Technical Instructions for POS and Cash Register Developers

Version 2.7

Change Log

Author	Change	Date
Ivan Pavlović DTI	Review of the initial version of the document	22 Aug 2017
Ivan Pavlović DTI	1. Detailed description of Invoice Request and Invoice response are added to JSON protocol 2. Json protocol examples are updated to include Options and PAC 3. Serial protocol field descriptions are updated to provide a better explanation of usage 4. Interpretations are updated 5. Procedure for obtaining digital certificates is detailed	04 Sep 2017
Ivan Pavlović DTI	1. GUID, RS232 and USB are added to the Interpretation section 2. Date and time formats are described in the Data Structures section 3. Anatomy of Fiscal Invoice – descriptions are updated 4. Invoice format and examples are updated for Json-based protocol – Options are added to the InvoiceFiscalizationRequest	15 Sep 2017
DTI	1. Invoice sample changed 2. Accepted syntax changes made by Peni	25 Sep 2017
Lena Stamenković	Revision and update	09 Oct 2017
Srboslav Subotic DTI	1. Added Status and Error Codes 2. Common JSON Based Protocol for both HTTP and Serial connection 3. Added Hash field into Sign Invoice request and response 4. Added commands Attention and Get Signed Invoice 5. POS to E-SDC Communication over Serial Port is JSON based	10 Oct 2017
Lena Stamenković Srboslav Subotic DTI	1. Added description for tax labels calculation	19 Oct 2017
Srboslav Subotić	1. Additional error codes added	13 Nov 2017
Lena Stamenkovic	1. Online POS and V-SDC integration	14 Nov 2017
Lena Stamenkovic	1. Rename Cashier and Total Amount fields in the textual representation of the invoice	16 Nov 2017
Lena Stamenkovic	1. Cashier TIN optional changed to mandatory 2. Authority > Service	29 / 30 Nov 2017

Ivan Pavlović	1. SDC returns four additional fields in **InvoiceFiscalizationResult**	01 12 2017
Lena Stamenkovic	1. Province > District	08 Dec 2017
Lena Stamenkovic	Swagger > OpenAPI-Specification	18 Dec 2017
Ivan Pavlović	- ReferentDocumentDateAndTime is added are required field for Copies and Refunds	02 Jan 2018
Srboslav Subotić	- Added field UnitPrice to InvoiceFiscalizationRequest - Added decimal format for InvoiceFiscalizationRequest	18 Jan 2018
Srboslav Subotić	- Added description for MRC in `InvoiceFiscalizationResonse` - Added Payment Method in Receipt example	22 Jan 2018
Lena Stamenkovic Srboslav Subotić	1. Added generic error codes 2. Assigned obsolete error codes 3. Deleted irrelevant error codes	29 Jan 2018
Lena Stamenkovic Srboslav Subotić	- Minor additional explanation regarding buyer TIN and BuyerCostCentar ID	01 Feb 2018
Srboslav Subotić	Fixed sample code for Online POS (`"PaymentType"`)	09 Feb 2018
Nemanja Đordjević	Fixed footer link for Online POS to minimized js	09 Feb 2018
Srboslav Subotić	• Removed field ReferentDocumentDateAndTime • Added Fiscal Invoice definition • Anatomy of Fiscal Receipt improved definition and description • Normal Refund Receipt improved definition and description • Training or Proforma Receipt definition and description • Connected Scenarios improved description and sample code • JSON Based Protocol - an improved common explanation for SignInvoice • JSON Based Protocol (POS to E-SDC Specific) - renamed	09/14 March 2018
Lena Stamenkovic	Message received from V-SDC	03 June 2018.
Lena Stamenkovic	DateTimeDisplayFormat - "dd/MM/yyyy HH:mm:ss" (e.g. 15/10/2018 14:28:24)	15 Oct 2018
Ivan Pavlović	• Non-Empty Collection of Tax Labels is required field for each Invoice Item • Discount is removed as obsolete • UnitPrice is required field	18 Jan 2018
Lena Stamenkovic	• Get Signed Invoice Command > Get Last Signed Invoice Command (added Last)	05-02-19

Lena Stamenkovic	• Added: Microsoft Internet Explorer 11 (not supported by Microsoft) • Added GTIN example in Invoice Request	11-04-19
Lena Stamenkovic	• Added explanation about invoice types	23-04-19
Lena Stamenkovic	• Recommendation on line items number added to section Sign Invoice Command/ invoice request / table Data fields	11-07-19
Srboslav Subotić	• Added Card Locked error code	23 July 2019
Ivan Pavlović	• Get Tax Authority Params method is added to the V-SDC and E-SDC	4. Sept 2019
Ivan Pavlović	• Reference (Document) Number is updated to better explain cases in which it is a required field	14. Nov 2019
Lena Stamenkovic	• Added section explaining the difference between E-SDC and V-SDC	05-12-19
Aleksandar Radovanovic	• Extended Iso 8061 format examples and information under Date And Time section	12-12-2019
Nenad Nikić	• Added how to extract VSDC endpoint from certificate and list of known endpoints in case they do not exist in certificate.	17-12-2019
Aleksandar Radovanovic	• Added Tax Calculation section	14-01-2020
Nenad Nikic Marko Denic	• Added Environments • Updated URLs	06-03-2020

Table of Contents

Introduction

Each POS, Cash register, ERP or invoice generation software (Accredited POS) shell be able to connect to a V-SDC or E-SDC and issue a fiscal invoice. Accredited POS are developed for different software and hardware platforms, designed to use a variety of communication standards to connect to other software or hardware components. As wide acceptance and low cost of integration are crucial for the success of fiscalization a Tax Service is dedicated to providing a detailed integration instruction for all manufacturers and software developers.

This document gives technical guidelines and instructions for the implementation of Accredited POS and integration with TaxCore V-SDC service or E-SDC devices. These Instructions set standards enabling a seamless integration of a third-party Accredited POS or E-SDC with the TaxCore.

V-SDC service shall be widely available and accessible from the variety of Accredited POS devices and software solutions.

Interpretations

Accredited POS (Accredited POS) is a computer program, electronic device or information system for issuing of receipts, in compliance with the requirements of the Regulation.

Electronic Fiscal Device (EFD) is composed of an Accredited POS and an E-SDC/V-SDC connected into one system. The EFD produces fiscal receipts and reports audit data to a Tax Service.

GUID is a Globally Unique Identification Number. In its canonical textual representation, the sixteen octets of a UUID are represented as 32 hexadecimal (base 16) digits, displayed in five groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters (32 alphanumeric characters and four hyphens). For example: 123e4567-e89b-12d3-a456-426655440000.

HTTP (Hypertext Transfer Protocol) is an application protocol for distributed, collaborative, and hypermedia information systems.

HTTPS is a communications protocol for secure communication over a computer network which is widely used on the Internet. HTTPS consists of communication over Hypertext Transfer Protocol (HTTP) within a connection encrypted by Transport Layer Security.

Invoice, see Receipt.

Public Key Infrastructure (PKI) is a set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates and manage public-key encryption.

Receipt is a digitally signed acknowledgment that a specified payment has been received. A receipt records the sale of goods or a service fee. In this Law, receipt is used interchangeably with the term invoice.

RS232 is a standard for serial communication transmission of data. It formally defines the signals connecting between a DTE (data terminal equipment) such as a POS, and a DCE (data circuit-terminating equipment or data communication equipment), such as an E-SDC

Sales Data Controller (SDC) contains a Secure Element and is used to generate an invoice by signing a request received from an Accredited POS and to produce audit data. It stores audit data to its own internal memory and enables local and remote audit. There are two implementations of SDC:

a) **External SDC (E-SDC)** is a black box that contains the Secure Element and enables semi-connected fiscalization scenarios
b) **Virtual Sales Data Controller (V-SDC)** is a web service operated by the Tax Service that exposes an SDC functionality to authorized taxpayers via the Internet. It contains and uses a Secure Element to sign invoices.

Secure Element (SE) is a fiscal component implemented as a special software or device designed to receive specific invoice data, to perform signing and data processing and to generate response data, sent back to the caller for further actions. The Response data provides the authenticity of an invoice data. The Secure Element is issued and controlled by a Tax Service. The main purpose of the Secure Element is to sign invoices using the taxpayer’s digital certificate, to control audits and maintain a set of fiscal counters.

TaxCore (Fiscalization System) is a set of web services, sites and database management software installed on the side of the Tax Service for communication with the Accredited POS and SE devices. **UID** – Unique Identifier (8 alphanumeric characters) assigned to each Smart card and embedded into the subject field of a digital certificate.

USB is an industry standard that defines cables, connectors and communications protocols for connection, communication, and power supply between computers and devices.

UTP is also the most common cable used in computer networking. Modern Ethernet, the most common data networking standard, can use UTP cables

Verification URL is a unified resource location used to verify a particular invoice using web service provided by the Tax Service.

Development Environment

The Sandbox environment is accessible to all registered developers of Accredited POS components. The Sandbox Environment exposes the same APIs and uses the same protocols as a Production environment.

Obtaining Test Certificates

As a developer of Accredited POS one can register on the Tax Service web site to obtain a set of test certificates and technical documentation. Test certificates shall enable a user to test both successful and failing scenarios, like trying to fiscalize an invoice with an expired certificate.

Obtaining Test SDC

Tax Service shall publish a notification to all interested parties.

Obtaining Smart Cards

Accredited POS Vendors shall apply to the Tax Service and get test smart cards and digital certificates in PKCS 11 format to use for development, integration and testing purposes. In order to achieve that, follow these steps:

1. Primary contact registers with Tax Service using application form published on web site
2. Tax Service verifies the application and sends the enrolment request to primary contact by e-mail or SMS
3. Primary contact enters the desired PIN code for their smart card
4. Tax Service delivers the smart card to the Primary contact
5. If additional smart cards are required, the Primary contact can submit a request using Taxpayer Admin portal in the staging environment (web address will be published on Tax Service Web Site)

Identification of Environment

The staging environment is a testing environment that exactly resembles a production environment and is used to test all the installation/configuration/migration scripts and procedures before they're applied to a production environment.

The production environment is also known as *live*, particularly for servers, as it is the environment that users directly interact with.

The demo environment is primarily for showcasing the product to key stakeholders and potential or existing customers.

A sandbox is a testing environment that isolates untested code changes and outright experimentation from the production environment or repository.

NOTE: Development, testing and technical review are all done in the Sandbox environment. The next step is the administrative review process which enables getting accreditation for a specific country. Different countries have different system names, and every environment has its own URL.

System name by country/state:

1. Fiji - VMS
2. Samoa – TIMS
3. Washington – TaxCore

For this reason, URLs and names in your documentation, code and UI should not be hardcoded but configurable or extracted from digital certificate (in case of connected scenario).

High-Level Architecture of TaxCore

The electronic monitoring system for billing is an initiative undertaken by many countries for the purpose of reducing the grey economy and tax evasion. An important and new component of this initiative is the certified systems put in place for taxpayers to electronically secure each transaction at the moment of sale.

TaxCore is built as a set of semi-connected services exposing public APIs to enable the integration of third-party solutions into the fiscalization ecosystem.

In order to have a true picture of taxpayer's business transactions and be able to expand the tax base and finance national needs, we are building an electronic invoicing system that will be used by taxpayers at their business premises.

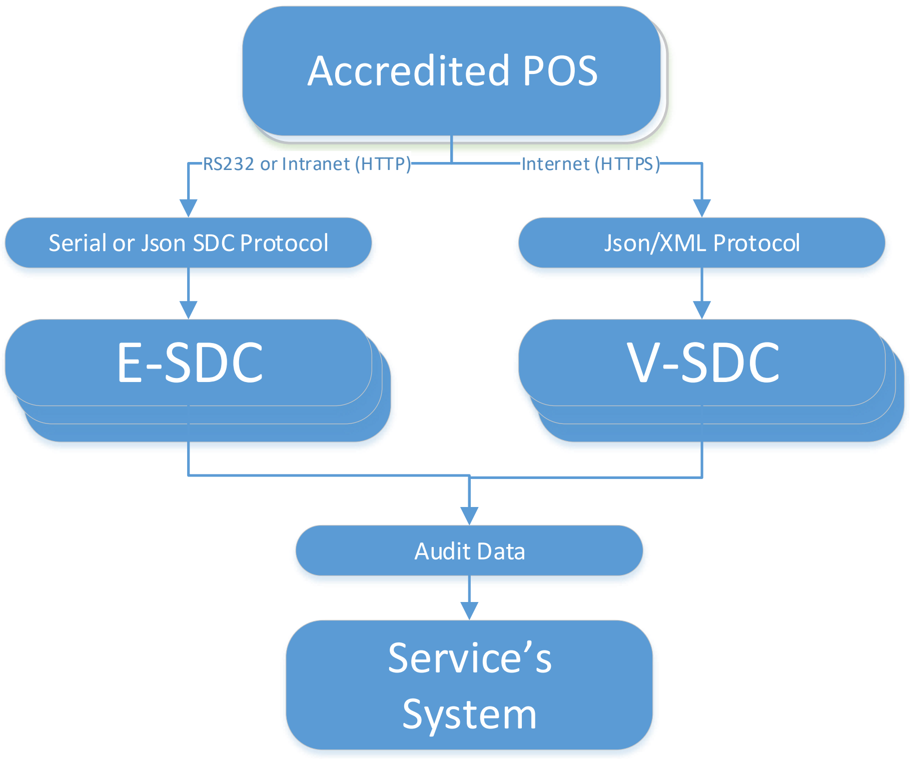


Figure 1 High-Level Architecture of TaxCore

This document will describe high-level requirements for all possible scenarios in order for Accredited POS to be compliant. In the chapter Clients, there will be given examples of different POS systems and preferred methods of integration with TaxCore.

Target audience are software developers and manufacturers of all software applications and hardware utilized to create invoices or receipts.

Data Formats

Date and Time

The date and time sent by POS to E-SDC or V-SDC is local time.

Date and time generated by E-SDC or V-SDC and printed on a receipt is local time (Fiji Winter Time is used as local time if E-SDC cannot track daylight saving time).

JSON based protocols use date and time according to ISO 8601 where applicable (for example: 2017-05-17T10:46:51.910Z).

Note: Date and Time display format on all TaxCore portals is: DateTimeDisplayFormat - "dd/MM/yyyy HH:mm:ss" (e.g. 15/10/2018 14:28:24)

Binary based protocols (Serial Protocol Communication) use Unix Timestamp format formatted as 64bit unsigned integer Big Endian (for example: 1495018011910 is 2017-05-17T10:46:51.910Z)

Fiscal Invoice

Fiscal invoice is, by definition, a digitally signed acknowledgment that a specified payment has been received or refunded. A fiscal invoice consists of two parts – Invoice Request and Invoice Response.

