interface for invoice and acknowledgement of data messages containing information on sales which the taxpayers are obliged to send for every sale made and subject to registration of Sales.

Files containing definition of the XML schema 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
CA	Certificate Authority
CIS	Central Invoice System
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
FIC	Fiscal Identification Code (generated at server side after successful verification of the invoice)
GUID	Global Unique Identifier
IIC	Issuer's invoice 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.
TCRN	Taxpayer Cash Register Number.
UC	Use case
UUID	Universally Unique Identifier
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

Fiscalization Service
Improvement of the Management Module of the Billing Control
Technical Specification (DRAFT)

1.2 TERMINOLOGY

Term	Definition
Response data message	A data structure in a defined format prescribed by the financial 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.
Error Data Message	A data structure in a defined format prescribed by the financial 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 information on a registered invoice.
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.
Registered invoice data message	<p>A data structure in a defined format prescribed by the financial 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.</p> <p>A registered invoice data message is sent by a cash register to the tax authority's common technical equipment.</p>
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.

Table 2

2. Environments

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

- **Non-production 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 non-production environment shall not be considered sending of registered invoice information. The FIC returned by the non-production environment is not a valid FIC (it is different per prefix). In the non-production environment, digital certificates for cash registers may be issued using a simplified process.
- **Production environment** is to be defined

Endpoints:

- o Non-production environment: <https://alimc.neos.hr/FiscalizationService>
- o Production environment: To be defined

2.1 PREPARATION WORKS FOR FISCALIZATION SERVICE USE

Details on this matter can be found in **Error! Reference source not found.**, chapter that covers this subject. Below is the process diagram.

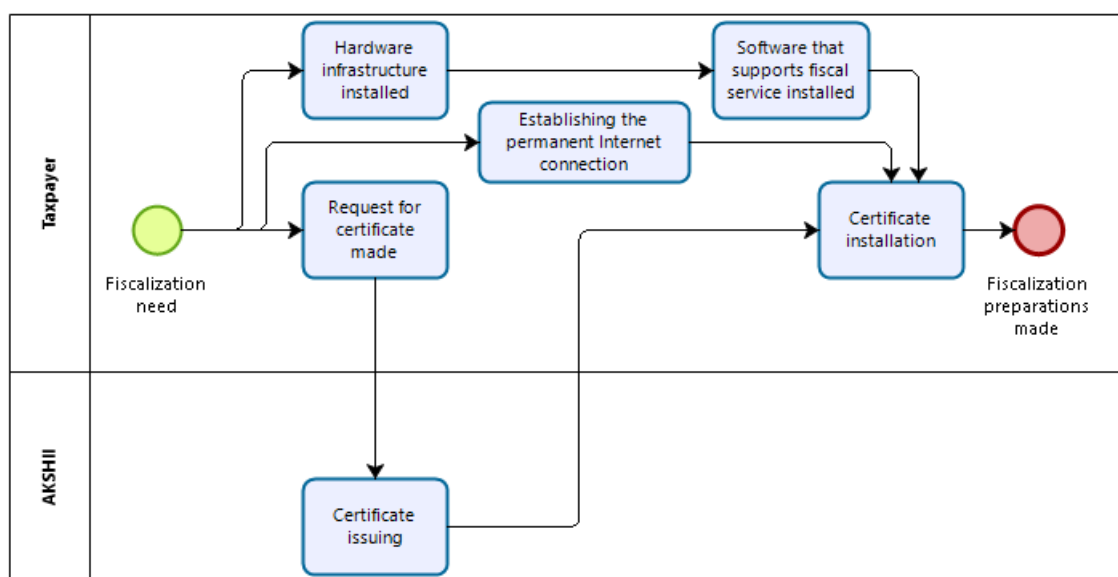


Figure 1

2.2 TOPOLOGY

Users access the CIS by initiating 1-way 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

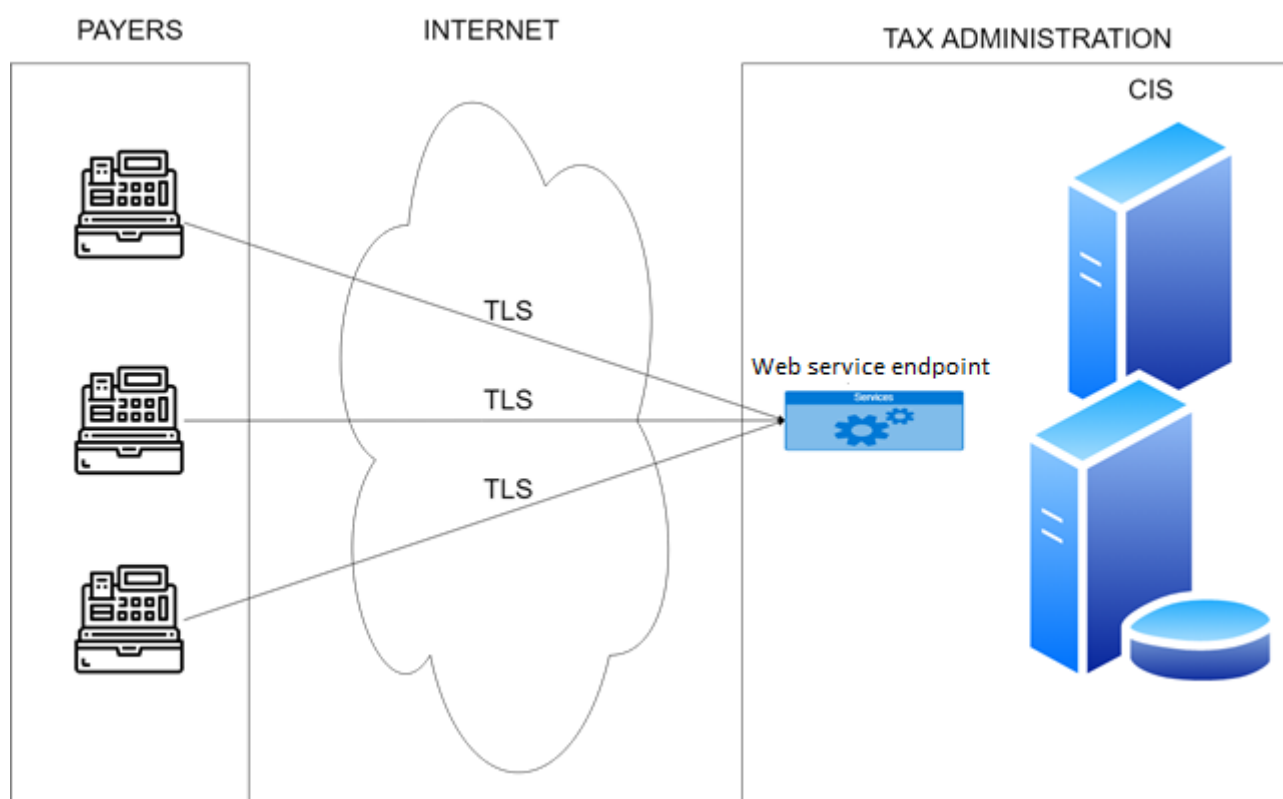
Implementation and maintenance of the access point is a TBD's task. TBD company will provide its users connection to the access point in two environments: production and test.

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 TBD company.



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. In production environment certificates are issued by TBD, while in test environment certificates are issued by TBD2. 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 Akshii production CA, while in test environment the certificate is issued by Akshii 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: applicationdigital certificate for fiscalization

2.3.3 APPLICATION 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 has items with negative amount and price which correspond to items in the original invoice. During the day, electronic billing devices for cash payments should register current cash balance. This is recommended to be done multiple times per day (to ensure the correct cash balance) and when the operator of the electronic billing device changes (because usually then the operator checks the cash balance). 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.

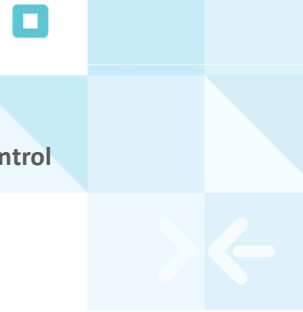
Message sent by the taxpayer to CIS is the request message to which CIS replies by sending the response message. In case of an error, the error message is sent in the response with its structure. Request and response messages (except for the error message) all have the following parts: header (general 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 Message Security.

3.1 INTERFACE VERSIONING

Versioning of the fiscalization service will be based on semantic versioning schema. Each version has a version number assigned which 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. /Fiscalization Service/v2. 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 3.7) 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.