Invoice Request is created by an Accredited POS and it contains the usual invoice information like items, tax labels and invoice number. The invoice request is submitted by the Accredited POS using standard, publicly available protocol for communication to V-SDC/E-SDC, using the preferred technology of the POS system.

Invoice Response is generated by V-SDC or E-SDC after data validation. It is an integral part of any fiscal invoice. Without this information, an invoice could not be considered a legal fiscal invoice.

Each invoice is associated with one of the following invoice types:

- "Normal", as a result of a normal operation consisting of receiving the transmission of goods and provision of services;
- "Training", for limited use in the training environment. It may be generated on the basis of information from a simulated receipt in "Normal" fiscal mode;
- "Copy", re-issuing of a normal type receipt
- "Pro-forma", which has the characteristics of an original receipt. However, such receipts are not fiscally usable for proof of transmission of goods and services.

Each invoice type is associated with one of the following transaction types:

- sale;
- refund.

Each invoice type is associated with one of the following payment types:

- cash
- card
- check
- wire transfer
- mobile money
- voucher
- other.

Anatomy of Fiscal Receipt

A receipt records the sale of goods or the provision of a service. The table below explains the structure of a fiscal receipt. All elements are mandatory unless specified otherwise in the column Explanations. POS is free to print any content (coupons, logos, etc.) before the beginning and after the ending mark of the fiscal invoice.

Textual representation of Fiscal Invoice	Explanations
`============ FISCAL INVOICE ============`	**Title line** – marks the beginning of the fiscal part of receipt

`TIN:                          502579006`
`Company:                          Golf V`
`Store:                        Sun Store`
`Address:                    7 Someplace`
`District:                          Suva`
	Header data is provided by V-SDC or E-SDC during fiscalization of the invoice and returned to POS as part of the *InvoiceFiscalizationResult* object (explained in section Invoice Response).

`Cashier TIN:                 1234567890`	Cashier's identification. Local regulations might mandate POS to send data instead of cahier's name like Employee ID or some other information that uniquely identifies the POS cashier.

`Buyer TIN:                   5123456789`
`Buyer Cost Centre:                   123`
`POS number:                 POS2017/998`
`POS time:           15/6/2017 8:56:23AM`
	Buyer TIN is mandatory only in case of B2B transaction and in that case, it must be printed on the receipt. **Buyer Cost Center** is optional and reserved for further use, it must be present only for B2B transactions. **POS (Invoice) Number** and **POS (Invoice) time** are optional fields.

`Ref no:       P22VC8VR-JTJC5V65-114906`	**Reference (Document) Number** is required only for Refund or Copy transaction. In that case, **Ref no** must be printed on the receipt, containing **SDC Invoice No** of the document that is being copied or refunded in format RequestedBy-SignedBy-OrdinalNumber. If Copy or Refund is issued for the transaction that was recorded before the introduction of fiscalization POS should sent XXXXXXXX-XXXXXXXX-0 as the value of **Ref No** field. For any other invoice/transaction type (for example Normal Sale invoice referencing to Proforma Sale invoice) this field is optional.
`--------------NORMAL SALE---------------`	**Invoice** and **transaction type** description. Normal Sale and Normal Refund are the most common types. Other types of transactions and invoices are defined in section Fiscal Invoice.

`Items`
`========================================`
`Name       Price       Qty.        Total`
`Sport-100 Helmet, Blue (E)`
`          34.99          10       349.90`
`Mountain Bike Socks, M (A)`
`           9.03           4        36.12`
`HL Road Frame - Red, 58 (F, A)`
`        1431.50           2      2863.00`
`Plastic bag (P)`
`        0.10              5         0.50`
	List of items with **gross price**, tax labels, unit price and quantity.
	
	Tax Labels and their validity dates shall be published by Tax Service.

`----------------------------------------`
`Total Purchase:                  3249.52`
`Payment Method:                     Cash`
`========================================`
`Label        Name    Rate            Tax`
`E             STT    6.00%         19.81`
`A             VAT    9.00%        219.51`
`F            ECAL   10.00%        240.59`
`P              PB    0.10$          0.50`
`----------------------------------------`
`Total Tax:                        480.41`
	Total Purchase, **Tax items** and **Total Tax** are calculated by V-SDC or E-SDC during fiscalization of the invoice and are returned to POS as a part of the response.
	Payment Method: Cash, Card, Check, Wire Transfer, Voucher, Mobile Money, or Other.
	Taxpayer's tax liability is based on these tax amounts, calculated by V-SDC or E-SDC.
	
	Calculation is explained in the section Tax Amounts.

`========================================`
`SDC Time:            2017-06-15 08:56:25`
`SDC Invoice No:    7AF4D923-E3B30A31-234`
	Fiscal metadata added to the invoice through fiscalization.

`Invoice Counter:                230/234NS`
`========================================`
	SDC Invoice No - Combination of Requested By (7AF4D923), Signed By (E3B30A31) and Ordinal Invoice Number (234) is a system-wide unique identification of fiscal invoice. It may be used instead of the current receipt/invoice number generated by POS.
	SDC Time is the official date and time relevant to the tax calculation and reporting.
	Invoice Counter is generated by V-SDC or E-SDC and explained in section Invoice Response, field IC.

	QR Code contains an Invoice verification URL. QR Code also contains Internal data and the digital signature used for invoice verification. Invoice is verifiable by the customer immediately after fiscalization. In case the invoice/receipt is delivered as an electronic document (email), QR Code shall be substituted with Invoice verification URL in (clickable) hyperlink format. **NOTE:** This is just a sample QR code image, not an actual URL.
`======== END OF FISCAL INVOICE =========`	**Title line** – marks the end of the fiscal part of receipt
`This is custom Message`	Custom message returned from V-SDC or E-SDC

Normal Refund Receipt

Receipt for Normal Refund Invoice must contain visible markings “REFUND”, below the receipt header and above the item description section. Totals on the refund receipt are displayed as negative values, starting with (-), except for Total Purchase. Tax Items are displayed as positive values.

For Refund transaction type **Ref no** element is mandatory.

Example:

```
============ FISCAL INVOICE ============
TIN:                           502579006
Company:                          Golf V
Store:                         Sun Store
Address:                     7 Someplace
District:                           Suva
Cashier TIN:                   123456789
POS number:                89347415-2017
POS time:            2018-03-09 14:57:25
Ref no:           7AF4D923-E3B30A31-234
-------------NORMAL REFUND--------------
Items
========================================
Name     Price         Qty.        Total
Sport-100 Helmet, Blue (E)
        34.99          10        -349.90
Mountain Bike Socks, M (A)
         9.03           4         -36.12
----------------------------------------
Total Purchase:                   386.02
Payment Method:                     Cash
========================================
Label         Name    Rate           Tax
E              STT    6.00%        19.81
A              VAT    9.00%         2.98
----------------------------------------
Total Tax:                         22.79
```

```
========================================
SDC Time:            2018-03-09 14:57:46
SDC Invoice No:    7AF4D923-E3B30A31-235
Invoice Counter:                 4/235NR
========================================
---- QR code omitted for simplicity ----
======== END OF FISCAL INVOICE =========
```

Training or Proforma or Copy Receipt

Receipt for Training or Proforma or Copy Invoice must contain visible markings “TRAINING” or “PROFORMA” or “COPY”, below the receipt header and above the item description section.

Receipt must also contain **“THIS IS NOT A FISCAL INVOICE”** below the total amount payable. Font size is at least twice the size of the text on the receipt that specifies the total amount payable.

Training or Proforma or Copy receipt is produced in the same way as normal, with an exception that totals are not accounted for.

For Copy invoice type **Ref no** element is mandatory.

Example:

```
===== THIS IS NOT A FISCAL RECEIPT =====
TIN:                           502579006
Company:                          Golf V
Store:                         Sun Store
Address:                     7 Someplace
District:                           Suva
Cashier TIN:                   123456789
POS number:                89347415-2017
POS time:            2018-03-09 14:57:25
-------------TRAINING SALE--------------
Items
========================================
Name    Price        Qty.          Total
Sport-100 Helmet, Blue (E)
        34.99         10          349.90
Mountain Bike Socks, M (A)
         9.03          4           36.12
----------------------------------------
Total Purchase:                   386.02
Payment Method:                     Cash
========================================
      THIS IS NOT A FISCAL INVOICE
========================================
Label        Name     Rate           Tax
E             STT    6.00%         19.81
A             VAT    9.00%          2.98
----------------------------------------
Total Tax:                         22.79
========================================
SDC Time:            2018-03-08 14:57:46
SDC Invoice No:    7AF4D923-E3B30A31-236
Invoice Counter:                 1/236TS
========================================
---- QR code omitted for simplicity ----
===== THIS IS NOT A FISCAL RECEIPT =====
```

Tax Amounts

It is essential to note, that POS never uses other taxes except the ones received from SDC. POS displays the total prices and only the tax values received from the SDC device, in the format described in the previous section.

A tax label can fall into one of the three tax types: Tax, Tax on Total and Amount per Quantity.

Tax amount for a tax label is calculated per the following formulas:

Tax Type	Formula	Explanation
Tax	**Tax Amount** = ∑(Base price * (**Rate/100**))	for each item with this label
Tax on Total	**Tax Amount** = ∑(Total price * (**Rate/100**))	for each item with this label (Total price here is item base price with all other regular taxes included)
Amount per Quantity	**Tax Amount** = ∑(Fix amount * quantity)	for each item with this label, item quantity is multiplied with fix amount as defined by tax Service. If other tax labels are defined on item, those taxes are calculated on the remainder

Calculate Taxes

Taxes are calculated by an SDC after a POS has sent a valid request. The tax amount for particular items on an invoice are defined by the tax labels associated with an item.

Process of a tax calculation depends on:

- Invoice and Transaction Type
- the tax rates for each label associated with an item on an invoice
- the *Type* value of tax category to which the label belongs

A POS sends an invoice fiscalization request with the line items. Items are sent with the total amounts (taxes included) and zero or more tax labels associated with them, which participated in the total price calculation.

In order to calculate a tax, the following algorithm shall be implemented:

1. Make an array of distinct tax labels associated with the items in the POS request (e.g. A, B, C, F, …).
2. Calculate the tax amount for each individual label in the array:
   a. Iterate through all items in the POS request
   b. For each item, calculate tax amounts. One item has one or more tax labels, and each label represents a tax amount. Each tax amount is a part of the item's total price. These tax amounts are calculated as follows:
      i. If an item has a label from the **amount-on-quantity** category applied, subtract the tax rate amount for that label, multiplied with quantity, from the item total price. The resulting amount (the remainder), is used in all further calculation steps instead of the item total amount.
      ii. If none of the labels' tax category type is ***tax-on-total*** (category 1):
         - Tax amount for one label is: $\frac{item\ total\ amount * label\ rate}{(100+\Sigma(all\ tax-on-net\ rates\ on\ item))}$

         Example 1: An item has a total price of 10$ and applied labels: A(5%) and B(6%).

         $A = \frac{10\$ * 5}{(100+\Sigma(5+6))}$ $B = \frac{10\$ * 6}{(100+\Sigma(5+6))}$

         Tax amount for label A=0.4505$ and for label B=0.5405$.

iii. If any of the labels' tax category is ***tax-on-total*** (category 1):

- Tax amount for every label whose category type is ***tax-on-total*** (category 1) is: $\frac{item\ total\ amount}{(1+\Sigma(all\ tax{-}on{-}total\ rates)/100)} * \frac{label\ rate}{100}$
- Tax amount for every other label from category 0 is: $\frac{item\ total\ amount}{(1+\Sigma(all\ tax{-}on{-}total\ rates)/100)} * \frac{label\ rate}{(100+\Sigma(all\ tax{-}on{-}net\ rates\ on\ item))}$

  Example 2: Item has a total price 10$ and applied labels: A(5% tax-on-net), B(6% tax-on-net), C(3% tax-on-total) and F(4% tax-on-total).

  $C = \frac{10\$}{(1+\Sigma(3+4)/100)} * \frac{3}{100}$ $F = \frac{10\$}{(1+\Sigma(3+4)/100)} * \frac{4}{100}$

  $A = \frac{10\$}{(1+\Sigma(3+4)/100)} * \frac{5}{(100+\Sigma(5+6))}$ $B = \frac{10\$}{(1+\Sigma(3+4)/100)} * \frac{6}{(100+\Sigma(5+6))}$

  Tax amount for label A=0.4210$, for label B=0.5052$ for label C=0.2804$ and for label F=0.3738$.

iv. Add calculated labels' tax amounts to the label's total amount sum.
Example 3: the request contains two items from Example 1 and Example 2, the total sum for labels are: A=0.8715$, B=1.0457$, C=0.2804$, F=0.3738$.

v. Add fixed tax amounts, multiplied with quantity, to the respective labels' total amount sum.
Example 4: An item has quantity 2 with total price 10$ and applied labels: A(5% tax-on-net) and E(0.10$ fixed tax) , total sum for labels are: A=0.4667$, E=0.2000$.
Example 5: An item has quantity 2 with total price 10$ and applied labels: A(5% tax-on-net), C(3% tax-on-total) and E(0.10$ fixed tax) , total sum for labels are: A=0.4531$, C=0.2854$, E=0.2000$.

3. After all the items have been processed, calculate the tax amount for all tax categories found in the request. One tax category can consist of one or more tax labels (e.g. A, B…). The tax amount for a tax category is a sum of all label tax amounts related to the category.
Example 6: the request contains two items from Example 1 and Example 2. Labels A and B are VAT category, C is STT category and F is ET category. Total VAT=1.9172$, STT=0.2804$, ET=0.3738$.

Rounding

SDC shall round all amounts to 4 decimal places internally using the half-round up method.

Examples:

```
3.44445555666  -> 3.4445
3.4440012345   -> 3.4440
3.44466012345  -> 3.4447
3.444116012345 -> 3.4441
```

Choosing an Appropriate Model

The following explanation should help you decide which fiscalization model is the most appropriate for your clients.

	V-SDC	**E-SDC**
Protocol	HTTPS	HTTP
To establish communication with POS	Certificate required	Certificate Not required
API methods	SignInvoiceGetTaxAuthorityParams	VerifyPINSignInvoiceAttentionGetStatusGetSignedInvoiceGetTaxAuthorityParams
Authentication method	PAC/PIN*	PIN code**
SignInvoice method	Same + PAC	Same
HASH property of InvoiceFiscalizationRequest or GetSignedInvoice	NO	YES
Internet connection required to work	YES	NO

* PAC code to be sent in the InvoiceFiscalizationRequest as property, as two-factor authentication method (PAC Code and POS PFX Certificate). It is also an option for POS to authenticate to VSDC using the POS secure element (smart card), in which case the PIN is sent by POS during authentication (handshake).

** ESDC requires PIN code to be sent through VerifyPIN method in order for ESDC to be activated

Differences between VSDC and ESDC

1. VSDC uses the HTTPS protocol.
2. ESDC uses the HTTP protocol.
3. To establish communication between POS and VSDC Authentication certificate is required.
4. To establish communication between POS and ESDC Authentication certificate is not required.
5. API methods used for VSDC are:

    - SignInvoice
    - GetTaxAuthorityParams

6. API methods used for ESDC are:

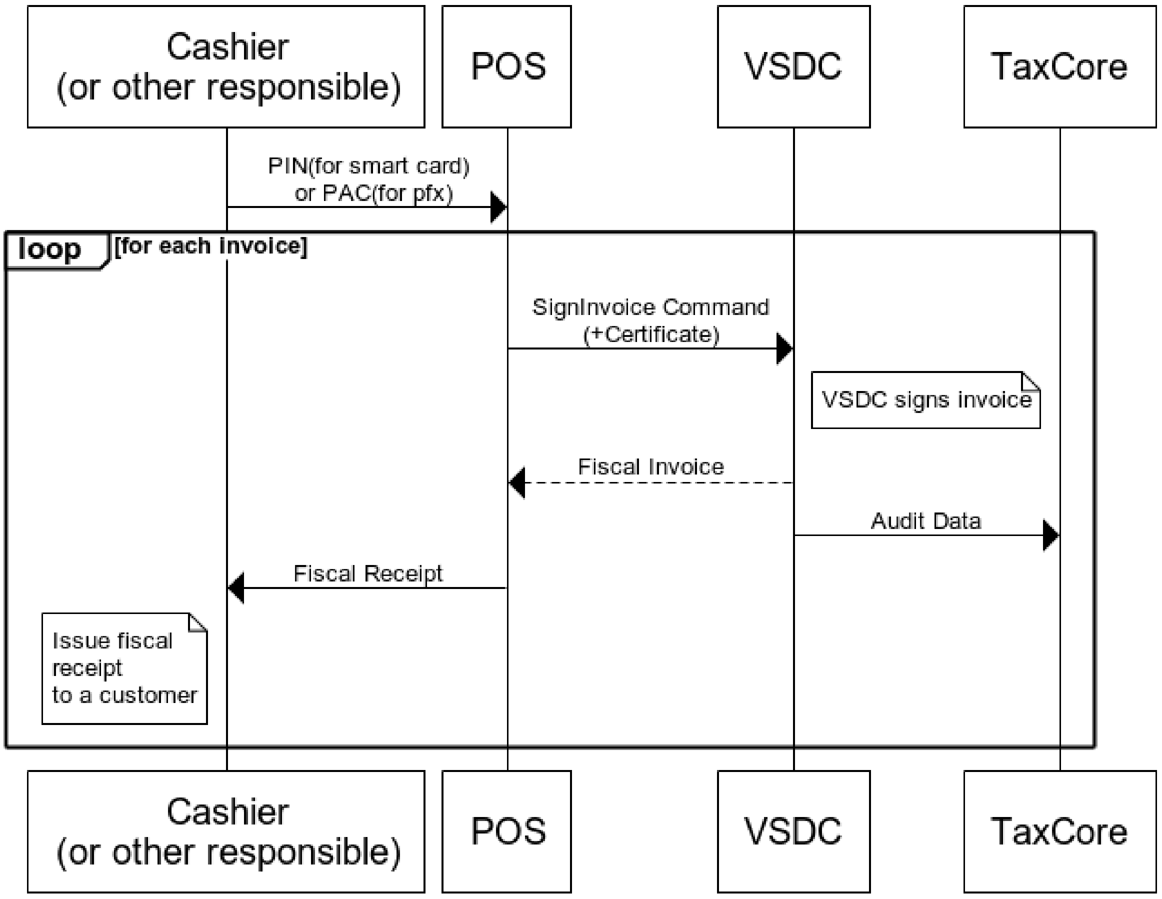
   - VerifyPIN
   - SignInvoice
   - Attention
   - GetStatus
   - GetSignedInvoice
   - GetTaxAuthorityParams

7. VSDC require PAC code to be sent in the InvoiceFiscalizationRequest as property, as two factor authentication method (PAC Code and POS PFX Certificate).
8. ESDC does not require PAC code to be sent in the InvoiceFiscalizationRequest as property. Instead, ESDC requires PIN code to be sent through the VerifyPIN method in order for ESDC to be activated.
9. SignInvoice methods for both ESDC and VSDC are identical except when PAC code is required to be sent in case authentication between POS and VSDC is established by the POS PFX certificate.
10. HASH property of InvoiceFiscalizationRequest or GetSignedInvoice is designed only for ESDC and not for VSDC.
11. VSDC is a cloud-based service, therefore POS requires available internet connection in order to sign invoices.
12. ESDC is an offline based solution, therefore it does not require an available internet connection to sign invoices, ESDC can be connected to the local network (LAN cable or WIFI) or directly to the POS with LAN cable.

Typical process flow

This section describes a typical process flow for successful scenarios.

V-SDC



Cashier (or other responsible)
POS
VSDC
TaxCore
PIN(for smart card) or PAC(for pfx)
loop
[for each invoice]
SignInvoice Command (+Certificate)
VSDC signs invoice
Fiscal Invoice
Audit Data
Fiscal Receipt
Issue fiscal receipt to a customer
Cashier (or other responsible)
POS
VSDC
TaxCore

Figure 2 V-SDC process flow

E-SDC

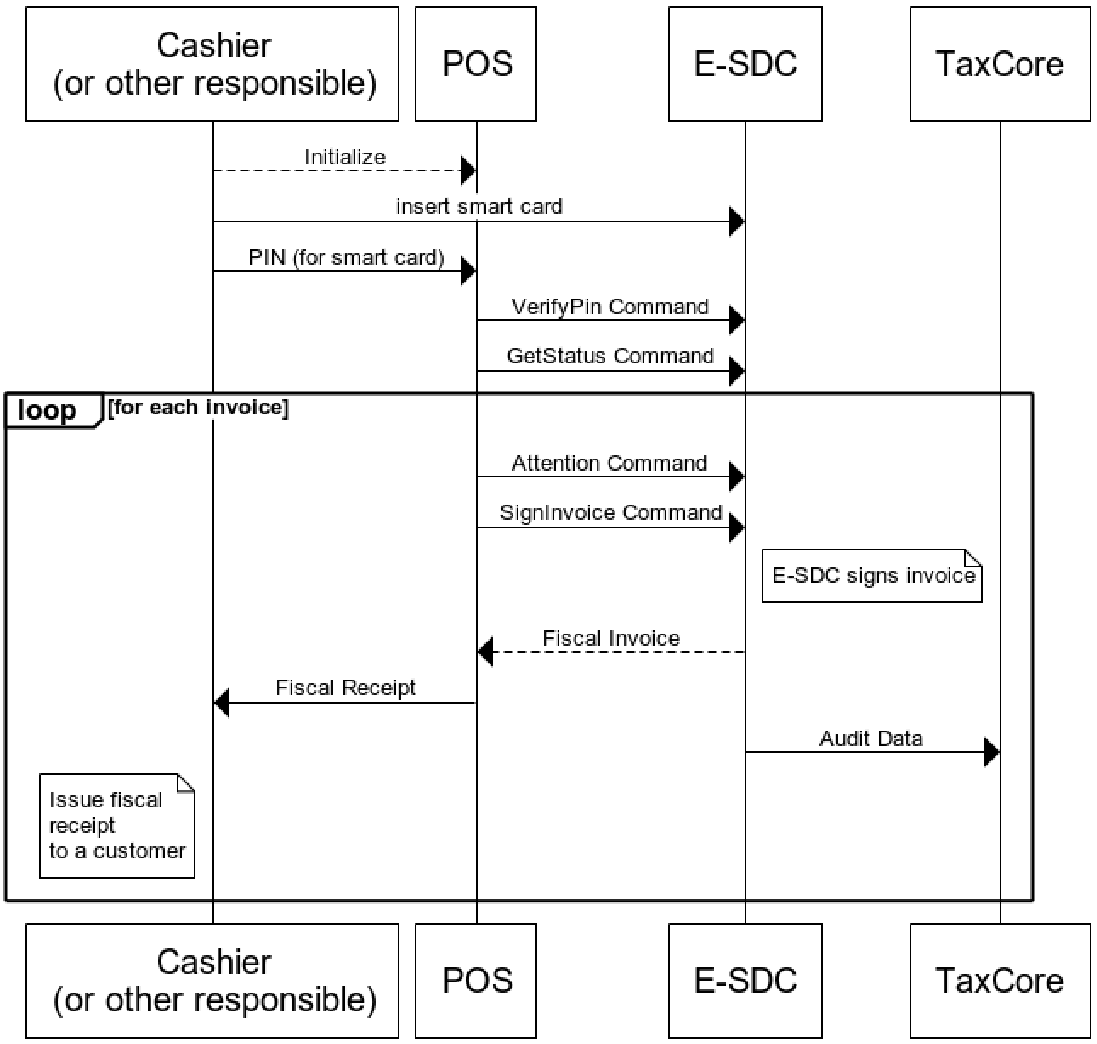


Figure 3 E-SDC process flow

V-SDC Pros and Cons

Pros

1. No specialized hardware is required
2. Accredited POS can be implemented as a mobile app
3. Compliance of the existing ERP system can be done quickly
4. Cost of fiscalization is reduced

Cons

1. Internet connection is required to issue a receipt

E-SDC Pros and Cons

Pros

1. Works without an internet connection
2. Supports older Cash registers with serial connection

Cons

1. Requires a specialized/dedicated hardware

2. Prone to physical damage
3. May require a specialized/dedicated hardware maintenance

Recommendation Examples

This section gives examples of the most common implementation scenarios, for different end users.

Small Shops

In small shops, it is possible to use all kinds of devices from tablets to POS applications. The choice of device generally depends on number of items, which are on the sale list (PLU) or on the environmental conditions.

Agencies, Individuals and Travelling Salesmen

Agencies are not issuing a large number of receipts and issuing is not time-critical; mobile POS application connection to V-SDC will probably cover their needs.

Supermarkets

Supermarkets are using high volume POS systems with additional different peripherals. Due to the very nature of the supermarket or shop sale process (on the counter) it is required to have offline capabilities to overcome interruptions of the internet connection.

Restaurants and Hotels

Restaurants have very specific applications. Proforma as transaction type is supported and recorded and can be verified. Offline capabilities are also important because receipts have to be printed on demand.

Taxi Drivers

Taxi drivers use taximeters that measure parameters from a ride. Old taximeters do this by mechanical methods; there are many challenges in their connection with modern EFD systems. If local regulations allow it, modern taxi terminals or mobile taxi applications are increasingly used worldwide (with GPS locator), which facilitates the connection with the EFD system. Taxi drivers prefer small and robust system. Depending on the chosen taximeter they can use E-SDC or V-SDC.

Remote Sites

POS on the remote or underground sites with an unstable internet connection will have to work with E-SDC devices to provide customers with fiscal invoices. Local audits would be conducted by tax inspectors or taxpayers on a regular basis.

Malls, Shopping Areas

Areas with a high concentration of small shops can contain wireless access point with a dedicated V-SDC for that area.

Enterprises

ERPs and Invoicing systems could utilize both V-SDC and on-site E-SDC devices to fiscalize invoices. It is safe to assume this kind of establishments have permanent (or even redundant) internet connection. Fiscalization using V-SDC service would probably be the most appropriate solution.

Web Shops

Web Shop applications could connect to V-SDC service using the digital certificate issued to the Taxpayer to fiscalize an invoice at the moment of payment.

Connected Scenarios

The simplest scenario is: a Client software application (usually POS) creates an invoice, applies tax labels and calls V-SDC web service to fiscalize the invoice. V-SDC authenticates the caller (verifies taxpayer's digital certificate), performs validation, calculates taxes based on applied tax labels, signs invoice and returns response to the Client.

V-SDC response consists of a digital signature of invoice data, internal encrypted message for Tax Service system, digital certificate metadata, textual representation of invoice, verification URL and optionally a QR code.

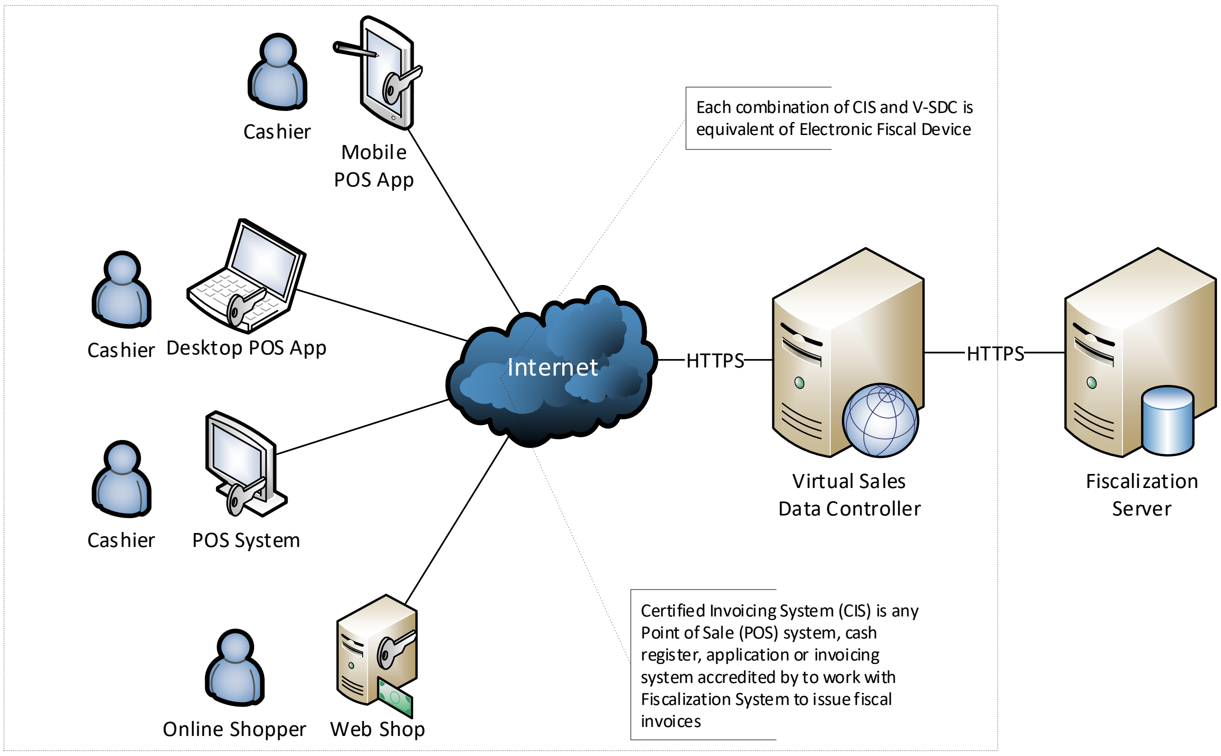

Cashier
Mobile POS App
Each combination of CIS and V-SDC is equivalent of Electronic Fiscal Device
Cashier
Desktop POS App
Internet
HTTPS
HTTPS
Virtual Sales Data Controller
Fiscalization Server
Cashier
POS System
Certified Invoicing System (CIS) is any Point of Sale (POS) system, cash register, application or invoicing system accredited by to work with Fiscalization System to issue fiscal invoices
Online Shopper
Web Shop

Figure 4 Connected Scenarios

Accredited POS prints a textual representation of the invoice and QR code on a receipt. In case the receipt is delivered in electronic form, the verification URL should be rendered as a 'clickable' hyperlink in email or web page.