Fiscalization Service
Improvement of the Management Module of the Billing Control
Technical Specification (DRAFT)

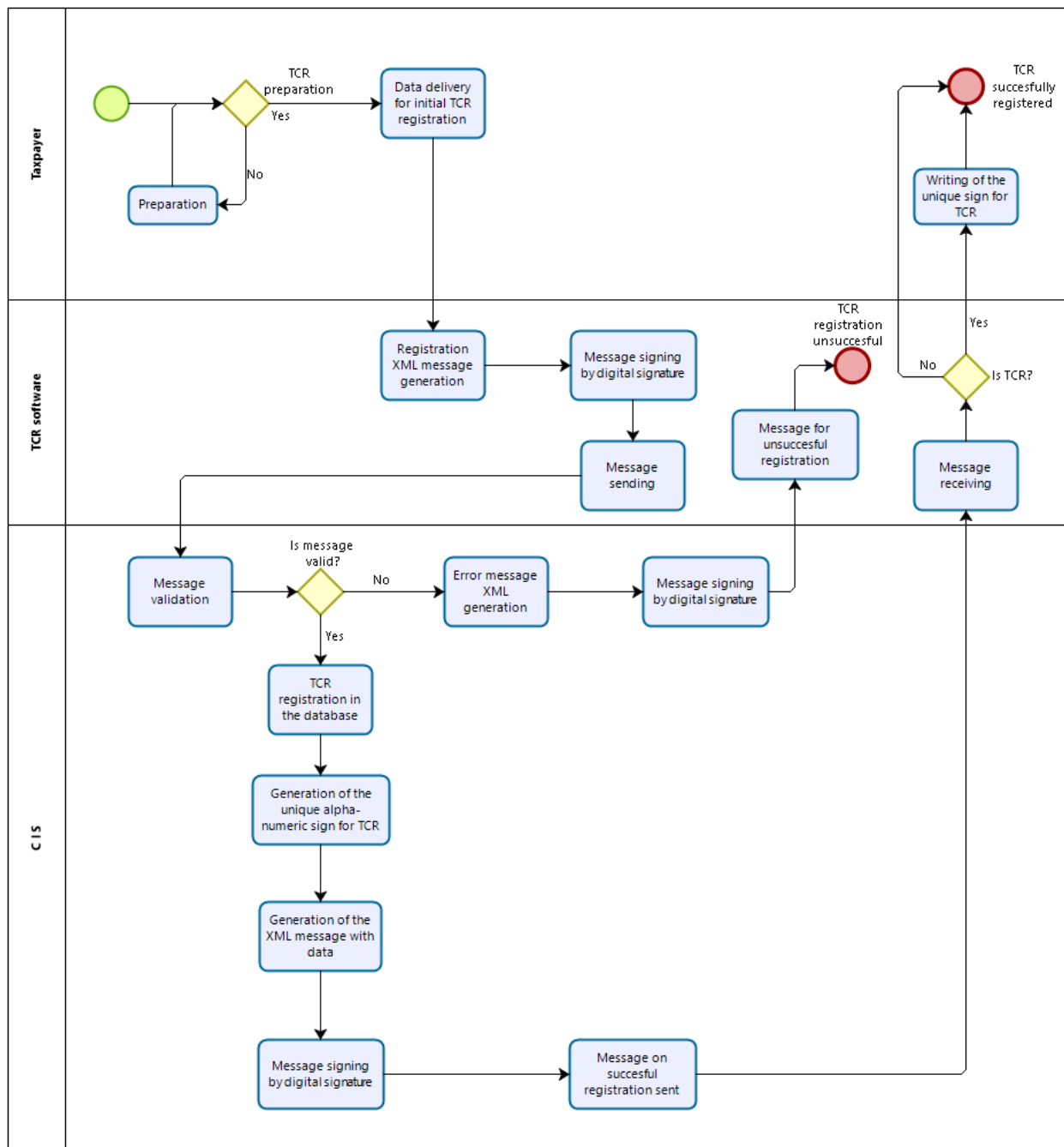


Figure 2 - Registration of electronic billing device

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.

Fiscalization Service
Improvement of the Management Module of the Billing Control
Technical Specification (DRAFT)

	SendDateTime	Attribute	[1, 1]	Date and time of sending the message to the Tax administration.
	TCRRequest	Element	[1, 1]	XML element representing a single registration message.
	RegDateTime	Attribute	[1, 1]	Date and time of the TCR registration.
	IssuerNUI5	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.

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

Table 5

3.4.1.4 TCRRequest

Element representing a single TCR registration request.

3.4.1.5 TCRRequest 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 TCRRequestIssuerNUIS

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 TCRRequestBusinUnit

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

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 8

3.4.1.8 TCRRequestTCROrdNum

Element representing the order number of the TCR.

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

Table 9

3.4.1.9 TCRRequest SoftNum

Number of the software used for invoice issuing.

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 10

3.4.1.10 TCRRequest ManufacNum

Number of the manufacturer of the software used for invoice issuing.

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 11

3.4.1.11 Signature

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

3.4.2 REGISTER TCR RESPONSEDATA 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

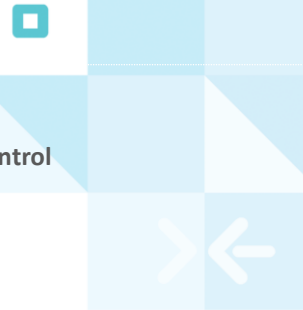
Element representing the header of the response data message.

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.

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 Request UUID

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.

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 Send Date Time

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 TCR Number

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 ERROR MESSAGE

Error message is defined in chapter 3.8

3.4.4 EXAMPLE XML

3.4.4.1 Request XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sch="http://alimc.neos.hr/FiscalizationService/schema" xmlns:xd="http://www.w3.org/2000/09/xmldsig#">
  <soapenv:Header/>
  <soapenv:Body>
    <RegisterTCRRequest xmlns="http://alimc.neos.hr/FiscalizationService/schema" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#" Id="Request">
```


Fiscalization Service Improvement of the Management Module of the Billing Control

Technical Specification (DRAFT)

```
<HeaderSendDateTime="2019-06-14T11:00:42+02:00"UID="9f4459da-e558-4c3c-bbff-217667096451"/>
<RequestBusinUnit="07790411-b967-454e-8e7c-a8154d719020"RegDateTime="2019-06-14T11:00:42+02:00"IssuerNUI="K72001009V"SoftNum="2191a92e-b3b1-44f5-a2eb-84abc817264f"ManufacNum="58e0a7d7-eebc-41d8-9669-0800200c9a66"TCROrdNum="58e0a7d7-eebc-41d8-9669-0800200c9a66"
</Request>
<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/xmldsig-more#sha256"><DigestValue>EIWEiHIM+ppzdWxR5tbh1TbyMGQZ7cEizG0eJuXButQ=</DigestValue></Reference></SignedInfo><SignatureValue>cvUspYF7Ui0J86su14VMWNH+OEKEzt0TcqJmL30cV2/uaDeqai1AH2IGW+NaHgZcs2mvrnbtpe7yxpVf/B4Yga02V8fIjW5UBMBmZnRes54bpfU34boPfz1ZZXHU9ujXwdkQJq0sBD9LuTPn8ye7MM9jswkNb3rdREuqmakaNZMHyIGfNgOpPpNhZhA++LoQBMTsMysQiuS2c1833dQcIEHPaf9MT8pMVZzelhOEqCep86pcz1T0N+LkihdMZCjUV1TR2xywGjNqGUYGMjZk0cBJF7fz9V19s0PxeCqVt5PF9kJs1Yy1Zfcs1/0PsVieFj1EQVx1E6GK1F2NVQ==</SignatureValue><KeyInfo><X509Data><X509Certificate>MIIETCCAtGgAwIBAgICEA8wDQYJKoZIhvcNAQELBQAwWDELMAKGA1UEBhMCWFxIDA0BgNVBAgM
B0Nyb2F0aWExFDASBgNVBAoMC05FT1MgZC5vLm8uSEwHwYDVQDDbH0RU9TIGQub3VlLjBjb3R1cm11ZG1hdGUwHhcNMjkwMjIyMTMxMTE1WncNMjkwMjIyMTMxMTE1WjBPMQswCQYDQGEwJiUjEQA4GA1UECAwHQ3JvYXRpYTEUMBIGA1UECgWLRGVtbyBmaXNRYWwGDABGMBAMMDZ2pc2thbDEuYmVvcy5ocjCCASIDQYJKoZIhvcNAQEBBQADgEPADCCAQoCggEBA0sZkeSLTWByTVMZY2bXfykFMhiKM25CfCfQnBHRJLUVeRRR74kGbmDg6cniUE7vynpfV4fmejmerebAA832J5MAq7G9BYU/tNZtjyB21FGYVbXka292MTVn4M1uLE708q2J0jtScprMf63JP1kUAdZqevGrvJoBjMxrpB3y+1pOIz10V09x5UrQcdTI1VvrKDeR1C/61hmmLp+mbJ/WmGBLo2zoSryg8vJzrTSEuvXanXaYmQJL3YfBgHrDRJYF2YXZmbB1ETJujnHj1D2TvxSquZSuryMV150pfTurVrFhrobpFhtMdLU8G03acBuBaXx1E61AyunLMBzo0CAwEAaA0BxTCBwJAjBgNVHREMAjAAMBEGCwAGSAGG+EIBAQQEAWIFoDAZBg1ghgBhvhCAQ0EJhYkT3B1b1NTCTBHZW51cmF0ZWQwQ2xpZW50IEN1cnRpZm1jYXRIMB0A1UdDQOWBQOPSMUP/iZc6yxybyFYV9quMDLq5uzAFBgNVHSMGDAWgBS7a9gudUVZTP3chYtQKDNcDqOzAOBgNVHQ8BAf8EBAMCBwAwHQYDVDR01BBYwFAyIKwYBBQUHAWGCCGAQUFBwMEMA0GCsQG5Tb3DQEB
CwUAA41CAQAJAQ+Fa3FtbxgHpKPagG9+hq8oCzdb5M/WfEmtLUUwum1y4nVf9C+ohn0BtA09A1UBDp4fNPGuPRjISB7jy1uDYxJ50aCLMnrvBnypcuAndxM07L1YX38wLiTiSdVjyJm2w6xNj49aXAPWexJwUfEcK270F9Pj7y7f8wc+zc29m1UBbmKvDQgPqHHdsCBYmpTf5SHFvBXQZBrqLXfU/P
PdkHxzi8337eUQGaUzn1S87oIDJYEmAVBLG90VU52B/eG1PBXKqR3tgrhRH1J5V/ANPP76n3UZ1SXqTR8ujK3dUxQPSzylyYqYXcW5H5gBwIAm1kbZEoXGuzd3M+NYPATQdkOLKYuPZKCRKLKcZ8XiYz7DHLjEjYTCvxdT7HLAw4LncgtJhIFBUa22Dh1Y2At2rznXoFyeDtdndQG6KR6XwL+EHPbD6FZPREmBcJpTAhQh67e4dVtNFjFAGYzKbB7C7g4LDGpdFBZiJ7GZ2H51GJUTUZ2uZ6T31hmE2DES66v70xHZFN8NElrZ/ZGVNWDh7XOYB7CmutL+ldydb0mpXnv1l1mkX84mC5019s8XqFHKd9YLVOCnZ1hhU7AUm6XBZtPt1t00AKpCZaub0eyrCwJm1XK1iigNagaXUjrrrii/j5a1LAq5sMhZeH0MEUFLuLuoqW5sLuMA==</X509Certificate></X509Data></KeyInfo></Signature></RegisterTCRRequest>
</soapenv:Body>

</soapenv:Envelope>
```

3.4.4.2 Response XML

```
<S:Envelopexmlns:env="http://schemas.xmlsoap.org/soap/envelope/"xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<env:Header/>
<S:Body>
<RegisterTCRResponseId="Response"xmlns="http://alimc.neos.hr/FiscalizationService/schema"xmlns:ns0="http://www.w3.org/2000/09/xmldsig#">
<TCRNumber>58e0a7d7-eebc-41d8-9669-0800200c9a66</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/xmldsig-more#sha256">
<DigestValue>gopN9YJPpj/gyqmkQQ/sBbH25sWkInwQ26Iy3bvQVg=</DigestValue>
</Reference>
</SignedInfo>
<SignatureValue>E50rIGE+pC3DIqpsSVjce+yItcxsp0U+yWqID0IyaxP2s90thYdihk6Z77treRWJBgsVm1X2+Yn
PcI31kqhztmKLwhdLqLHPJRURV+IV/3cQNfwd/79AypN1g48CXyEkwHwHMW2i1VLcdFOR86cvMF
1+jVC2cvM12ogLmaix70E7N9tfJkXApbQ5b8+btz13jnB+y0iQhhtV4ZofDrDpwgUggjFW2F2N6
Pdwngdx7b89ENphgppUyV0g+PCQY0tI4aygJngQVPay+9bSga66eFwYehJcYMCb4afFDIKLJIUvk
+baqfKZPA82LGvHG0xaJ4wt+ZkfXqZUCJ7qo56Q==</SignatureValue>
<KeyInfo>
<X509Data>
<X509Certificate>MIIETCCAtGgAwIBAgICEA8wDQYJKoZIhvcNAQELBQAwWDELMAKGA1UEBhMCWFxIDA0BgNVBAgM
B0Nyb2F0aWExFDASBgNVBAoMC05FT1MgZC5vLm8uSEwHwYDVQDDbH0RU9TIGQub3VlLjBjb3R1cm11ZG1hdGUwHhcNMjkwMjIyMTMxMTE1WncNMjkwMjIyMTMxMTE1WjBPMQswCQYDQGEwJiUjEQA4GA1UECAwHQ3JvYXRpYTEUMBIGA1UECgWLRGVtbyBmaXNRYWwGDABGMBAMMDZ2pc2thbDEuYmVvcy5ocjCCASIDQYJKoZIhvcNAQEBBQADgEPADCCAQoCggEBA0sZkeSLTWByTVMZY2bXfykFMhiKM25CfCfQnBHRJLUVeRRR74kGbmDg6cniUE7vynpfV4fmejmerebAA832J5MAq7G9BYU/tNZtjyB21FGYVbXka292MTVn4M1uLE708q2J0jtScprMf63JP1kUAdZqevGrvJoBjMxrpB3y+1pOIz10V09x5UrQcdTI1VvrKDeR1C/61hmmLp+mbJ/WmGBLo2zoSryg8vJzrTSEuvXanXaYmQJL3YfBgHrDRJYF2YXZmbB1ETJujnHj1D2TvxSquZSuryMV150pfTurVrFhrobpFhtMdLU8G03acBuBaXx1E61AyunLMBzo0CAwEAaA0BxTCBwJAjBgNVHREMAjAAMBEGCwAGSAGG+EIBAQQEAWIFoDAZBg1ghgBhvhCAQ0EJhYkT3B1b1NTCTBHZW51cmF0ZWQwQ2xpZW50IEN1cnRpZm1jYXRIMB0A1UdDQOWBQOPSMUP/iZc6yxybyFYV9quMDLq5uzAFBgNVHSMGDAWgBS7a9gudUVZTP3chYtQKDNcDqOzAOBgNVHQ8BAf8EBAMCBwAwHQYDVDR01BBYwFAyIKwYBBQUHAWGCCGAQUFBwMEMA0GCsQG5Tb3DQEB
CwUAA41CAQAJAQ+Fa3FtbxgHpKPagG9+hq8oCzdb5M/WfEmtLUUwum1y4nVf9C+ohn0BtA09A1UBDp4fNPGuPRjISB7jy1uDYxJ50aCLMnrvBnypcuAndxM07L1YX38wLiTiSdVjyJm2w6xNj49aXAPWexJwUfEcK270F9Pj7y7f8wc+zc29m1UBbmKvDQgPqHHdsCBYmpTf5SHFvBXQZBrqLXfU/P
PdkHxzi8337eUQGaUzn1S87oIDJYEmAVBLG90VU52B/eG1PBXKqR3tgrhRH1J5V/ANPP76n3UZ1SXqTR8ujK3dUxQPSzylyYqYXcW5H5gBwIAm1kbZEoXGuzd3M+NYPATQdkOLKYuPZKCRKLKcZ8XiYz7DHLjEjYTCvxdT7HLAw4LncgtJhIFBUa22Dh1Y2At2rznXoFyeDtdndQG6KR6XwL+EHPbD6FZPREmBcJpTAhQh67e4dVtNFjFAGYzKbB7C7g4LDGpdFBZiJ7GZ2H51GJUTUZ2uZ6T31hmE2DES66v70xHZFN8NElrZ/ZGVNWDh7XOYB7CmutL+ldydb0mpXnv1l1mkX84mC5019s8XqFHKd9YLVOCnZ1hhU7AUm6XBZtPt1t00AKpCZaub0eyrCwJm1XK1iigNagaXUjrrrii/j5a1LAq5sMhZeH0MEUFLuLuoqW5sLuMA==</X509Certificate></X509Data></KeyInfo></Signature></RegisterTCRRequest>
</S:Body>
</S:Envelope>
```