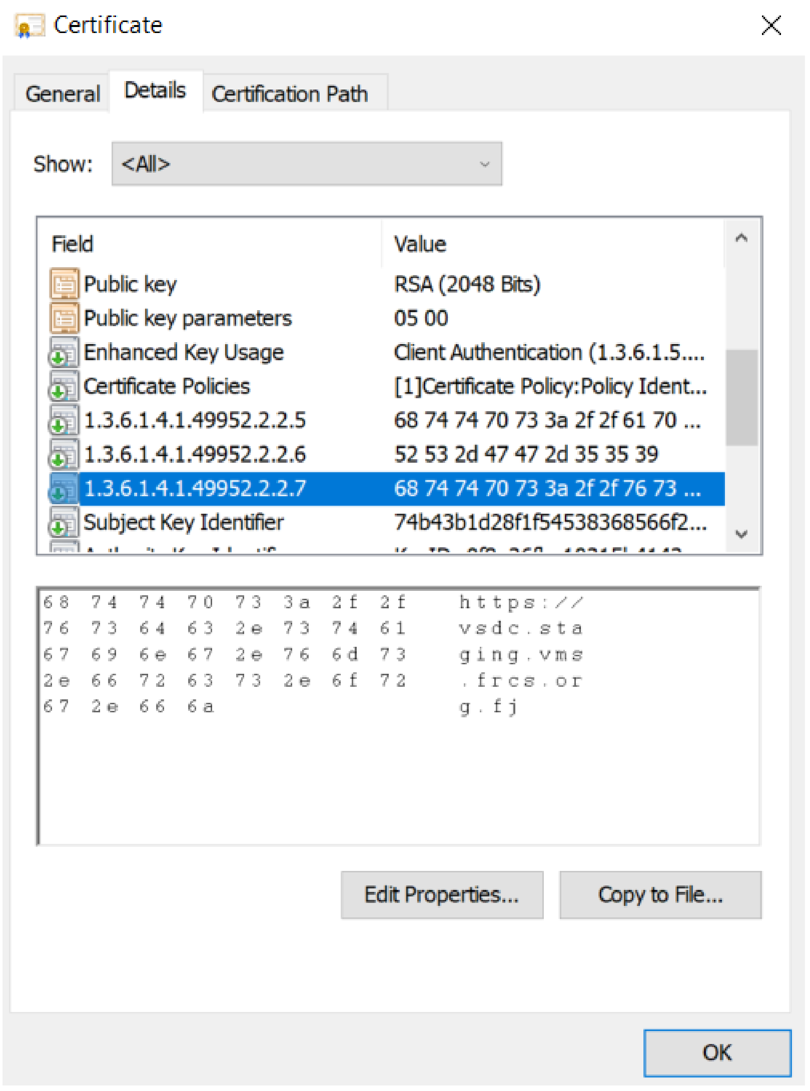
Basically, the receipt fiscalization process consists of the following steps:

1. POS creates an invoice (standard fields like shopping items, see Data Structures)
2. POS submits invoice (JSON format) to V-SDC REST service for fiscalization. POS and V-SDC are mutually authenticated using digital certificates
3. V-SDC authenticates the caller (taxpayer), performs validations and returns the result of the fiscalization (see Protocols section)
4. POS prints a textual representation of the invoice and QR code containing the Verification URL, on the receipt. Paper width should be 58mm / 2.28in or wider.

Accessing V-SDC API

Once valid Test certificate(s) are obtained you can access the V-SDC API description on the following URL: https://vsdc.sandbox.taxcore.online/.

To extract VSDC api URL information from certificate make sure you target:
OID: 1.3.6.1.4.1.49952.**X.Y.7**
X and Y parameters represent environment data, while 7 represents VSDC URL under certificate OID.



Field	Value
Public key	RSA (2048 Bits)
Public key parameters	05 00
Enhanced Key Usage	Client Authentication (1.3.6.1.5....
Certificate Policies	[1]Certificate Policy:Policy Ident...
1.3.6.1.4.1.49952.2.2.5	68 74 74 70 73 3a 2f 2f 61 70 ...
1.3.6.1.4.1.49952.2.2.6	52 53 2d 47 47 2d 35 35 39
1.3.6.1.4.1.49952.2.2.7	68 74 74 70 73 3a 2f 2f 76 73 ...
Subject Key Identifier	74b43b1d28f1f54538368566f2...

```
68 74 74 70 73 3a 2f 2f   https://
76 73 64 63 2e 73 74 61   vsdc.sta
67 69 6e 67 2e 76 6d 73   ging.vms
2e 66 72 63 73 2e 6f 72   .frcs.or
67 2e 66 6a               g.fj
```

Figure 5 Certificate containing VSDC URL

Environments

In case certificate does not contain VSDC URL, use one of the following addresses depending on the target environment:

Samoa:

https://vsdc.staging.tims.revenue.gov.ws/Swagger (for Staging),

https://vsdc.tims.revenue.gov.ws/Swagger (for Production),

Fiji:

https://vsdc.staging.vms.frcs.org.fj/Swagger (for Staging)

https://vsdc.vms.frcs.org.fj/Swagger (for Production).

USA, WA:

https://vsdc.wa.us.taxcore.online/

Demo environment:

https://vsdc.demo.taxcore.dti.rs/

Sandbox environment:

https://vsdc.sandbox.taxcore.online/

This page contains *SignInvoice* service operation details, invoice format and some basic examples.

API is designed and based on OpenAPI-Specification V2 (https://github.com/OAI/OpenAPI-Specification). You can use OpenAPI-Specification code generators (e.g. https://swagger.io/swagger-codegen/) to quickly build a proxy library for almost any programming language and platform.

Client Authentication

Accredited POS Systems are authenticated by V-SDC servers using client digital certificates distributed as PKCS11 file (*.pfx or *.p11) or on the Smart Cards.

You will be able to access the test system using test digital certificates only.

Example

This example illustrates how to create and initialize an instance of **`HttpClient`** class in C# language, use it to authenticate against V-SDC and submit an invoice.

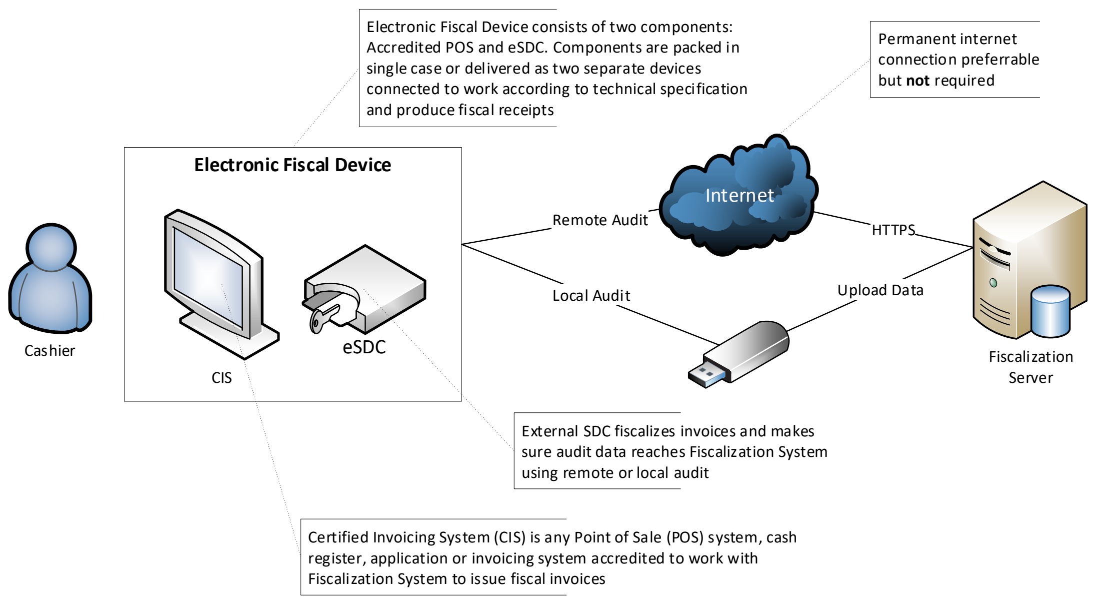
When executing this code, you will be asked to provide the PIN for the smart card certificate, which you selected in GetClientCertificate() method. In case you selected the installed PFX certificate, which you obtained from the Tax Service, you will need to provide PAC value in field PAC.

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

static void Main(string[] args)
{
    string invoiceRequest = @"{
       ""DateAndTimeOfIssue"": ""2017-06-15T08:56:23.286Z"",
       ""Cashier"": ""Oliver"",
       ""BD"": ""8902798054"",
       ""BuyerCostCenterId"": """",
       ""IT"": ""Normal"",
       ""TT"": ""Sale"",
       ""PaymentType"": ""Cash"",
       ""InvoiceNumber"": ""POS2017/998"",
       ""ReferentDocumentNumber"": ""ABCD1234-EFGH5678-198"",
       ""PAC"":"""",
       ""Options"":{
              ""OmitQRCodeGen"": ""1"",
              ""OmitTextualRepresentation"": ""1""},
       ""Items"": [{
              ""Name"": ""Sport-100 Helmet, Blue"",
              ""Quantity"": 2,
              ""UnitPrice"": 34.23,
              ""Labels"": [""A""],
              ""TotalAmount"": 68.46}],
       ""Hash"": ""W331EEgkSRsqTFMO86a8Og==""}";
    var httpContent=new StringContent(invoiceRequest,Encoding.UTF8,"application/json");
    HttpClient client;
    WebRequestHandler handler;
    GetClientAndHandler(out handler, out client);

    var response = client.PostAsync($"api/Sign/SignInvoice", httpContent).Result;
    if (response.StatusCode == HttpStatusCode.OK)
    {
        var jsonString = response.Content.ReadAsStringAsync();
        jsonString.Wait();
        var invoiceResponse = jsonString.Result;
        Console.Write(invoiceResponse);
    }
}

static void GetClientAndHandler(out WebRequestHandler handler, out HttpClient client)
{
    handler = CreateWebRequestHandler();
    client = new HttpClient(handler);
    client.BaseAddress = new Uri("https://vsdc.sandbox.taxcore.online/");
    client.DefaultRequestHeaders.Accept.Clear();
}

static WebRequestHandler CreateWebRequestHandler()
{
    var handler = new WebRequestHandler();
    var cert = GetClientCertificate();

    handler.ClientCertificateOptions = ClientCertificateOption.Manual;
    handler.ClientCertificates.Add(cert);

    return handler;
}

static X509Certificate2 GetClientCertificate()
{
    string certName = "9AH3 My Store inc.";
    var store = new X509Store(StoreName.My, StoreLocation.CurrentUser);
    store.Open(OpenFlags.OpenExistingOnly | OpenFlags.ReadOnly);