```
ve0t0lkXPX1X8SisGY0ZnsQ+cxZKZm0cwL0ILdniVQERUFXPYRQ24D1mqGhyVWjSJUXSvCjhknhB
nJGnyTQ5+DQ9tnLdQu7+azeDwNNTG/QgnLbgFVKzXwNqXJsi1v1/Vqw4F8/k9+D/Wdi5FX28TXf
Kf6LjkGHiCKItYQaVuNGEVSTFU0YvgQWpSvvtSF7yJ0T7S+/0Ui905H+NZ0biLZe9ZrdaB2VK4pz
FEvYEPtz2XyY2U0MGJALd0f9McEUuXEz0OrVmgT0xSEGI+3wys1Jx/pe0zpS8pMSm9S3a/ukK0FA
t0ajqoT3QYijguqetwyL/c+iSNY9Wm6XGXPH2RwkrOUFmdskN511uDerDD8RYD/5sYXzy+J12deT
FbRi7FZKdQZc/WdwrUYIHYKTCznxUZu2SQ/hsah</X509Certificate>
</X509Data>
</KeyInfo>
</Signature>
</RegisterInvoiceResponse>
</S:Body>
</S:Envelope>
```

3.5 REGISTER TCR CASH BALANCE

Every day before registering the first invoice on CIS, each electronic billing device for handling cash transactions should register the amount of cash in the deposit. During the day, the operator can put or take cash from the electronic billing device and check the cash balance and then the operator should initiate the action of registering new (current) TCR cash balance.

Fiscalization Service
Improvement of the Management Module of the Billing Control
Technical Specification (DRAFT)

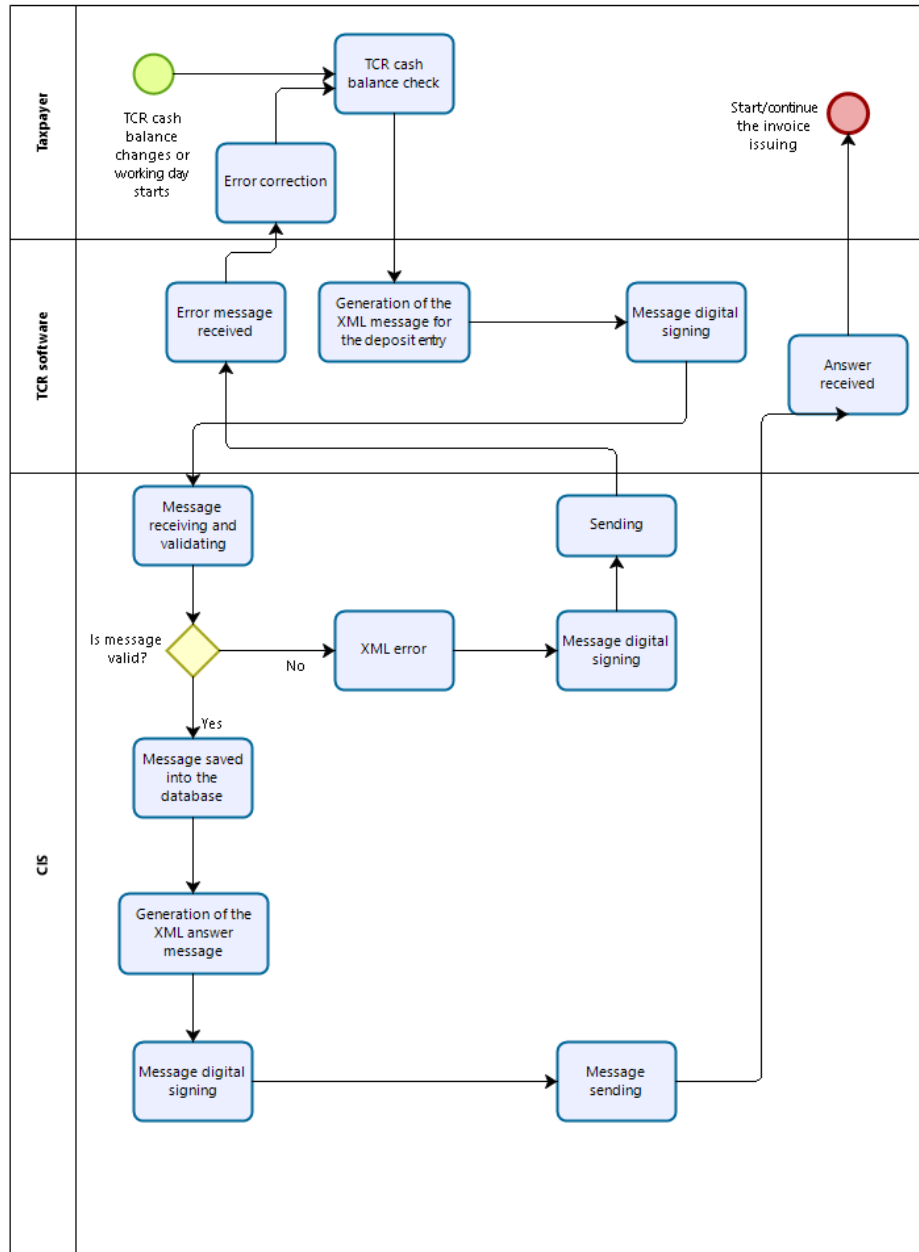


Figure 3

For more details see **Error! Reference source not found.** document, chapter about registration of cash balance of electronic billing device.

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 invoice.
BalanceCheckDateTime	Attribute	[1, 1]	Date and time when the cash balance was checked.
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.
IssuerNUIIS	Attribute	[1, 1]	Taxpayer's NUIIS.
Signature	Element	[1, 1]	XML element with signature.

Table 17

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.

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 18

3.5.1.3 Header Send Date Time

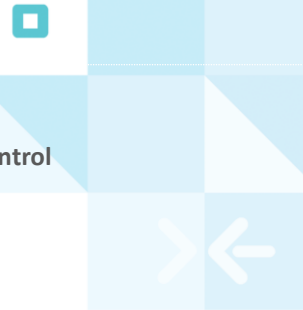
Element represents date and time of sending the request message to the CIS. Date and time should be in ISO 8601 format.

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 19

3.5.1.4 Cash Balance

Element representing a single cash balance registration.



3.5.1.5 Cash Balance Balance Check Date Time

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 20

3.5.1.6 Cash Balance Cash Amt

Element representing the amount of cash found in the cash register at the time it was checked.

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

Table 21

3.5.1.7 Cash Balance TCR Number

Element representing the unique number of the TCR in question.

Data type	string
Length	36 charaters
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 22

3.5.1.8 Cash Balance Issuer NUIS

Element representing issuer's NUIS (tax number).

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

Table 23

3.5.1.9 Signature

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

3.5.2 REGISTER TCR CASH BALANCERESPONSE 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 24

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.

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 25

3.5.2.3 Header Request UUID

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.

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 26

3.5.2.4 Header Send Date Time

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 27

3.5.2.5 Signature

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

3.5.3 ERROR MESSAGE

Error message is defined in chapter 3.8.

3.5.4 EXAMPLE XML

3.5.4.1 Request XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sch="http://alimc.neos.hr/FiscalizationService/schema" xmlns:xd="http://www.w3.org/2000/09/xmldsig#">
  <soapenv:Header/>
  <soapenv:Body>
    <RegisterTCRCashBalanceRequest xmlns="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"/>
      <CashBalanceCashAmt="57.16" BalanceCheckDateTime="2019-06-14T11:00:42+02:00" TCNNumber="58e0a7d7-eebc-41d8-9669-0800200c9a66" IssuerNUIIS="K72001009V"/>
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#"><SignedInfo><CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"><SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/><Reference URI="#Request"><Transforms><Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/></Transforms></Reference></SignedInfo><SignatureValue>cUusypF7U10J86su14VMwNH+OEKEzt0TcqJmL30cV2/uaDeqa11AH2IGW+NaHgZcs2mvrnbtpe7yxpVf/B4Yga02V8f1JW5UBMBzNreS4bpFu34boPfz1ZXXHU9uJxWdkQJQ0sBD9LuTPn8ye7MM9jswkN3r3rEuqmaknZMHYIGfNgOpQpNhZHA++LoQBmUTsMYsQ1uS2ci833dQcIEHPaf9MT8pMVZze1hOEQxCep86pczLT0N+LkihdnzDCjUV1TR2xywGjNqGUyGMjZk0cBJF7fzh9V19soPXcQVt5PF9K3s1Yz1Zfcs1/0PsVieFjEiEQVx1E6GK1F2NVQ==</SignatureValue><KeyInfo><X509Data><X509Certificate>MIIIE6TCCAtGgAwIBAgICEA8wDQYJKoZIhvcNAQELBQAwWDELMAkGA1UEBhMCSFJxEDA0BgNVBAGM
      B0Nyb2F0aWExFDASBgNVBAoMC05FT1MgZC5vLm8uMSEwHwYDVQDDbH0RU9TIGQub3y5vL1BJbnR1cm11ZG1hdGUwHhcnMTkwMjIyMTMxMTE1WmcNMjcwMjIwMTMxMTE1WjBPMQswCQYDVQGEWJlUjEQA4GA1UECAwH03jvYXRpYUeUMBGA1UECgwLRGVtbyBmaXNrYXVwGDAwBgNVBAMMD2Zpc2thbDEuYmVycySocjCCASIdWQYJKoZIhvcNAQELBQADggEPAADCCAQoCggEBAOsZkeSLTWBYVTMTZy2bXfykFhkiM25yCfCFgN8HjLUeVRRR74kGbMdg6cn1UE7vynpFV4fmejmerebAA832J5MAq7G9BYU/tNZtjyB2iFGYVbXkaZ9IMTVn4M1uLE708q2J0jtSscprHf63JP1kUAdZqevGrvJoBjMxrpB3y+1pOIz10VQ9y5UrQcdT1IvvrKDeR1C/61hmmLp+mbJ/WmGBLo2zoSryg8vJzrTSEuvXanXaYmQJL3YfBgHrDRYF2YXZmb81ETJujnhJ1D2VxxSqu0ZSuryMV1S0pfTurVrFhrobpFhtMdLU8G03acBuBaXX1E61AyunLMBzoU0CAwEAAQOBxTCBwjAJBgNVHRMEAIAAMBGCCWGSAGG+IEBAQQEAWIfoDAzBg1ghkgBhvChCAQ0EJhYkT3B1b1NTTCBHW51cmF0ZWQgQ2xpZW50IEN1cnRpZm1jYXR1MB0GA1UdDgQWBBOQSMUP/iZCy6xybFYV9quMDLq5uzAFBgNVHSMGDAWgBS7a9gudUvZTTP3chYtQKdNCdQoZA0BgNVHQ8BAf8EBAMCBAwHQYDVDR01BBYwFAYIKwYBBQUHAWIGCCsGAUQFwBMEAAOGCSqGSIb3DQEB
      CwUAA4CAQAAQ+fa3FtbxgHpkEPagG9+hq8oCzdb5M/WfEMTLKUQuwm1y4nvf9C+ohn0BtA09A1UBDp4FNPGuPRjLSB7jyUdyXj50aCLMnfvBnypcuANdxM07L1YX38wLiTiSdvYjMi2w6xNj49aXAPWexjwUFeCCK270FB9yJ7yF8wc+zcZ9m1UBBmKvDQgPqHHhdsCBbYmpTtF5HfVbXqZBrqzLXfU/PDkHxzwi8337eUQGaIUzn1S87oIDJYEmAVBLG90VU52B/eG1PBXKqR3tgrhRHj5V/ANPP76n3Uz1SXqTR8ujK3dUxQzPy1YqYXcWHSgBwIAm1kbZEoVXGuzd3M+NYPATQdkOLKYuPZKCRKLKcZ8XiYz7DHLJEYTCvxdT7HLAw4LNCtgJhIFBUa22Dh1Y2At2rznXoFyeDtdnQG6KR6XwL+EHpbD6FZPREmBcJpTahqKH67z4dVTNFjAGYAZkBB7C7g4LDGpdfBzIj7GZ2H51GJuTUZ2uZ6T31hMEZDES6v70xHZFN8NElrZ/ZGYNWDh7XOYB7CmutL+ldydbOmpXnv1v1mkX84mC5o19s8XqFHKd9YLVOCNz1hhU7AUm6XBZtPt1t004KpCZaub0eyrCWjM+1XKiiGNagaXUjrriiif/j5a1LAq5sMhZeH0MWEUfLuLuoqW5SuLm==</X509Certificate></X509Data></KeyInfo></Signature></RegisterTCRCashBalanceRequest>
    </soapenv:Body>
  </soapenv:Envelope>
```

3.5.4.2 Response XML

```
<S:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header/>
  <S:Body>
    <RegisterTCRResponse Id="Response" xmlns="http://alimc.neos.hr/FiscalizationService/schema" xmlns:ns0="http://www.w3.org/2000/09/xmldsig#">
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
          <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
          <SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
        </Signature>
      </Signature>
    </RegisterTCRResponse>
  </S:Body>
</env:Envelope>
```


3.6 REGISTER INVOICE

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.

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).

3.6.1 UML PROCESS DIAGRAM

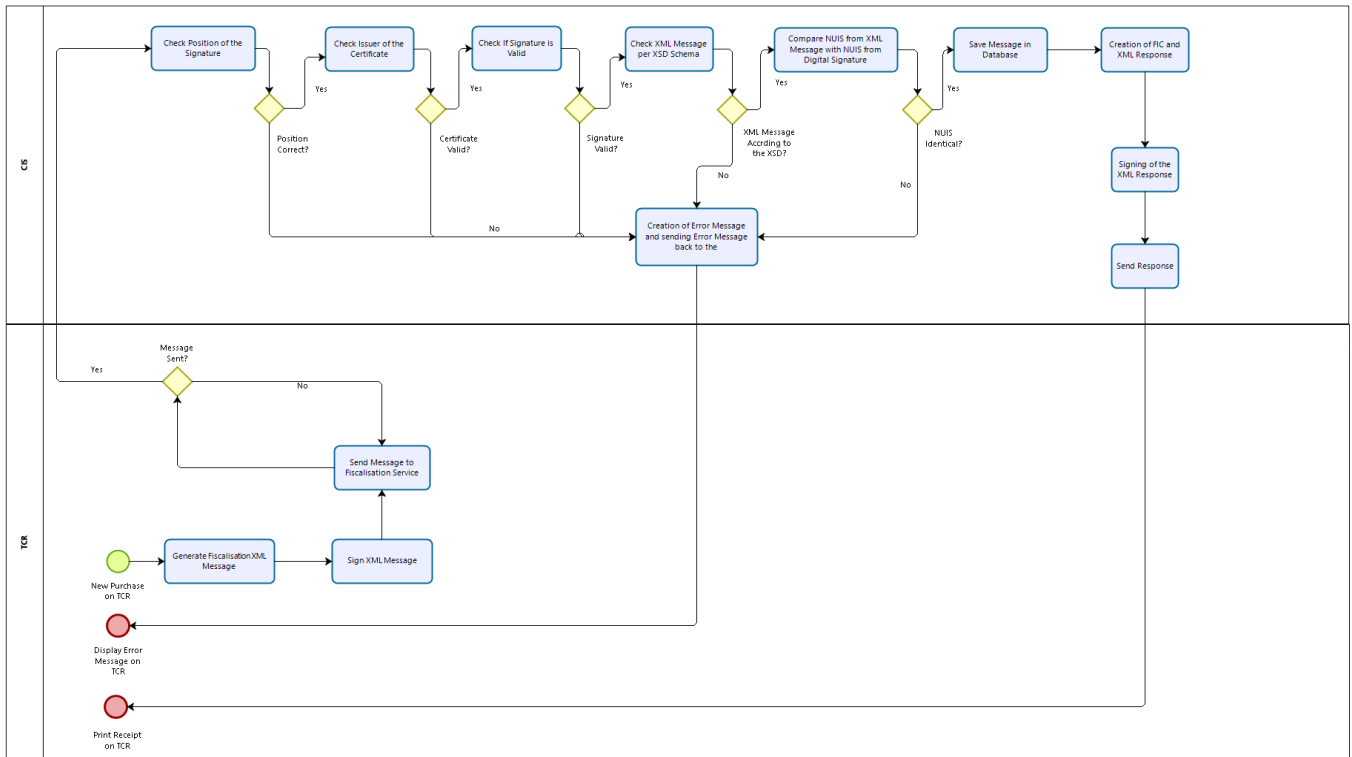


Figure 4

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 150kB, 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
 - DateTimeCreated
 - IssuerTaxNumber
 - Operator
 - BusinessUnit
 - CashRegister
 - TotalPrice
 - TotalVATAmount
 - PaymentMethod
 - TypeOfInvoice (invoice type)
 - ParagonNumber
 - TSCReference
- 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
RegisterSaleRequest	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.

Fiscalization Service Improvement of the Management Module of the Billing Control

Technical Specification (DRAFT)

	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 codeconcatenated 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	[1, 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.
	Item	Element	[1, *]	XML element representing one item.
	Name	Attribute	[1, 1]	Name of the item (goods or services).
	Code	Attribute	[0, 1]	Code of the item from the barcode or similar representation.
	Unit	Attribute	[1, 1]	What is the item's unit of measure (piece, weight measure, length measure...).
	Quantity	Attribute	[1, 1]	Amount or number (quantity) of items
	UnitPrice	Attribute	[1, 1]	Amount or number (quantity) of items
	Rbt	Attribute	[1, 1]	Percentage of the rebate.
	IsRbtRed	Attribute	[0, 1]	Is rebate reducing base amount?
	PriceBefVAT	Attribute	[1, 1]	Total price of goods and services before the tax
	VATRate	Attribute	[1, 1]	Rate of value added tax
	VATAmt	Attribute	[1, 1]	Amount of value added tax for goods and services
	PriceAftVAT	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.
	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.

Fiscalization Service
Improvement of the Management Module of the Billing Control
Technical Specification (DRAFT)

	ConsTaxItems	Element	[0, 1]	Special, consumption taxes (alcohol, beverages).
	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 28

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.

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 29

3.6.2.4 Invoice

XML element representing a single invoice.

3.6.2.5 Invoice TypeOfInv

Type of the item represents the type of invoice item, e.g. regular sale or a returned item.

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

Table 30

Following table list allowed values inside TypeOfInv attribute.

Value	Description
C	Cash
N	Non-cash

Table 31

3.6.2.6 Invoice Selfissuing

The invoice can be issued by buyer himself.

Data type	boolean
Values	true, false
Example	true

Table 32

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 33

Following table list 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

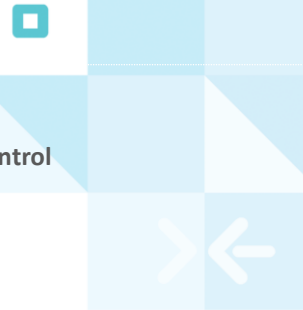
Table 34

3.6.2.8 Invoice DateTimeCreated

Time and date when the invoice is created.

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 35



3.6.2.9 Invoice InvNum

Invoice number printed on the receipt.

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

Table 36

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 37

3.6.2.11 Invoice CashRegister

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

Data type	string
Length	36
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 38

3.6.2.12 Invoice IssuerInVAT

Is taxpayer in the VAT system.

Data type	boolean
Values	true, false
Example	true

Table 39

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 40

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 41

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 42

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 43

3.6.2.17 Invoice TotVATAmt

Total amount of value added tax which needs to be payed 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 44

3.6.2.18 Invoice TotPrice

Total price which needs to be payed 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 45

3.6.2.19 Invoice PaymentMeth

Method of payment.

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

Table 46

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 47

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
Pattern	[a-zA-Z]{1}[0-9]{8}[a-zA-Z]{1}
Example	K72001008V

Table 48

3.6.2.21 Invoice BusinUnit

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

Data type	string
Length	36
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 49

3.6.2.22 Invoice SoftNum

Number of the software used for invoice issuing.

Fiscalization Service
Improvement of the Management Module of the Billing Control
Technical Specification (DRAFT)

Data type	string
Length	36
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 50

3.6.2.23 Invoice IIC

Issuers's 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 chapter 5.3.2.3.