    return store.Certificates.Find(X509FindType.FindBySubjectName, certName, true)[0];
}
```

Semi-Connected Scenarios

Taxpayers will be encouraged to use online capabilities whenever possible – V-SDC service will be widely available and accessible from the variety of Accredited POS devices and software solutions. But, in order to rollout a fiscalization system, it has to be able to close any possible gaps in fiscal discipline that may have arisen from a poor or no internet connection.

External Sales Data Controller (E-SDC) device exposes Json-based protocols for communication with the Accredited POS via RS232, USB-to-serial or UTP cable. E-SDC is using a Secure Element to digitally sign invoices received from the Accredited POS and to produce audit data. Audit data is stored on an E-SDC internal memory which enables local and remote audits.



Electronic Fiscal Device consists of two components: Accredited POS and eSDC. Components are packed in single case or delivered as two separate devices connected to work according to technical specification and produce fiscal receipts
Permanent internet connection preferrable but **not** required
Electronic Fiscal Device
Internet
Remote Audit
HTTPS
Local Audit
Upload Data
Cashier
eSDC
CIS
Fiscalization Server
External SDC fiscalizes invoices and makes sure audit data reaches Fiscalization System using remote or local audit
Certified Invoicing System (CIS) is any Point of Sale (POS) system, cash register, application or invoicing system accredited to work with Fiscalization System to issue fiscal invoices

Protocols

Status and Error Codes

Code	0-Info 1-Warning 2-Error	Description	Applies to
	INFO		
0000	All OK	Command is executed without warnings or errors	VSDC, E-SDC
0100	Pin OK	This code indicates that the provided PIN code is correct	E-SDC
0210	Internet Available	Internet Connection is available (optional)	E-SDC
0220	Internet Unavailable	Internet Connection is not available (optional)	E-SDC
	WARNINGS		
1100	Storage 90% Full	Storage used to store audit packages is 90% percent full. It is time to perform the audit.	E-SDC
1300	Smart Card is not present	Secure element card is not inserted in the E-SDC smart card reader	E-SDC
1400	Audit Required	Total Sale and Refund amount reached 75% of the SE limit. It is time to perform the audit.	E-SDC
1500	Pin Code Required	Indicates that POS must provide the PIN code	E-SDC
1999	Undefined Warning	Something is wrong but specific warning is not defined for that situation. Manufacturer can use manufacturer-specific codes to describe warning in more details	E-SDC
	ERRORS		
2100	Pin Not OK	PIN code sent by the POS is invalid	E-SDC
2110	Card Locked	The number of allowed PIN entries exceeded. The card is locked for use	E-SDC
2210	SE Locked	Secure Element is locked. No additional invoices can be signed before the audit is completed	E-SDC
2220	SE Communication Failed	E-SDC cannot connect to the Secure Element applet	E-SDC
2230	SE Protocol Mismatch	Secure Element does not support requested protocol version (reserved for later use)	E-SDC
2310	Invalid tax labels	Tax Labels sent by the POS are not defined	VSDC, E-SDC
2400	Not configured	SDC device is not fully configured for invoice signing (i.e. tax rates or verification URL are missing etc.)	E-SDC
2800	Field Required	The field is required	E-SDC
2801	Field Value Too Long	The length of the field value is longer than expected	VSDC, E-SDC
2802	Field Value Too Short	The length of the field value is shorter than expected	VSDC, E-SDC
2803	Invalid Field Length	The length of the field value is shorter or longer than expected	VSDC, E-SDC
2804	Field Out Of Range	The field value out of expected range	VSDC, E-SDC

2805	Invalid Field Value	The field contains an invalid value	VSDC, E-SDC
2806	Invalid Data Format	The data format is invalid	VSDC, E-SDC
2807	List Too Short	The list contains less than minimum required elements count	VSDC, E-SDC
2808	List Too Long	The list exceeds the maximum allowed elements count.	VSDC, E-SDC
	OBSOLETE		
~~**2811**~~	Invalid Invoice Type	Value of Invoice Type field is invalid or out of range (replaced with **2805**)	VSDC, E-SDC
~~**2812**~~	Invalid Transaction Type	Value of Transaction Type field is invalid or out of range (replaced with **2805**)	VSDC, E-SDC
~~**2813**~~	Invalid Payment Type	Value of Payment Type field is invalid or out of range (replaced with **2805**)	VSDC, E-SDC
~~**2814**~~	BuyerIdLenghtInBytes Length Exceeded	BuyerID field maximum length is exceeded (20 chars) (replaced with **2803**)	VSDC, E-SDC
~~**2815**~~	BuyerCostCenterId Length Exceeded	BuyerCostCenterId field maximum length is exceeded (15 chars) (replaced with **2803**)	VSDC, E-SDC
~~**2816**~~	POSInvoiceNumber Length Exceeded	POSInvoiceNumber field maximum length is exceeded (20 chars) (replaced with **2803**)	VSDC, E-SDC
~~**2817**~~	GTIN Length Invalid	POSInvoiceNumber field length is less than 8 or greater than 14 (replaced with **2803**)	VSDC, E-SDC
~~**2818**~~	Name Length Exceeded	Name field maximum length is exceeded (2048 chars) (replaced with **2803**)	VSDC, E-SDC
~~**2819**~~	Name is Required	An item name is required (replaced with **2800**)	VSDC, E-SDC
~~**2820**~~	Labels Length Exceeded	Label length is exceeded (replaced with **2803**)	VSDC, E-SDC

JSON Based Protocol (Common for POS to V-SDC or E-SDC)

JSON API is designed and based on OpenAPI-Specification V2 (https://github.com/OAI/OpenAPI-Specification). You can use OpenAPI-Specification code generators (e.g. https://swagger.io/swagger-codegen/) to quickly build a proxy library for almost any programming language and platform.

Get Tax Authority Params

This method returns environment-specific information to the caller.

This information is included for informational purposes only. Method is not supported by older versions of E-SDCs.

Endpoint

For V-SDC:	https://vsdc.sandbox.taxcore.online/api/Status/GetTaxAuthorityParams **NOTE:** Development, testing and technical review are all done in SandBox environment. The next step is the administrative review process which enables getting accreditation for a specific country. Different countries have different environment names, and every environment has its own URL. See *Environments*
For E-SDC:	http://<ESDC_ip_address>/api/Status/GetTaxAuthorityParams

HTTP Method

GET

Response

Field	Description
UID	UID of the digital certificate stored in the Subject field
BackendEndpoint	Returns URL of the Backend API stored in the digital certificate.
TaxRateGroup	Returns current `TaxRateGroup` to caller

Sign Invoice Command

HTTP POST request data is sent to:

For V-SDC:	https://vsdc.sandbox.taxcore.online/api/Sign **NOTE:** Development, testing and technical review are all done in SandBox environment. The next step is the administrative review process which enables getting accreditation for a specific country. Different countries have different environment names, and every environment has its own URL. See *Environments*
For E-SDC:	http://< ESDC_ip_address>/api/Sign

Invoice Request

Data Fields

Field	Description
DateAndTimeOfIssue	Current **Local** Date and Time in ISO 8601 format
IT	Invoice Type enumeration value
TT	Transaction Type enumeration value
PaymentType	Payment Type enumeration value
Cashier	Cashier's identification.
BD	Taxpayer ID of the Buyer. **It is mandatory for B2B** transactions; otherwise, it's optional.
BuyerCostCenterId	Cost Center ID provided by the buyer to the cashier in case Buyer's company wants to track spending in Taxpayer Portal. **It is optional and may exist only for B2B transactions; otherwise, it shall be ignored by E-SDC.**
InvoiceNumber	Invoice number generated by POS.
ReferentDocumentNumber	Mandatory only in case Invoice Type is **Refund** or **Copy**. In both cases, this field must contain Invoice Number of previously issued Invoice or Refund. In any other case (for example Normal Sale invoice) this field is optional. ASCII, in ***RequestedBy-SignedBy-OrdinalNumber*** format
PAC	POS Access Code assigned to and used along with digital certificate distributed as PFX file, used to authenticate POS to VSDC. If Smart Card is used to authenticate POS to V-SDC this field is not used and should be left blank.
Items (n)	Each invoice contains at least one Item in Items collection (Every SDC that is used for integration, should support at least 250

	items, up to 500, for all other cases, please contact your SDC vendor)
GTIN	Global Trade Item Number (GTIN) is an identifier for trade items, incorporated the ISBN, ISSN, ISMN, IAN (which includes the European Article Number and Japanese Article Number) and some Universal Product Codes, into a universal number space.
Name	Human-readable name of the product or service.
Quantity	Quantity of an item, e.g. 2 (pieces), 0.100 (grams).
UnitPrice	Unit price of the line item. It does not take part in tax calculation.
TotalAmount	Gross price for the line item.
Labels	The array of labels. Each Label represents one of the Tax Rates applied on invoice item. Tax Items are calculated based on TotalAmount and applied Labels. **This field is required. The caller must submit a non-empty collection.**
Options	Key/value collection defines the output of V-SDC/E-SDC invoice fiscalization, to optimize resources. Key: **`OmitQRCodeGen`** Value: "1" to omit QR Code generation by E-SDC and "0" to generate and return QR code. Key: **`OmitTextualRepresentation`** Value: "1" to omit generation of textual representation by E-SDC and "0" to generate return textual representation to POS.
Hash	Base64 encoded MD5 hash of the request data (used only for E-SDC). It is used only for later invoice search.

Model

```
InvoiceFiscalizationRequest {
       DateAndTimeOfIssue (string, optional),
       Cashier (string, optional) Unicode MaxLength:50,
       BD (string, optional) ASCII MaxLength:20,
       BuyerCostCenterId (string, optional) Unicode MaxLength:15,
       IT (string) = ["Normal", "ProForma", "Copy", "Training"] (int) = [0,1,2,3],
       TT (string) = ["Sale", "Refund"] (int) = [0,1],
       PaymentType (string) = ["Other", "Cash", "Card", "Check", "WireTransfer",
       "Voucher", "MobileMoney" (int) = [0,1,2,3,4,5,6],
       InvoiceNumber (string, optional) Unicode MaxLength:60,
       ReferentDocumentNumber (string, optional),
       PAC (string, optional),
       Options (inline_model, optional),
       Items (Array[Item]) MinLength:1,
       Hash (string, optional) MaxLength:32
}

inline_model {
       OmitQRCodeGen (string, optional) = ["0", "1"] (int) = [0,1],
       OmitTextualRepresentation (string, optional) = ["0", "1"] (int) = [0,1]
}

Item {
       GTIN (string, optional) MinLength:8 MaxLength:14,
       Name (string) Unicode MaxLength:2048,
       Quantity (number) Decimal(14,3) MinValue:0.001,
       UnitPrice (number) Decimal(14,2),
       Labels (Array[string]) MinLength:1,
        TotalAmount (number) Decimal(14,2)
}
```

Example

```
        {
          "DateAndTimeOfIssue": "2017-06-15T08:56:23.286Z",
          "Cashier": "123456789",
          "IT": "Normal",
          "TT": "Sale",
          "PaymentType": "Cash",
          "InvoiceNumber": "POS2017/998",
          "Options": {
            "OmitQRCodeGen" : "1" ,
            "OmitTextualRepresentation" : "0"
          },
          "Items": [
            {
              "GTIN": "987654321",
              "Name": "Sport-100 Helmet, Blue",
              "Quantity": 2,
              "UnitPrice": 34.23,
              "Labels": [
                "A"
              ],
              "TotalAmount": 68.46
            }
          ],
          "Hash": "W33lEEgkSRsqTFMO86a8Og=="
        }
```

Invoice Response

Data Fields

Field	Description
RequestedBy	UID of the digital certificate which requested signing. In case E-SDC is used, its value will be the same as SignedBy.
SignedBy	UID of the digital certificate of the secure element which signed the invoice. In case E-SDC is used, its value will be the same as RequestedBy.
DT	Local Date and time in ISO 8601 format provided by E-SDC/V-SDC.
IC	Invoice Counter in format **TransactionTypeCounter/TotalCounter InvoiceCounterExtension** For Example: 14/17NS
InvoiceCounterExtension	First letters of Transaction Type and Invoice Type of the invoice. NS for Normal Sale, CR – Copy Refund, TS – Training Sale, etc.
IN	Invoice number in format **RequestedBy-SignedBy-TotalCounter**
VerificationUrl	Verification URL generated through fiscalization.
VerificationQRCode	Base64 encoded byte array of GIF images.
Journal	Textual representation of the invoice.
Messages	Custom human-readable message that shall be printed or displayed by POS.
TotalCounter	Total number of documents protected by this instance of Secure Element.
TransactionTypeCounter	Total number of documents of the selected transaction type protected by this instance of Secure Element.

TotalAmount	Sum of all Items totals – grand total billed to the customer.
ID	Internal Data.
S	Digital Signature.
TaxItems	(for each label that exists on the invoice).
Label	Tax Label (i.e. A, F, G)
Name	Tax Category Name (i.e. VAT, Consumption)
Rate	TaxRate i.e. 12.50(%) or 0.10($)
Amount	Tax amount calculated by E-SDC. Refer to section Tax Amounts.
Hash	Hash received from POS in request field Hash (used only for E-SDC)
BusinessName	Taxpayer Business Name obtained from the digital certificate.
LocationName	Location Name obtained from digital certificate.
Address	Street address obtained from digital certificate.
TIN	Tax Identification Number obtained from digital certificate.
District	District name obtained from digital certificate.
MRC	E-SDC manufacturer registration code (in format MakeCode-SW-Serial). -MakeCode: unique 2 characters obtained on Tax Service Accreditation -SW: software version -Serial: manufacturer serial number (max 32 characters)

Model

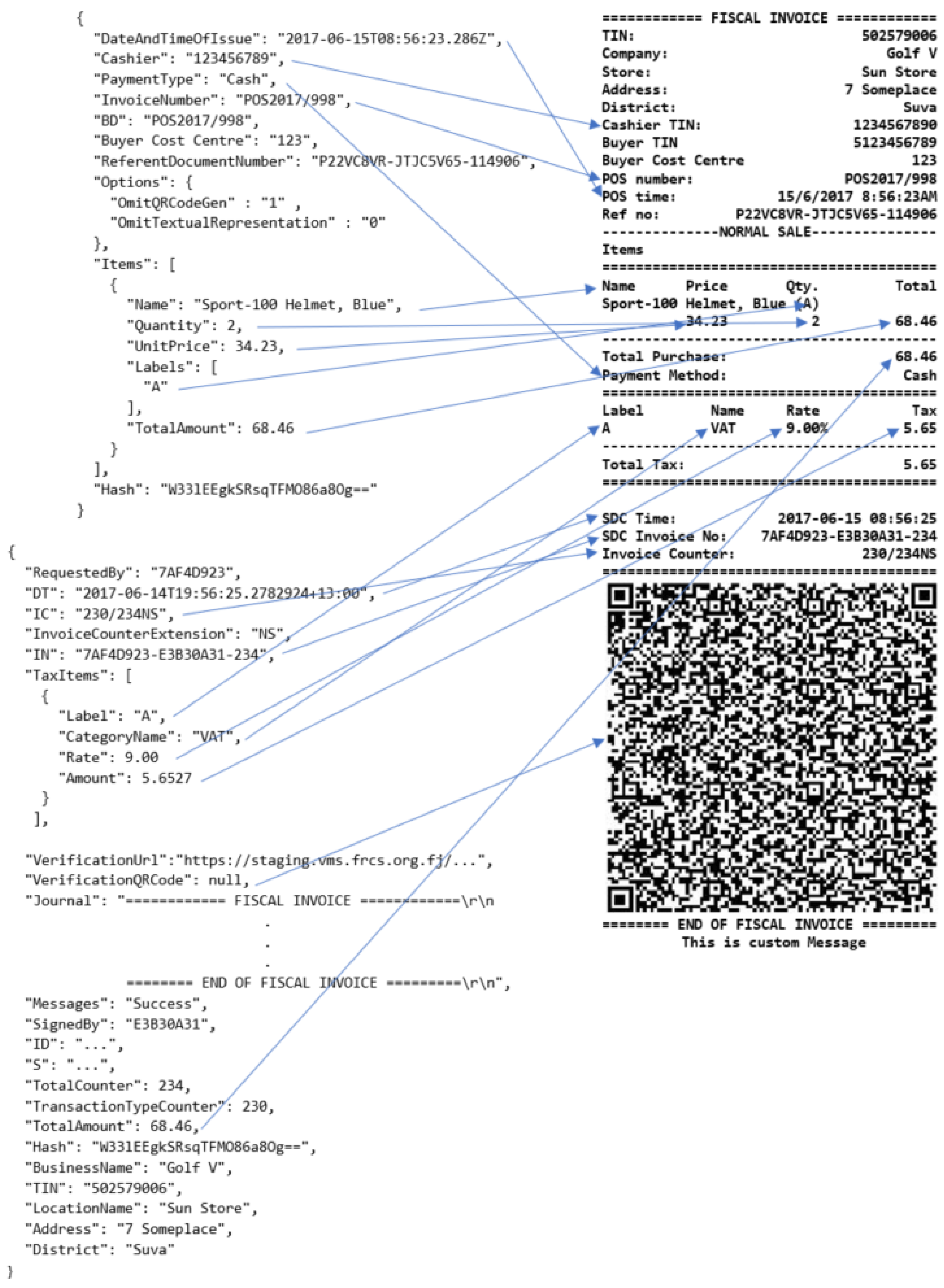
```
InvoiceFiscalizationResult {
       RequestedBy (string),
       DT (string),
       InvoiceCounterExtension (string),
       TaxItems (Array[TaxItem]) MinLength:0,
       VerificationUrl (string),
       VerificationQRCode (string, optional),
       Journal (string, optional),
       Messages (string, optional),
       SignedBy (string),
       ID (string),
       S (string),
       TotalCounter (integer),
       TransactionTypeCounter (integer),
       TotalAmount (number),
       Hash (string, optional),
       BusinessName (string, optional),
       LocationName (string, optional),
       TIN (string, optional),
       Address (string, optional),
       District (string, optional),
       MRC (string, optional)
}