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

Table 51

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 52

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	B2C218486302EC553EE1AB9124E1A14705742E870E8872EF34E63617AB252E189ACDF7A3E3F5C82061FFFF8AC2826A5588596A8807F648410899B6193F77F4BDCDFA 87553A62079A2EF9E6E6F0B8DA1038968D2FCB920B580EBF33ACEEDFEA0DAA78067F916ADC5D278CC237EFD53A6156EABAFBE98A8F3CE99E854818822FA20C0FF46 E5B3805264B8CD085F0A8A9BD503A1304E9202D7304FF93541FB7FAA4629EE0BD7ED566F610DCD047721AEA828DFECA651087CDE5AF95C125793D4CD8E83B801D E171335A866D7E31F1473BFC93EBFD994326C0FE97ACB8DA722F788EA27B8D9E15E8E7B6EF772AB7534060F2BCAF1C3E82645235C9D1857B0790C2

Table 53

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 54

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 55

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 56

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

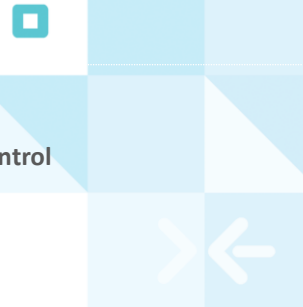
Table 57

3.6.2.31 Invoice Issuer Name

Issuer’s name.

Data type	string
Length	100 characters
Example	Name Surname

Table 58



3.6.2.32 Invoice Issuer Address

Issuer's address.

Data type	string
Length	100 characters
Example	Plaza Tirana 1

Table 59

3.6.2.33 Invoice Issuer Town

Issuer's town.

Data type	string
Length	100 characters
Example	Tirana

Table 60

3.6.2.34 Invoice Issuer Country

Issuer's country.

Data type	string
Length	100 characters
Example	Albania

Table 61

3.6.2.35 Invoice Buyer

XML element representing an 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 62

3.6.2.37 Invoice Buyer Name

Buyer's name.

Data type	string
-----------	--------

Length	100 characters
Example	Name Surname

Table 63

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 64

3.6.2.40 Invoice Buyer Country

Buyer's country.

Data type	string
Length	100 characters
Example	Albania

Table 65

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 Item Name

Name of the item.

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

Table 66

3.6.2.43 Invoice 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 67

3.6.2.44 Invoice Item Unit

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

Data type	string
Max length	50
Example	kg

Table 68

3.6.2.45 Invoice Item Quantity

Amount or number (quantity) of items.

Data type	double
Example	3.5

Table 69

3.6.2.46 Invoice Item UnitPrice

Price of the one item.

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

Table 70

3.6.2.47 Invoice Item Rbt

Rebate percentage.

Data type	double
Example	12

Table 71

3.6.2.48 Invoice Item IsRbtRed

Is rebate reducing base price?

Data type	boolean
Values	true, false
Example	true

Table 72

3.6.2.49 Invoice Item PriceBefVAT

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 73

3.6.2.50 Invoice Item VATRate

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 74

3.6.2.51 Invoice Item VATAmt

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

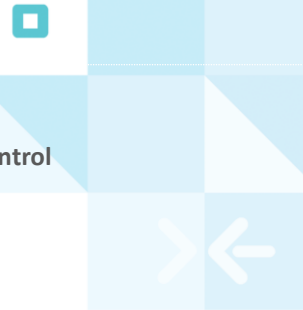
Table 75

3.6.2.52 Invoice Item PriceAftVAT

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 76



3.6.2.53 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.

3.6.2.54 Invoice SameTaxItems NumOfItems

Number of items of same tax rate.

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

Table 77

3.6.2.55 Invoice SameTaxItems PriceBefVAT

Price of the item before VAT

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

Table 78

3.6.2.56 Invoice SameTaxItems 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 79

3.6.2.57 Invoice SameTaxItems VATAmt

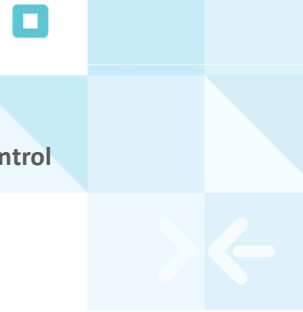
VAT amount for items from same tax group.

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

Table 80

3.6.2.58 Invoice ConsTaxItems

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



3.6.2.59 Invoice ConsTaxItems NumOfItems

Number of items under the consumption tax.

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

Table 81

3.6.2.60 Invoice ConsTaxItems 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 82

3.6.2.61 Invoice ConsTaxItems ConsTaxRate

Consumption taxrate.

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

Table 83

3.6.2.62 Invoice ConsTaxItems ConsTaxAmt

Consumption taxamount.

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

Table 84

3.6.2.63 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.

Fiscalization Service
Improvement of the Management Module of the Billing Control
Technical Specification (DRAFT)

	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 85

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.

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 86

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.

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 87

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 88

3.6.3.5 FIC

Element represents unique number 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 89

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 [3.9](#).

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 **Error! Reference source not found.**).

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).
Check format for cancelled invoice	Check that IIC reference is provided when invoice type is CANCELLED
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 paid in cash. It can only be paid by cash if its amount is at most ALL 150,000. The amount is subject to change.

Table 90

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 CPCMC. There will be other controls implemented as part of anomaly management which are not listed here.

Description should be available in multiple languages.

Control ID	Control Name	Control Description
-10	Valid TIN of invoice issuer	TIN is valid and in the register of active taxpayers
-11	Valid TCR	Code of TCR should be valid and TCR should be active. It should belong to listed business unit of the invoice issuer.
-12	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.
-13	Valid operator code	Operator code is valid in the moment of invoice issuing and the operator is assigned to the invoice issuer
-14	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.
-20	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.
-30	Corrective invoice reference	Referenced invoice exists
-31	Corrective invoice correction check	Corrections to the original invoice are invalid. Negative prices from all corrective invoices of the same original invoice should be compared to positive prices of original invoice. Items should be paired and grouped by item unit price. This pairing is by approximation because it is hard to pair the items from the invoices. In the error description there should be problematic invoice.
-32	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

Table 91

3.6.7.1 Request XML

45 | 68

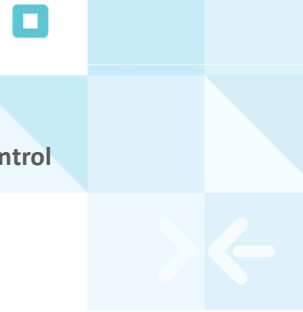
46 | 68

**Fiscalization Service
Improvement of the Management Module of the Billing Control**

Technical Specification (DRAFT)

	Note	Element	[1, 1]	XML element representing header of the note containing data about the message sent.
	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.
	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]	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.
	Items	Element	[1,1]	XML element representing list of items.
	Name	Attribute	[1,1]	Name of the item (goods or services).
	Code	Attribute	[0,1]	Code of the item from the barcode or similar representation.
	Unit	Attribute	[1,1]	What is the item's unit of measure (piece, weight measure, length measure...).
	Quantity	Attribute	[1,1]	Amount or number (quantity) of items

Table 92



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.

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

3.7.1.3 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 94

3.7.1.4 Note

XML root element representing the note.

3.7.1.5 Note Issuer

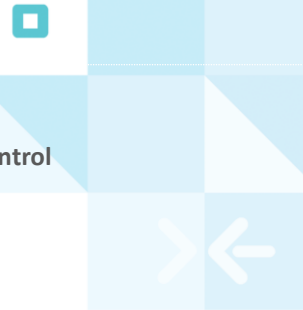
XML element representing the issuer of the note.

3.7.1.6 NoteIssuer 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 95



3.7.1.7 Note Issuer Name

Name of the note issuer.

Data type	String
Length	100 characters
Pattern	[a-zA-Z]
Example	Name Surname

Table 96

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 97

3.7.1.9 Note WTNum

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 98

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-zA-Z]{1}[0-9]{8}[a-zA-Z]{1}
Example	K72001008V

Table 99

3.7.1.11 Note BusinUnit

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

Fiscalization Service
Improvement of the Management Module of the Billing Control
Technical Specification (DRAFT)

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 100

3.7.1.12 Note SoftNum

Number of the software used for invoice issuing.

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 101

3.7.1.13 Note StartAddr

Starting address of the transportation.

Data type	string
Length	100 characters
Example	Street Name 888

Table 102

3.7.1.14 Note StartCity

City where the transportation started in.

Data type	string
Length	100 characters
Example	Tirana

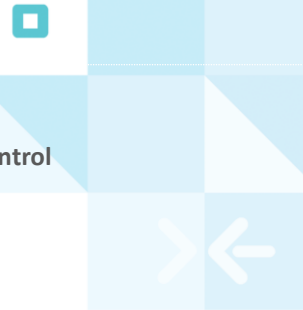
Table 103

3.7.1.15 Note DestinAddr

Destination address of the transportation.

Data type	string
Length	100 characters
Example	Street Name 888

Table 104



3.7.1.16 Note DestinCity

City where the transportation will end.

Data type	string
Length	100 characters
Example	Tirana

Table 105

3.7.1.17 Note WTNIC

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 106

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 107

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 108

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 109

3.7.1.21 Note VehPlates

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

Table 110

3.7.1.22 Note Items

XML element representing list of items on the note.

3.7.1.23 Note Items Name

Name of the item.

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

Table 111

3.7.1.24 Note Items 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 112

3.7.1.25 Note Items Unit

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

Data type	String
Max length	50 characters
Example	Kg

Table 113

3.7.1.26 Note Items Quantity

Amount or number (quantity) of items.

Data type	double
Example	3.5

Table 114

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 sent 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.
WTNIC	Element	[1, 1]	CIS generated verification code that can be used to uniquely identify registered note.
Signature	Element	[1, 1]	XML element with signature.

Table 115

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.

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 116

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.