TaxItem {
Label (string),
Amount (number),
Rate (number),
CategoryName (string)
}
```

Example

```
{
  "RequestedBy": "7AF4D923",
 "DT": "2017-06-14T19:56:25.2782924+13:00",
 "IC": "230/234NS",
 "InvoiceCounterExtension": "NS",
 "IN": "7AF4D923-E3B30A31-234",
 "TaxItems": [
   {
     "Label": "A",
     "CategoryName": "VAT",
     "Rate": 9.00
     "Amount": 5.6527
   }
  ],
"VerificationUrl":"https://staging.vms.frcs.org.fj/v/?vl=AVAyMlZDOFZSSlRKQzVWNjUBAAAA
  AQAAAAAuGQ4AAAFeM%2BvCJgAAAAbDDPkm7R9I1NPLierP%2Bh3UQswb%2FXa8xYKiEnLjyClHqh6X26FruP
  VNksB7wMoG2LpA85uvbG9txf2CndYl5JZshBJsq7TLF%2BqOmRs3EaykUVf05mFbTrrgQmUROZE76lciqaxv
  aVEGK83ic1q2HVz0mryqHna6Iu%2FuTn4q2wQ4gJ9bc%2BD6pvyhY%2BZB8c3SgYNGPm4Eq81%2BC8tjJpPC
  YLlHrVKPjbQEE6FSm2II0YEeQqWEGNCHqatxHmjS8sJTT4BJJ%2FlhzTQyuFWoI5ko3oAm8AZsEdgx54oEEN
  r3LUm3Jg%2Fd75tGcUoweEngRfoEP0EiqaOkt2sdSg18hrjd4PUdZ8QUksSeIDmjLMsqZoLmGiqycdajhMN2
  eMeo%2B9LZ%2FhLnxDsROkbOWlArVGfQ%2B9MfBmyJsILCEIT6myTAC2HZCvQ%2Bc0MEO%2F0euynCkCQO6B
  Bv39zNn8yNRagmsEOslkQydty66gphme%2BCOx76u%2F4lCjxPOOxc%2F6zNR8SAe1MNaDPVH3PU7IlQdTox
  fXY3pvWSqtK%2FUY5JGXpvmMpLP6kUXr1qOCjt2Uj6QsH%2BbgwjEZVpHep%2Byh5myEQcI9A4NDUtoUjPpI
  TbOIxPO4vyne%2Bgv4UnpAKQigAv%2FywKeD9noHDgCiFSfLZCJ0IXMSleo%2BjIf%2BIfE2YXX84gH7n7Nc
  pn",
 "VerificationQRCode": null,
"Journal": "============ FISCAL INVOICE ============\r\n
            TIN:                         502579006\r\n
            Company:                        Golf V\r\n
            Store:                       Sun Store\r\n
            Address:                   7 Someplace\r\n
            District:                         Suva\r\n
            Cashier TIN:                 123456789\r\n
            POS number:                POS2017/998\r\n
            POS time:           2017-06-15 8:56:23\r\n
            --------------NORMAL SALE---------------\r\n
            Items\r\n
            ========================================\r\n
            Name    Price        Qty.         Total\r\n
            Sport-100 Helmet, Blue (A)             \r\n
                    34.23          2          68.46\r\n
            ----------------------------------------\r\n
            Total Purchase:                   68.46\r\n
            Payment Method:                    Cash\r\n
            ========================================\r\n
            Label        Name     Rate          Tax\r\n
            A             VAT    9.00%         5.65\r\n
            ----------------------------------------\r\n
            Total Tax:                         5.65\r\n
            ========================================\r\n
            SDC Time:           2017-06-15 8:56:25\r\n
            SDC Invoice No:    7AF4D923-E3B30A31-234\r\n
            Invoice Counter:             230/234NS\r\n
            ========================================\r\n
            ======== END OF FISCAL INVOICE =========\r\n",
 "Messages": "Success",
 "SignedBy": "E3B30A31",
 "ID":
  "BsMM+SbtH0jU08uJ6s/6HdRCzBv9drzFgqIScuPIKUeqHpfboWu49U2SwHvAygbYukDzm69sb23F/YKd1iX
  klmyEEmyrtMsX6o6ZGzcRrKRRV/TmYVtOuuBCZRE5kTvqVyKprG9pUQYrzeJzWrYdXPSavKoedroi7+5Ofir
  bBDiAn1tz4Pqm/KFj5kHxzdKBg0Y+bgSrzX4Ly2Mmk8JguUetUo+NtAQToVKbYgjRgR5CpYQY0Iepq3EeaNL
  ywlNPgEkn+WHNNDK4VagjmSjegCbwBmwR2DHnigQQ2vctSbcmD93vm0ZxSjB4SeBF+gQ/QSKpo6S3ax1KDXy
  GuN3g9Q==",
 "S":
   "HWfEFJLEniA5oyzLKmaC5hoqsnHWo4TDdnjHqPvS2f4S58Q7ETpGzlpQK1Rn0PvTHwZsibCCwhCE+pskwAt
   h2Qr0PnNDBDv9HrspwpAkDugQb9/czZ/MjUWoJrBDrJZEMnbcuuoKYZnvgjse+rv+JQo8TzjsXP+szUfEgHt
   TDWgz1R9z1OyJUHU6MX12N6b1kqrSv1GOSRl6b5jKSz+pFF69ajgo7dlI+kLB/m4MIxGVaR3qfsoeZshEHCP
   QODQ1LaFIz6SE2ziMTzuL8p3voL+FJ6QCkIoAL/8sCng/Z6Bw4AohUny2QidCFzEpXqPoyH/iHxNmF1/OIB+
   5+zXKZw==",
 "TotalCounter": 234,
 "TransactionTypeCounter": 230,
 "TotalAmount": 68.46,
 "Hash": "W33lEEgkSRsqTFMO86a8Og==",
 "BusinessName": "Golf V",
 "TIN": "502579006",
 "LocationName": "Sun Store",
 "Address": "7 Someplace",
 "District": "Suva"
}
```

Mapping Fiscal Invoice to Fiscal Receipt

In case POS does not use Journal (generated by E-SDC or V-SDC) as a content for a fiscal receipt, but it generates a custom-designed receipt instead, it shall use the following element mappings:



```
{
  "DateAndTimeOfIssue": "2017-06-15T08:56:23.286Z",
  "Cashier": "123456789",
  "PaymentType": "Cash",
  "InvoiceNumber": "POS2017/998",
  "BD": "POS2017/998",
  "Buyer Cost Centre": "123",
  "ReferentDocumentNumber": "P22VC8VR-JTJC5V65-114906",
  "Options": {
    "OmitQRCodeGen" : "1" ,
    "OmitTextualRepresentation" : "0"
  },
  "Items": [
    {
      "Name": "Sport-100 Helmet, Blue",
      "Quantity": 2,
      "UnitPrice": 34.23,
      "Labels": [
        "A"
      ],
      "TotalAmount": 68.46
    }
  ],
  "Hash": "W331EEgkSRsqTFMO86a8Og=="
}
```

```
{
  "RequestedBy": "7AF4D923",
  "DT": "2017-06-14T19:56:25.2782924+13:00",
  "IC": "230/234NS",
  "InvoiceCounterExtension": "NS",
  "IN": "7AF4D923-E3B30A31-234",
  "TaxItems": [
    {
      "Label": "A",
      "CategoryName": "VAT",
      "Rate": 9.00
      "Amount": 5.6527
    }
  ],

  "VerificationUrl":"https://staging.vms.frcs.org.fj/...",
  "VerificationQRCode": null,
  "Journal": "============ FISCAL INVOICE ============\r\n
                         .
                         .
                         .
              ======== END OF FISCAL INVOICE =========\r\n",
  "Messages": "Success",
  "SignedBy": "E3B30A31",
  "ID": "...",
  "S": "...",
  "TotalCounter": 234,
  "TransactionTypeCounter": 230,
  "TotalAmount": 68.46,
  "Hash": "W331EEgkSRsqTFMO86a8Og==",
  "BusinessName": "Golf V",
  "TIN": "502579006",
  "LocationName": "Sun Store",
  "Address": "7 Someplace",
  "District": "Suva"
}
```

```
============ FISCAL INVOICE ============
TIN:                           502579006
Company:                          Golf V
Store:                         Sun Store
Address:                     7 Someplace
District:                           Suva
Cashier TIN:                  1234567890
Buyer TIN                     5123456789
Buyer Cost Centre                    123
POS number:                  POS2017/998
POS time:             15/6/2017 8:56:23AM
Ref no:         P22VC8VR-JTJC5V65-114906
--------------NORMAL SALE---------------
Items
========================================
Name          Price      Qty.      Total
Sport-100 Helmet, Blue (A)
              34.23        2       68.46
----------------------------------------
Total Purchase:                    68.46
Payment Method:                     Cash
========================================
Label         Name     Rate          Tax
A             VAT      9.00%        5.65
----------------------------------------
Total Tax:                          5.65
========================================

SDC Time:            2017-06-15 08:56:25
SDC Invoice No:    7AF4D923-E3B30A31-234
Invoice Counter:               230/234NS
========================================

======== END OF FISCAL INVOICE =========
        This is custom Message
```

Figure 5 Mapping Fiscal Invoice to Fiscal Receipt

JSON Based Protocol (POS to E-SDC Specific)

There are 5 types of Commands (request/response) that can be used for communication between a POS and an E-SDC:

1. Get Status
2. Verify PIN
3. Sign Invoice
4. Attention
5. Get Last Signed Invoice

Get Status Command

This command is used to get status information from E-SDC.

Request Data

JSON data field with a predefined string value.

Example

```
{
  "GS": "GetStatus"
}
```

Response Data

JSON formatted data in accordance with Table 1 Get Status response data.

Filed	Description	Example
`IsPinRequired`	If PIN is not entered, or if a wrong PIN is entered in the previous attempt, this field shall be set to true; otherwise set to false	`true`

`AuditRequired`	If Audit is required, this field shall be set to true. Audit is required if the Total Amount of all invoices is 75% or more of the Maximum Limit. Maximum Limit and Total Amount are obtained from the Secure Element	`False`
		if
		Total Amount is 1554879
		Maximum Limit is 9000000
		
		`True`
		If
		Total Amount is 7504899
		Maximum Limit is 10000000

`DT`	Current **Local** Date and Time in ISO 8601 format	`2017-08-30T11:53:05+13:00`
`LastInvoiceNumber`	Invoice number of the last invoice signed by this E-SDC.	`ORG674J1-ORG674J1-98637`
`ProtocolVersion`	Always 1.0.0.0	`1.0.0.0`
`SecureElementVersion`	Version of the Secure Element	`1.0.0.0`
`HardwareVersion`	Manufacturer-specific hardware version, if applicable	`1.2.7.21`
`SoftwareVersion`	Manufacturer-specific software version	`1.7.6.5`
`DeviceSerialNumber`	Manufacturer specific serial number	`1289A24EB67F22C1`
`Make`	Manufacturer-specific Make Name	`Acme`
`Model`	Manufacturer-specific Model Name	`The Device 442`
`MSSC`	Manufacturer-Specific Errors, Warnings and info messages	Array of error codes

GSC	General Errors, Warnings and info messages defined in section Status and Error Codes.	Array of error codes

Table 1 Get Status response data

Example

```
{
  "IsPinRequired": true,
  "AuditRequired": false,
  "DT": "2017-08-30T11:53:05+13:00",
  "LastInvoiceNumber": "ORG674J1-ORG674J1-98637",
  "ProtocolVersion" : "1.0.0.0",
  "SecureElementVersion" : "1.0.0.0",
  "HardwareVersion" : "1.2.7.21",
  "SoftwareVersion" : "1.7.6.5",
  "DeviceSerialNumber" : "1289A24EB67F22C1",
  "Make" : "Acme",
  "Model" : "The Device 442",
  "MSSC" : ["0440", "5541", "5442"],
  "GSC" : ["1100", "1101", "1102", "1103"]
}
```

Verify PIN Command

This command is used to verify a PIN entered by a cashier on a POS. PIN is verified by the E-SDC. Command should be provided each time the Secure Element smart card is inserted into the E-SDC reader.

Request Data

JSON string with PIN code sent from POS.

Example

```
{
   "VPIN": "1234"
}
```

Response Data

JSON string returned from E-SDC, content can be one of General Status Codes: 0100, 1300, 2100, 2210, 2220, 2230 or 2400. For more information consult Status and Error Codes.

Example

```
{
  "VPIN_GSC": "0100"
}
```

Attention Command

This command is used by POS to verify if ESDC is available. The command should be used prior to Sign Invoice or Send Pin requests. This significantly lowers the possibility of communication errors, including timeout errors. Only if a valid response is received, the POS shall immediately send the next command.

Request Data

JSON data field with a predefined string value.

Example

```
{
  "ATT": "Attention"
}
```

Response Data

JSON string returned from E-SDC, content can be General Status Code: 0000. For more information consult Status and Error Codes.

Example

```
{
  "ATT_GSC": "0000"
}
```

Get Last Signed Invoice Command

Get the last signed invoice. It is used in case POS did not get a response from ESDC after Sign Invoice Command was sent. The response is the same as for Sign Invoice Command. Hash field from response should be used to determine whether the last invoice was successfully signed.

Request Data

JSON data field with Hash string value of the last Sign Invoice Command request sent by POS.

Example

```
{
  "GI": "MDNDN0MwQUNFMzk1RDgwMQ=="
}
```

Response Data

Refer to response for Sign Invoice Command.

Error Messages Format

This section describes the structure and format of error messages returned by SDC to POS.

Data Fields

Message	Human-readable error information. If unsure which error message to return to a POS, use "The request is invalid" phrase
ModelState	Dynamic object containing key-value pairs of FieldName-ErrorCode .
Object	Name of the object (if applicable)
FieldName	Path to the field error codes are associated with. If some of the fields in the path is an array, order number shall be included in square brackets.
ErrorCode	Error code associated with a particular field, from the section Status and Error Codes

Model

```
ModelStateDictionary {
        Message (string),
        ModelState (Array(ModelError))
        }


ModelError {
        FieldName (Array[ErrorCode (string)])
}
```

Implementation example in C# programming language:

```
public class ModelStateDictionary
{
    public string Message { get; set; }
    public Dictionary<string, string[]> ModelState { get; set; }
}
```

Example

```
{
  "Message": "The request is invalid.",
  "ModelState": {
    "invoice.IT": [
      "1234"
    ],
    "invoice.Items[0].Labels[0]": [
      "1234"
    ],
    "invoice.Items[0].GTIN": [
      "1234",
      "5678"
    ]
  }
}
```

POS to E-SDC Communication over HTTP Protocol

E-SDC device should be equipped with an Ethernet port or a Wireless controller in accordance with IEEE 802.3, with speed no less than 100Mb/s, in order to access a local area network.

The physical connection to a network can be established with a standard LAN cable, Cat.5 or similar with better features. The ends of the cables should be equipped with RJ-45 plug male connectors, since an E-SDC is equipped with a female RJ-45 connector.

E-SDC should have a globally unique MAC-48 address in accordance with IEEE 802, which is stored on a specialized MAC Address chip, or an address obtained by the authorized vendor stored in the permanent memory during the manufacturing.

IP Address and other network settings on an E-SDC should be configurable. The technical implementation of these features is in the scope of E-SDC manufacturer.

When an HTTP connection is used between a POS and an E-SDC, data is exchanged in JSON text-based format. POS device must be able to send JSON formatted data to the specified E-SDC IP address using HTTP protocol and to receive response data from the E-SDC using the same protocol.

Get Status Command

This command is used to get status information from E-SDC.

HTTP POST request is sent to: http://<E-SDC_ip_address>:<port>/api/Status/GetStatus

Example: `http://192.168.88.112:8888/api/Status/GetStatus`

Verify PIN Command

HTTP POST request data is sent to: http://<ESDC_ip_address>:<port>/api/Status/VerifyPin

Example: `http://192.168.88.112:8888/api/Status/VerifyPin`

Attention Command

HTTP POST request data is sent to: http://<ESDC_ip_address>:<port>/api/Status/Attention

Example: `http://192.168.88.112:8888/api/Status/Attention`

Sign Invoice Command

HTTP POST request data is sent to: http://<ESDC_ip_address>:<port>/api/Sign/SignInvoice

Example: `http://192.168.88.112:8888/api/Sign/SignInvoice`

Get Last Signed Invoice Command

HTTP POST request data is sent to: http://<ESDC_ip_address>:<port>/api/Sign/GetSignedInvoice

Example: `http://192.168.88.112:8888/api/Sign/GetSignedInvoice`

POS to E-SDC Communication over Serial Port

Some E-SDC developers could choose to support older Accredited POS devices by exposing a serial port data transfer protocol.

In that case, the Accredited POS must be connected to the E-SDC by using NULL MODEM (crossover) serial cable with Transmit (Tx), Receive (Rx) and common ground (GND) cores. Cables with integrated “Serial to USB” converters can be used, too. Physical parameters of the serial protocol are defined by the following parameters:

Databits	8
Parity	Non
Stopbits	1
Baudrate	115200 b/s
Handshake	Non

Above mentioned parameters are defined during the manufacturing process, they are hardcoded in hardware, so they can't be changed later. Automatic baud rate detection is not possible.

The order of transmission of bits is LSB (least significant bit) first.

Initialization of the serial communication is always done by a POS, it is never started by an E-SDC. In normal working mode, when the communication is uninterrupted, every request from the POS to the E-SDC is followed by an appropriate response in the opposite direction.

Serial transmission protocol doesn't implement any error detection protocol by default, so SLIP protocol with Fletcher-16 checksum is used.

Serial port protocol defines the following commands that will be executed by POS: **VerifyPIN**, **SignInvoice, Attention**, **GetStatus** and **GetSignedInvoice**. All commands must be UTF-8 encoded string.

SLIP Protocol

The Serial Line Internet Protocol (SLIP) is an encapsulation of the Internet Protocol designed to work over serial ports and modem connections. It is documented in RFC 1055. On microcontrollers, SLIP is the preferred way of encapsulating IP packets due to its very small overhead.

SLIP defines the following special bytes to be used:

Hex value	**Dec Value**	**Oct Value**	**Abbreviation**	**Description**
0xC0	192	300	END	Frame End
0xDB	219	333	ESC	Frame Escape
0xDC	220	334	ESC_END	Transposed Frame End
0xDD	221	335	ESC_ESC	Transposed Frame Escape

SLIP modifies a standard TCP/IP datagram by

- appending a special "END" byte to it, which distinguishes datagram boundaries in the byte stream,
- if the END byte occurs in the data to be sent, the two byte sequence ESC, ESC_END is sent instead (0xDB, 0xDC),
- if the ESC byte occurs in the data, the two byte sequence ESC, ESC_ESC is sent (0xDB, 0xDD).
- variants of the protocol may begin, as well as end, packets with END.

Therefore, an ESC byte in a SLIP packet shall always be followed by an ESC_END or an ESC_ESC byte; anything else shall be considered a protocol error. Although the implementation code proposed by RFC 1055 ignores such errors, ESDC and POS shall detect and report following SLIP errors:

- ESC character at the end of the packet.
- ESC character in the middle or at the beginning of the packet but not followed by ESC_END or ESC_ESC characters.

Request

Every request sent from a POS over a serial communication protocol is a SLIP packet consisted of the following segments:

<Command><Payload><Checksum><SLIP End>

- Command identifier, 1 byte (alphanumeric) symbol that uniquely identifies command type.
- Payload is a UTF-8 encoded JSON based command. Commands are defined in section JSON Based POS to E-SDC Protocol.
- Checksum is Fletcher-16 checksum calculated on <Command> and <Payload> segments, as defined in section Fletcher-16 checksum.

Example

The following is the example request bytes (in hexadecimal format) for the Verify PIN Command.

507B225650494E223A2231323334227DB6C4C0

Response

Every response sent by E-SDC to POS over a serial communication protocol is a SLIP packet consisted of the following segments:

<Payload><Checksum><SLIP End>

- Payload is a UTF-8 encoded JSON based command response as defined in section JSON Based POS to E-SDC Protocol.
- Checksum is Fletcher-16 checksum calculated on <Payload> segment, as defined in section Fletcher-16 checksum.

Example

The following is the example response bytes (in hexadecimal format) for the successful Verify PIN Command response.

7B225650494E5F475343223A2230313030227D67F8C0

Fletcher-16 checksum

The following code represents optimized C language 8-bit implementation of the checksum calculation (https://en.wikipedia.org/wiki/Fletcher%27s_checksum):

```
uint16_t fletcher16(uint8_t* dataIn, uint16_t bytes)
{
    uint16_t sum1 = 0xff, sum2 = 0xff;
    uint16_t tlen;
    while (bytes){
        tlen = bytes >= 20 ? 20 : bytes;
        bytes -= tlen;
        do {
            sum2 += sum1 += *dataIn++;
        } while (--tlen);
        sum1 = (sum1 & 0xff) + (sum1 >> 8);
        sum2 = (sum2 & 0xff) + (sum2 >> 8);
    }
    /* Second reduction step to reduce sums to 8 bits */
    sum1 = (sum1 & 0xff) + (sum1 >> 8);
    sum2 = (sum2 & 0xff) + (sum2 >> 8);
    return sum2 << 8 | sum1;
}
```

Timeouts

All requests should use timeout period of 10 seconds. If there was no response within timeout period this is considered as a timeout error.

In case of the timeout error on Attention request, POS should simply repeat this request.

Timeout error on Sign Invoice request should be handled by sending Get Signature request until a proper response is received:

- If Hash field from Get Signature response is different from the currently processed invoice, POS should repeat Sign Invoice request because it means that ESDC did not sign that last invoice.
- If Hash field from Get Signature response is the same as for the currently processed invoice, POS can finish processing the current invoice.

It is important for POS to ensure that hash of the currently processing invoice is stored in non-volatile memory until the invoice is successfully signed. This ensures that invoice can be signed even in case of POS power failure.

In case of timeout error on the Get Signature request, POS should simply repeat this request.

Please note that it is good practice to send the Attention request and wait for a valid response before sending any other Command.

GetStatus Command

GetStatus Command is used by a POS to gather information on the state of the connected E-SDC device.

Command Identifier: S (“0x53” in hexadecimal).

Verify PIN Command

This command is used to provide a PIN code to the Secure Element.

Command Identifier: P (“0x50” in hexadecimal).

Sign Invoice Command

Sign Invoice Command performs the tax calculation, creates a verification URL, applies a digital signature and optionally generates a QR code and a textual representation of the invoice.

Command Identifier: I (“0x49” in hexadecimal).

Attention Command

This command is used by POS to verify if E-SDC is available.

Command Identifier: A (“0x41” in hexadecimal).

Get Last Signed Invoice Command

This command is used by POS to get the last signed invoice.

Command Identifier: G (“0x47” in hexadecimal).

POS to E-SDC Communication over TCP

Shall be supported in the future revision of the document.

Online POS and V-SDC integration

Since Taxpayers are encouraged to use online POS capabilities, TaxCore supports scenarios for browser-based client applications. Accredited online POS shall create **Invoice Requests**, and submit them via HTTPS protocol directly to V-SDC API, using the **digital certificate** issued to Taxpayer. This process shall complete invoice fiscalization with a signed invoice returned to online POS.

Online POS **shall** submit requests to V-SDC API **directly,** in order to create an HTTP request with the client certificate. For client-side, JavaScript-based applications, the most common solution for achieving the best user experience is using AJAX. For security reasons, browsers restrict cross-origin HTTP requests initiated from within the scripts. TaxCore offers a solution for online POS, which shall overcome issues with CORS and digital certificates sent from the client.

We recommend the usage of a secure network communication with SSL protocol for online POS, although the V-SDC API will accept requests from an unsecured online POS.

Quick start

Our pre-built TaxCore element is used to collect data from the online POS page. Online POS prepares Invoice Request data for this page thus TaxCore element can collect and send this data to V-SDC. Quick integration can be achieved as follows:

1. Online POS needs to provide a page for collecting and sending **Invoices**. This is the **integration point** on the POS side.
2. Add the following script tag to integration point, with src attribute reffering to taxcore.min.js file:
   `<script src="[vsdcurl]/onlinepos/v1/taxcore.min.js"></script>`

   Instead of **`[vsdcurl]`** provide correct V-SDC URL for appropriate environment.[1]

3. Add TaxCore Sign Element to your integration point with required id and data-* attributes.

```
<!--TaxCore html element-->
<button id="taxcore_sign_element"
        data-taxcore-vsdc-url="[vsdcurl]"
        data-taxcore-input-id="[inputDataContainerId]"
        data-taxcore-output-id="[outputDataContainerId]">Sign Invoice</button>
```

- **`Id`** attribute is required and it must be equal to string **"taxcore_sign_element"**.
- **`vsdcurl`** – provide V-SDC API url
- **`inputDataContainerId`** – provide id of HTML tag element which contains invoice request json, usually input tag
- **`outputDataContainerId`** – provide id of HTML tag element which will be populated by response from V-SDC API

4. Click on the TaxCore Sign Element in order to receive a signed invoice as the value of HTML tag element with id equal to provided [**`outputDataContainerId`**]

[1] For example, js file location at staging environment is
`<script src=" https://vsdc.sandbox.taxcore.online/onlinepos/v1/taxcore.min.js"></script>`

Detailed specs

There are two important components that enable the integration of online POS with V-SDC.

1. taxcore.min.js file
2. taxcore sign element

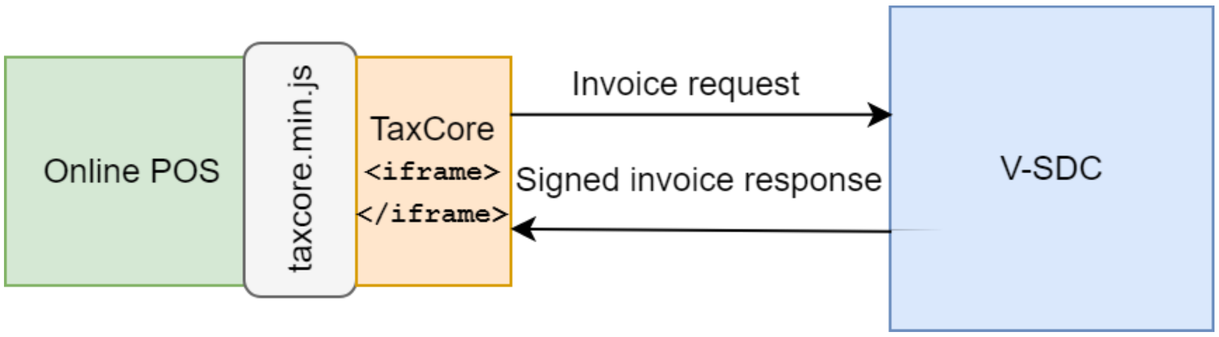
Note: Both components must exist at integration point for successful integration. JavaScript will first try to find taxcore sign element with `id="taxcore_sign_element"`. If it can't be found, the exception will be thrown with an appropriate message (e.g. "*Could not find taxCoreElement. Check if you have valid html tag with expected id taxcore_sign_element*").

TaxCore JavaScript file - taxcore.min.js

The first step in enabling your online POS application page to work with V-SDC is to include script reference to **taxcore.min.js** file, provided by TaxCore. This JavaScript will prepare your page to handle HTTP invoice requests to V-SDC API.

Online POS must directly submit data to V-SDC API to transport the digital certificate over the network from the client side (in this case from Taxpayer's web browser). In order to achieve this requirement, taxcore.min.js script will create **<iframe> element** within online POS, effectively embedding the V-SDC HTML page into the current POS page. POS (Parent page) sends messages to iframe (Child page), and then iframe performs post to V-SDC. By doing so, we can overcome CORS issues with digital certificates, since client certificate in CORS preflight OPTIONS requests is omitted[2].

The other features, provided by this js file are related to sending and receiving data to and from V-SDC. Your online POS needs to prepare data for input and to handle returned data. All other job is delegated to taxcore.min.js.



taxcore.min.js as the communication interface between POS and V-SDC

After successfully creating the iframe element, we can send messages between the Online POS parent page and the TaxCore child page. The responsibility of initiating the message is part of the second component *TaxCore Sign Element* described in the next section.

[2] https://www.w3.org/TR/cors/#cross-origin-request-with-preflight-0

TaxCore Sign Element

TaxCore Sign element is nothing more than an HTML tag with predefined id, that you'll place at your online POS page. Its task is to collect invoice data and forward them to V-SDC.

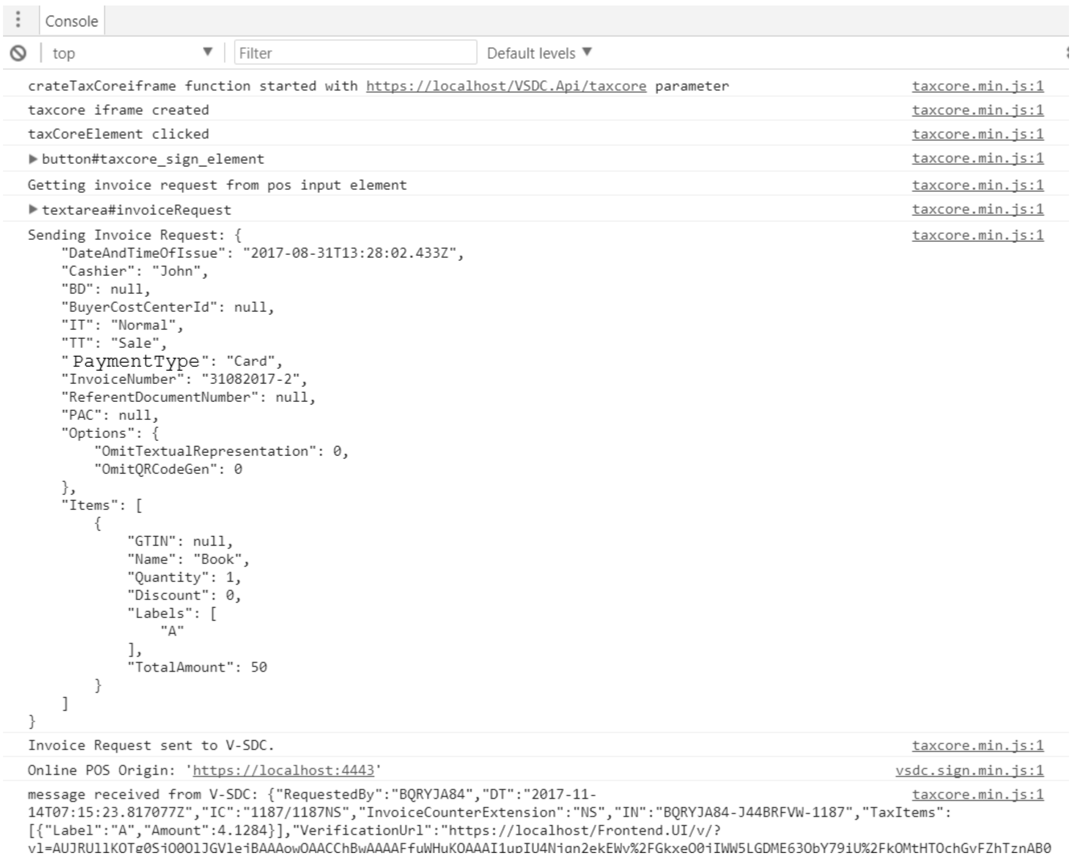
Note: The id of TaxCore Sign Element **must** be equal to string **`"taxcore_sign_element"`**, so that code in taxcore.min.js could be able to find it. Otherwise, the data collection would not be possible.

The best practice for this element is to be clickable HTML element, e.g. a button, but it's not restricted to it. TaxCore Sign Element is also used to configure its behavior using HTML data-* attributes, as follows:

```
<!--TaxCore html element-->
<button id="taxcore_sign_element"
    data-taxcore-vsdc-url=" https://vsdc.sandbox.taxcore.online/"
    data-taxcore-input-id="invoiceRequest"
    data-taxcore-output-id="invoiceResponse"
    data-taxcore-invoice-request=""
    data-taxcore-debug="true"
    data-taxcore-signed-invoice-response="">
Sign Invoice
</button>
```

Data attribute	Restriction	Description
data-taxcore-vsdc-url	Required	V-SDC URL
data-taxcore-input-id	required if **data-taxcore-invoice-request** attribute is not used	Id of the HTML element on POS page, which contains prepared invoice request as required JSON scheme.
data-taxcore-output-id	It is optional since V-SDC will always store signed invoice response at **data-taxcore-signed-invoice-respons**e attribute	Id of the HTML element on POS page used to store signed invoice response JSON from V-SDC.
data-taxcore-invoice-request	required if **data-taxcore-input-id** attribute is not used	If you do not want to use separate HTML element for invoice request JSON, you can store it at this data attribute, and it will be collected when TaxCore Sign Element is clicked.
data-taxcore-signed-invoice-response		This is the default storage location for the signed invoice response JSON from V-SDC. It will be always populated.

data-taxcore-debug	Optional	Used to log relevant information during invoice fiscalization. The log is written to the browser console.



```
crateTaxCoreiframe function started with https://localhost/VSDC.Api/taxcore parameter          taxcore.min.js:1
taxcore iframe created                                                                          taxcore.min.js:1
taxCoreElement clicked                                                                          taxcore.min.js:1
▶ button#taxcore_sign_element                                                                   taxcore.min.js:1
Getting invoice request from pos input element                                                  taxcore.min.js:1
▶ textarea#invoiceRequest                                                                       taxcore.min.js:1
Sending Invoice Request: {                                                                      taxcore.min.js:1
    "DateAndTimeOfIssue": "2017-08-31T13:28:02.433Z",
    "Cashier": "John",
    "BD": null,
    "BuyerCostCenterId": null,
    "IT": "Normal",
    "TT": "Sale",
    "PaymentType": "Card",
    "InvoiceNumber": "31082017-2",
    "ReferentDocumentNumber": null,
    "PAC": null,
    "Options": {
        "OmitTextualRepresentation": 0,
        "OmitQRCodeGen": 0
    },
    "Items": [
        {
            "GTIN": null,
            "Name": "Book",
            "Quantity": 1,
            "Discount": 0,
            "Labels": [
                "A"
            ],
            "TotalAmount": 50
        }
    ]
}
Invoice Request sent to V-SDC.                                                                  taxcore.min.js:1
Online POS Origin: 'https://localhost:4443'                                                   vsdc.sign.min.js:1
message received from V-SDC: {"RequestedBy":"BQRYJA84","DT":"2017-11-                          taxcore.min.js:1
14T07:15:23.817077Z","IC":"1187/1187NS","InvoiceCounterExtension":"NS","IN":"BQRYJA84-J44BRFVW-1187","TaxItems":
[{"Label":"A","Amount":4.1284}],"VerificationUrl":"https://localhost/Frontend.UI/v/?
vl=AUJRUllKOTg0SjO0OlJGVleiBAAAowOAACChBwAAAAFfuWHuKOAAAI1upIU4Nign2ekEWv%2FGkxeO0iIWW5LGDME63ObY79iU%2FkOMtHTOchGvFZhTznAB0
```

Logs written to browser console when data-taxcore-debug is set to true

Subscribe to TaxCore messages

If you need to perform some tasks, after the message from V-SDC is received, you can listen on the following event and do the necessary handling. For example, the following code will be executed after the signed invoice response JSON from V-SDC is written to the appropriate storage location.

```
// Listen to message from taxcore
window.onmessage = function (e) {
    console.log(e.data);
}
```

Message received from V-SDC is well formatted JSON message in the following format:

```
{
 "status_code": 400,
 "response": {
   "Message": "The request is invalid.",
   "ModelState": {
     "invoice.PaymentType": [
       "2805"
     ],
     "invoice.Items[0].Labels[0]": [
       "2310"
     ]
   }
 }
}
```

If the status code is 200, the request is successfully signed with VSDC and the response is a signed invoice in JSON format.

If the status code is not 200, the response filed will contain an error message and error description received from VSDC. You can use the following code to determine the reason.

```
window.onmessage = function (e) {
           var response = JSON.parse(e.data);
           if (response.status_code != '200') {
               var error_reason = response.response;
           }
       }
```

Supported browsers

Online POS and V-SDC are tested with the following browsers:

- Google Chrome 61.0.3163.100+
- Microsoft Internet Explorer 11 (not supported by Microsoft, use it on your own responsibility)
- Microsoft Edge 40.15063.674.0+
- Firefox 56.0.2+
- Opera 48+

All current major browsers are supported.

Example of integration using simple HTML page

The following code is an example of a simple integration. The HTML serves as an integration point for V-SDC. Instead of this HTML, you should use page of your online POS application.

```
<!DOCTYPE html>
<html>
<head>
    <title>Online POS</title>
    <meta charset="utf-8">
</head>
<body>
    <h1>Online POS</h1>
    <label for="invoiceRequest">Invoice request json</label>
    <textarea id="invoiceRequest" cols="100" rows="30" style="display:block"></textarea>
    <label for="taxcore_sign_element">Send Invoice Request:</label>
    <!--TaxCore HTML element-->
    <button id="taxcore_sign_element"
                data-taxcore-vsdc-url=" https://vsdc.sandbox.taxcore.online/"
                data-taxcore-input-id="invoiceRequest"
                data-taxcore-output-id="results"
                data-taxcore-invoice-request=""
                data-taxcore-debug="true"
                data-taxcore-signed-invoice-response="">Sign Invoice</button>
    <label for="results">Received Signed Invoice:</label>
    <textarea readonly id="results" cols="100" rows="30"></textarea>
    <!-- TAXCORE.JS -->
    <script src=" https://vsdc.sandbox.taxcore.online/onlinepos/v1/taxcore.min.js"></script>
    <!-- Custom script at Online POS -->
    <script>
        document.getElementById("invoiceRequest").innerHTML =
                JSON.stringify(CreateExampleInvoiceRequest(), undefined, 4);

        document.getElementById("taxcore_sign_element").dataset.taxcoreInvoiceRequest =
                JSON.stringify(CreateExampleInvoiceRequest());

        // Listen to messages from TaxCore
        window.onmessage = function (e) {
            console.log(e.data);
        }

        function CreateExampleInvoiceRequest() {
            var invoiceRequest = {
                "DateAndTimeOfIssue": "2017-08-31T13:28:02.433Z",
                "Cashier": "John",
                "BD": null,
                "BuyerCostCenterId": null,
                "IT": "Normal",
                "TT": "Sale",
                "PaymentType": "Card",
                "InvoiceNumber": "31082017-2",
                "ReferentDocumentNumber": null,
                "PAC": null,
                "Options": {
                    "OmitTextualRepresentation": 0,
                    "OmitQRCodeGen": 0
                },
                "Items": [
                    {
                        "GTIN": null,
                        "Name": "Book",
                        "Quantity": 1,
                        "Labels": [
                            "A"
                        ],
                        "TotalAmount": 50
                    }
                ]
            };

            return invoiceRequest;
        }
    </script>
</body>
</html>
```

Test Cases

Regardless of the type of invoicing system you are building, the same test cases shall apply.

1. Every Normal sale (NS) is assigned with unique SE ID “SDC Invoice No” consisted of RequestedBy UID – SignedBy UID – OrdinalNumber
2. Refund (NR) is made because of the Sale, so naturally you’ll use NS’s “SDC Invoice No” in ref number field
3. Copy (CS) is made based on Normal receipt, so you’ll use NS or NR “SDC invoice No” in ref number field

Different invoice use-cases and their appropriate labels are presented in the following table

INVOICE TYPE	TRANSACTION TYPE	Invoice label
Normal	Sales	NS
Normal	Refund	NR
Copy	Sales	CS
Copy	Refund	CR
Training	Sales	TS
Training	Refund	TR
Proforma	Sales	PS
Proforma	Refund	PR

Table 2 Invoice Use Cases

Issue Invoice

Receipt must contain visible markings Receipt Type “NORMAL”, “COPY”, “TRAINING” or “PROFORMA”.

Steps

Cashier on Accredited POS selects an invoice type, and then registers the transaction by: typing items, selecting items from previously made list or scanning bar code with a bar code reader. In the end, the cashier chooses the payment method and finishes the invoice.

Accredited POS sends a message to E-SDC. After a successful invoice data verification, the invoice is signed, counters and totals are updated and internal data is completed.

E-SDS sends back an Invoice response to Accredited POS.

Receipt is delivered to the customer.

Expected Result

Fiscal receipt is the final result of this procedure. The receipt can be printed or sent by SMS or email message if a customer is asking for it. Every receipt is digitally signed. Internal data is stored in the TaxCore database. Valid QR code is at the end of the receipt. Receipt counter is in the form a/b IL (a-total number of signed invoices per type/ b-total number of signed invoices, IL – Invoice label).

Accredited POS sends a message to E-SDC. After successful invoice data verification, the invoice is signed and invoice counters are updated.

E-SDS sends back an Invoice response to Accredited POS.

Receipt is issued.

Issue Normal Sale or Refund B2B Invoice

B2B invoice supports all invoice types and transaction types. At a beginning of invoice creation cashier must ask the buyer for Buyers TIN, and optionally Buyer Cost Center.

Special Cases

In addition to all the above mentioned, any combination of the previous test cases is possible.

All payments shall conform to the following rules:

- If the payment is not considered as a daily turnover, then no fiscal receipt shall be issued for that payment (as it is not an actual transaction).
- If the payment is considered as advance payment, then the fiscal receipt shall be issued as a NORMAL-SALE transaction.

The following examples illustrate applications of these rules:

Deposit

Depending on the purpose of a taken deposit, this case can be resolved in two ways:

1. If the Deposit is subject of some internal agreement/document (not considered as a daily turnover), then no fiscal receipt shall be issued for the Deposit. Only final SALE is a NORMAL-SALE transaction.
2. If the Deposit is considered as advance payment, then the fiscal receipt shall be issued as a NORMAL-SALE transaction. It is advised to name this Item similar to “40% of an Item”.

When Items are ready for Delivery and final SALE, POS shall provide one or both of the following options:

1. Create another transaction (“60% of an Item”), or
2. Create another two transactions: NORMAL-REFUND (“40% of an Item”) and NORMAL-SALE (“Item”).

Quotes

Assuming no advance payment takes part in the transaction, this is a PROFORMA-SALE transaction.

Layby / Installments

This is an advance payment and shall be treated as NORMAL-SALE, and fiscal receipts shall be issued. It is advised to name each installment similar to “10% Item”. When the last installment payment is made, POS shall provide one or both of the following options:

1. Create another transaction for the last installment (“15% Item”), or
2. For each previous paid installment create transactions of type NORMAL-REFUND (“X% Item”), followed with one NORMAL-SALE (“Item”).