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 117

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 118

3.7.2.5 WTNIC

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 119

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 3.8.

3.7.4 EXAMPLE XML

3.7.4.1 Request XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sch="http://alimc.neos.hr/FiscalizationService/schema" xmlns:xd="http://www.w3.org/2000/09/xmldsig#">
  <soapenv:Header/>
  <soapenv:Body>
    <WTRRequest xmlns="http://alimc.neos.hr/FiscalizationService/schema" xmlns:ns2="http://www.w3.org/2000/09/xmldsig#" Id="Request">
      <Header SendDateTime="2019-06-14T11:00:42+02:00" UUID="9f4459da-e558-4cbc-bbff-217667096451"/>
      <NoteBusInUnit="07790411-b967-454e-8e7c-a8154d719020" DateTimeCreated="2019-06-14T11:00:42+02:00" WTNIC="093FB373E719C5CDFEF23E5C3D415B7C" WTNNum="322019" IsAfterDel="false" OperatorCode="K72001008V" SoftNum="2191a92e-b3b1-44f5-a2eb-84abc817264f" StartAddr="Address 1" StartCity="Tirana" DestAddr="Address 2" DestCity="Shkoder" TransDate="Shkoder" CarrierId="123123HKJHK" VehPlates="TI-2099">
        <Issuer NUIS="K72001009V" Name="Name"/>
        <Items>
          <Item Code="123451" Name="Kola1" Quantity="1.0" Rbt="1.0" Unit="kg"/>
          <Item Code="123451" Name="Kola1" Quantity="1.0" Rbt="1.0" Unit="kg"/>
          <Item Code="123451" Name="Kola1" Quantity="1.0" Rbt="1.0" Unit="kg"/>
          <Item Code="123451" Name="Kola1" Quantity="1.0" Rbt="1.0" Unit="kg"/>
        </Items>
      </Note>
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#"><SignedInfo><CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" /><SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" /><Reference URI="#Request"><Transforms><Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /></Transforms></Reference></SignedInfo></Signature>
    </WTRRequest>
  </soapenv:Body>
</soapenv:Envelope>
```


3.7.4.2 Response XML

55 | 68

</Signature> </WTNResponse> </S:Body> </S:Envelope>	
--	--

3.8 ERROR MESSAGES

In case of an error in the processing 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 code XML element which represents numeric error code.

3.8.1 XML FORMAT

Name	Occurrence [Min, Max]	Description
header	[1, 1]	XML root element representing the header of the error message.
UUID	[1, 1]	UUID generated by a CIS for every error message created.
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.
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 120

3.8.1.1 Header

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

3.8.1.2 Header UUID

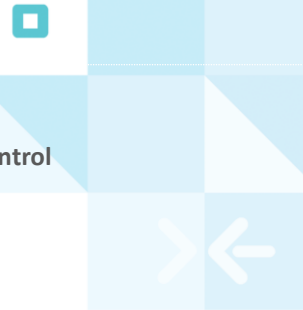
This is an attribute that uniquely describes the message and gives it the unique identification.

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 121

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 122

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 123

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 124

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 chapter [3.9.2](#).

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

Table 125

3.8.2 ERROR CODES

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

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.
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.
40	Client	IIC already exists in the database.
41	Client	Invoice amount too large to be payed by cash.
900	Server	Service property file not found.
901	Server	Exception occurred during extracting properties for the property file.
902	Server	Exception occurred during loading XSD from a file.
903	Server	Exception occurred during loading keystore.
904	Server	Exception occurred during key extraction from the keystore.
905	Server	Exception occurred during certificate extraction from the keystore.
910	Server	Exception occurred while extracting constructed XML response message during signing.
911	Server	Signature XML element not found during signing XML response message.
912	Server	Exception occurred during signing of the XML response message.
920	Server	Exception occurred during inserting received XML request.
921	Server	Database JNDI not found.

Fiscalization Service
Improvement of the Management Module of the Billing Control
Technical Specification (DRAFT)

922	Server	Exception occurred during accessing database with provided JNDI.
999	Server	Undefined exception.

Table 126

3.8.3 EXAMPLE XML

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <S:Fault>
      <faultcode>S:Client</faultcode>
      <faultstring>[40] Invoice with IIC 093FB373E719C5C0FEF23E5C3D415B7C already exists.</faultstring>
      <detail>
        <code>40</code>
      </detail>
    </ns0:Fault>
  </S:Body>
</S:Envelope>
```


4. Data export of the non-fiscalized invoices

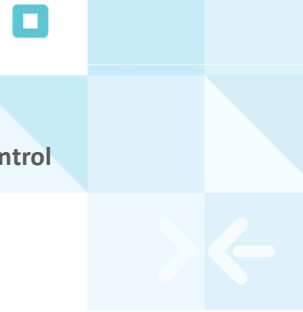
There are cases where invoice fiscalization will not be possible at the moment of invoice issuing. Reason for that is lack of the internet connection because of issuer being in remote areas or because of giving its services in places where having internet connection is not optional or it is difficult.

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.



5. Security

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

5.1 PKI

AKSHI will be the issuer of digital certificates for fiscalization service. For this purpose, three types of certificates shall be issued:

- Certificate for transport security with common name *.tatime.gov.al issued to service.
Used to secure transport between the service and an issuer.
- Certificate for message security issued to the service.
Used to digitally sign response data messages.
- Certificate for message security issued to an issuer.
Used to digitally sign request data messages and IIC data elements.

Every certificate is issued by the following certificate authorities:

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

Procedure for acquiring certificates for signing request data messages and IIC data elements can be found inside a **Error! Reference source not found.** document.

5.2 TRANSPORT SECURITY

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

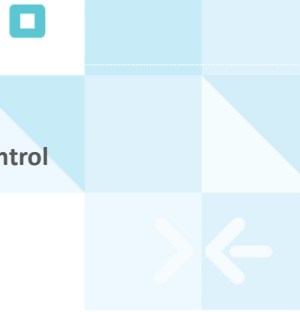
5.3 MESSAGE SECURITY

To ensure unambiguous identification of the taxpayer and to provide unchanged content, each request data message and IIC 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 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 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/xmlsig-core/>. Additional description is available in the chapter 5.3.1.

IIC data element is created and digitally signed according to the custom cryptographic algorithm described in the chapter 5.3.2.



5.3.1 REQUEST AND RESPONSE DATA MESSAGE 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/xmlsig-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/xmlsig#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/xmlsig-more#rsa-sha256>

5.3.2 IIC DATA ELEMENT

IIC, Issuer's Invoice Code is 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
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(Chapter 3.6.2.10)
- Business unit number(Chapter 3.6.2.21)
- 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 to UTF-8 encoding. Parameters are concatenated with pipe character UTF-8 with decimal code 124.

For example, for parameters:

- Issuer NUIS: K72001009V
- Date and time created: 2019-06-12T17:05:43+02:00
- Invoice number: 9952
- Business unit number: 4c31c6d1-74e4-4645-b209-6384e6de7b30
- Cash register number: 9380559f-46ff-4837-9fce-008abeac99ba
- Software number: 71a3db25-fef3-4194-a567-6bd079eb32f0
- Total price: 99.01

Resulted concatenated value is: K72001009V|2019-06-12T17:05:43+02:00|9952|4c31c6d1-74e4-4645-b209-6384e6de7b30|9380559f-46ff-4837-9fce-008abeac99ba|71a3db25-fef3-4194-a567-6bd079eb32f0|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: K72001009V|2019-06-12T17:05:43+02:00|9952|4c31c6d1-74e4-4645-b209-6384e6de7b30|9380559f-46ff-4837-9fce-008abeac99ba|71a3db25-fef3-4194-a567-6bd079eb32f0|99.01
- PEM encoded private key:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAA6zOR5ItNYHJNVmx1jZtd/KQUyGIZbnIJ8IWqcEesktRV5FF
HviQZsx2DpyeVQTu/Kel9Xh+Z6OZ6t5sADzfYnkwCrSb0FhT+01m2PIHaIUZhVtc
ppn0gxNWfgzW4sTvTyrYk601Kxymxs/rck/WRQB1mp68au8mgGMzGukHfL7Wk4jO
U5VD3H1StBx1MjVW+soN5GUL/rWGaYun6Zsn9aYYEuJb0hKvKDy8n0tNIS69dqqd
piZAKvdh9sYdF1ElgXZhdmsGURMm6OcePUPZO/HFKq7R1K6vIxXVI61906tWt+G
uhul8e0x2VTwbTdpwG4FpdFUTqUDK6cswH0HTQIDAQABAoIBAQCqBWJuUqDBmn76
ULMM1YzWjFAUFpkmdikRTIVzew4ElutbMIF7Sr9lMm2sFLoZKOZ8lr0wqalpqc
GFT8KwTU04SWDUICTwbuF7pcE0F1tdmIBE5KhLozUnRQTF1WHkRb9z40I+Zf3ttG
W0mpHbtNr/hTqHHN30j2wD7+MfvemPbcAvu9JLCYUzUZ06qxUwAjyFgsW7YyLa0a
qFB0Q0Yc6RsLvoSFXW0M5ghdtgoZvl+ayt4fgz1L3FjAMuXoLEX/778VA92/NZ0Q
mzQdKTT6B4Pm5s8XrY90hLlsYqKuyR/aoSHC/anSLw0yJ/5Gis2gmCwo3a7+PEYy
LUN7C0yFAoGBAPhgYufTkdod5PqG/SCEE2i6pjK0ZnuIUu9f2cmhxnvYChlig2wk
oDWUSGuXwItNF+X7j3XoZz8FNJcriK7KP2UPDOWP0ZvxZgZEcmmwt27x1vVjzjCG
s10w5fF0363hhtX35Jq21VZGbN1LpIoEZgCeS/nBs+9DcRjDoXliKwFHAoGBAPJr
qSWLV03gIG1wikXBWCYZUTSzs06NWfxcWPHKTNKVR0ifBTK23zuZ6gg1uNqLz/Ae
64ZwssMoIViyXE01XMPP8io4QidyVED2n70pjrVcUVYyr9IwKmhmbNBFKFMof05f
NV29P1Am1Jqv2EQi5jE/BbBu9kLifs2YyGBAn/ZLAoGAVsLsqciZAVVCAFWZJHue
gA37NK5eQja7qcyUuj9dozxIVNe5ytp8dtrmdVccNkzm1TqLwYc+UaBS35+gblZN
0NJyEdqsQMoRdo0AX1PuVb369ds4UnEq6yzC1gmUTxwhyqp+W6D+B5YwPx1GT8P7
kam6JnOIIEK9xgXIaStmBU8CgYB6RwXVszcOmYuhyC9mygSNix2j6LNpUJFAMtCG
fZYeRBMobvWwRADLznH21Bgu3HDxXJd0g9AXkk1kbZSTOURmXKB43VG5Ffke5t3i
C3E5V6yLPxvieHsa9B5hlG4BrB6yyGFhVBCQfFwNBOWgUL4tvu0+tmnvCRI04G7J
5i8JiWKBgQCQHTfRrGaEsq1BG7zPQSQo9q5cxL8WzYd0sTs3FDcwCtHqxBEQ3rr
O/1+HvRa+y6ZEH6q4pREewTIymfv9tmGxVe3f8zrKGR5litvN6OnZuWJdq57Y11N
J1sdpMxTtxQQmexsADif+QByCvdeFKE5C3veMLdgS5I6HTMN9k51aA==
-----END RSA PRIVATE KEY-----
```


Resulting signature value is:

```
B2C218486302EC553EE1AB9124E1A14705742E870E8872EF34E63617AB252E189ACDF7A3E3F5C82061FFFF8AC2826A
5588596A8807F648410899B6193F77F4BDCDFA87553A62079A2EF9E6E6F0B8DA1038968D2FCB920B580EBF33ACEED
FEA0DAA78067F916ADC5D278CC237EFD53A6156EABAFBE98A8F3CE99E854818822FA20C0FF46E5B3805264BBBCD085
FOA8A9BD503A1304E9202D7304FF93541FB7FAA4629EE0BD7ED566F610DCD047721AEAA828DFECA651087CDE5AF9
5C125793D4CD8E83B801DE171335A866D7E31F1473BF0C93EBFD994326C0FE97ACB8DA722F788EA27B8D9E15E8E7B
6EF772AB7534060F2BCAF1C3E82645235C9D1857B0790C2
```

5.3.2.3 Calculate digest

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

For example, for avalue:

- Signature value:
B2C218486302EC553EE1AB9124E1A14705742E870E8872EF34E63617AB252E189ACDF7A3E3F5C82061FFFF8
AC2826A5588596A8807F648410899B6193F77F4BDCDFA87553A62079A2EF9E6E6F0B8DA1038968D2FCB920
B580EBF33ACEEDFEA0DAA78067F916ADC5D278CC237EFD53A6156EABAFBE98A8F3CE99E854818822FA20C
0FF46E5B3805264BBBCD085F0A8A9BD503A1304E9202D7304FF93541FB7FAA4629EE0BD7ED566F610DCD047
721AEAA828DFECA651087CDE5AF95C125793D4CD8E83B801DE171335A866D7E31F1473BF0C93EBFD99432
6C0FE97ACB8DA722F788EA27B8D9E15E8E7B6EF772AB7534060F2BCAF1C3E82645235C9D1857B0790C2

MD5 digest value is: C701FB4839E7D2C3D8DBC81BBAC06164

6. Code examples

This chapter covers the code examples for specific actions.

6.1 IIC CALCULATION 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.

```
package hr.neos.alimc.fiscalization.utility;

import java.io.ByteArrayInputStream;
import java.io.StringWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.math.BigDecimal;
import java.security.Key;
import java.security.KeyStore;
import java.security.PrivateKey;
import java.security.Signature;
import java.util.GregorianCalendar;
import java.util.UUID;

import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBElement;
import javax.xml.bind.Marshaller;
import javax.xml.datatype.DatatypeConstants;
import javax.xml.datatype.DatatypeFactory;
import javax.xml.datatype.XMLGregorianCalendar;
import javax.xml.namespace.QName;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.w3c.dom.Document;

import hr.neos.alimc.fiscalization.model.SigningKey;
import hr.neos.alimc.fiscalization.util.SignUtil;
import hr.neos.alimc.fiscalization.wsd1.BuyerType;
import hr.neos.alimc.fiscalization.wsd1.ConsTaxType;
import hr.neos.alimc.fiscalization.wsd1.InvoiceItemType;
import hr.neos.alimc.fiscalization.wsd1.InvoiceItemsType;
import hr.neos.alimc.fiscalization.wsd1.InvoiceSType;
import hr.neos.alimc.fiscalization.wsd1.InvoiceType;
import hr.neos.alimc.fiscalization.wsd1.IssuerType;
import hr.neos.alimc.fiscalization.wsd1.PaymentMethodSType;
```



```
import hr.neos.alimc.fiscalization.wsd1.RegisterInvoiceRequestHeaderType;
import hr.neos.alimc.fiscalization.wsd1.RegisterInvoiceRequestType;
import hr.neos.alimc.fiscalization.wsd1.SameTaxType;
import hr.neos.alimc.fiscalization.wsd1.SelfIssSSType;

import org.apache.commons.codec.digest.DigestUtils;

public class GenerateIicForDocument {

    public static void main(String[] args) {
        String issuerNuis = "K72001009V";
        String concat = issuerNuis;
        String dateTimeCreated = "2019-06-12T17:05:43+02:00";
        concat = concat + dateTimeCreated;
        String invoiceNumber = "9952";
        concat = concat + invoiceNumber;
        String busiUnit = "4c31c6d1-74e4-4645-b209-6384e6de7b30";
        concat = concat + busiUnit;
        String cashRegister = "9380559f-46ff-4837-9fce-008abeac99ba";
        concat = concat + cashRegister;
        String softNumber = "71a3db25-fef3-4194-a567-6bd079eb32f0";
        concat = concat + softNumber;
        String totalPrice = "99.01";
        concat = concat + totalPrice;
        byte[] signed = null;

        try {
            FileInputStream file_inputstream = new FileInputStream
("C:\\_dev\\Certificates\\selfca\\keystores\\fiskalX.neos.hr\\fiskal1.p12");

            KeyStore keyStore = KeyStore.getInstance("PKCS12");
            keyStore.load( file_inputstream, "welcome1".toCharArray());
            Key privat = keyStore.getKey("1", "welcome1".toCharArray());
            Signature notary = Signature.getInstance ("signature");
            notary.initSign ((PrivateKey) privat);
            notary.update (concat.getBytes());
            signed = notary.sign();
        } catch (Exception e) {
            e.printStackTrace();
        }

        String iicPrint = DigestUtils.md5Hex(signed);
        System.out.println ("The IIC is: " + iicPrint);
    }
}
```

6.1.2 C# EXAMPLE

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

```
using System;
```



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

namespace Iic
{
    class GenerateIic
    {
        static void Main(string[] args)
        {
            string issuerNuis = "K72001009V";
            string concat = issuerNuis;
            string dateTimeCreated = "2019-06-12 17:05:43";
            concat = concat + dateTimeCreated;
            string invNumber = "9952";
            concat = concat + invNumber;
            string busiUnit = "4c31c6d1-74e4-4645-b209-6384e6de7b30";
            concat = concat + busiUnit;
            string cashRegister = "9380559f-46ff-4837-9fce-008abeac99ba";
            concat = concat + cashRegister;
            string softNumber = "71a3db25-fef3-4194-a567-6bd079eb32f0";
            concat = concat + softNumber;
            string totalPrice = "99.01";
            concat = concat + totalPrice;

            byte[] signed = null;
            try
            {
                X509Certificate2 certificate = new X509Certificate2("c:\\certificate.p12",
                    "welcome1");
                RSACryptoServiceProvider rsa = (RSACryptoServiceProvider)certificate.PrivateKey;
                byte[] data = Encoding.ASCII.GetBytes(concat);
                signed = rsa.SignData(data, new SHA1CryptoServiceProvider());

                MD5 md5Hash = MD5.Create();

                string concatPrint = GetMd5Hash(md5Hash, signed);

                Console.WriteLine("The IIC is: " + concatPrint);
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
    }
}
```



```
static string GetMd5Hash(MD5 md5Hash, byte[] input)
{
    byte[] data = md5Hash.ComputeHash(input);
    StringBuilder sBuilder = new StringBuilder();
    for (int i = 0; i < data.Length; i++)
    {
        sBuilder.Append(data[i].ToString("x2"));
    }
    return sBuilder.ToString();
}
```