Technical Documentation

This section contains TaxCore technical documentation.

1. EFD Vendors
   This documentation puts forward detailed technical instructions for all EFD vendors. Make sure you read them carefully if you plan to develop an invoicing system (POS) or an E-SDC solution.

2. Licenses
   This section describes the different licenses that are used within the TaxCore system.

EFD Vendors

This documentation puts forward detailed technical instructions for all EFD vendors. Make sure you read them carefully if you plan to develop an invoicing system (POS) or an E-SDC solution.

Only solutions that are fully compliant with these instructions will be officially accredited by the accreditation authority.

Components of Electronic Fiscal Device

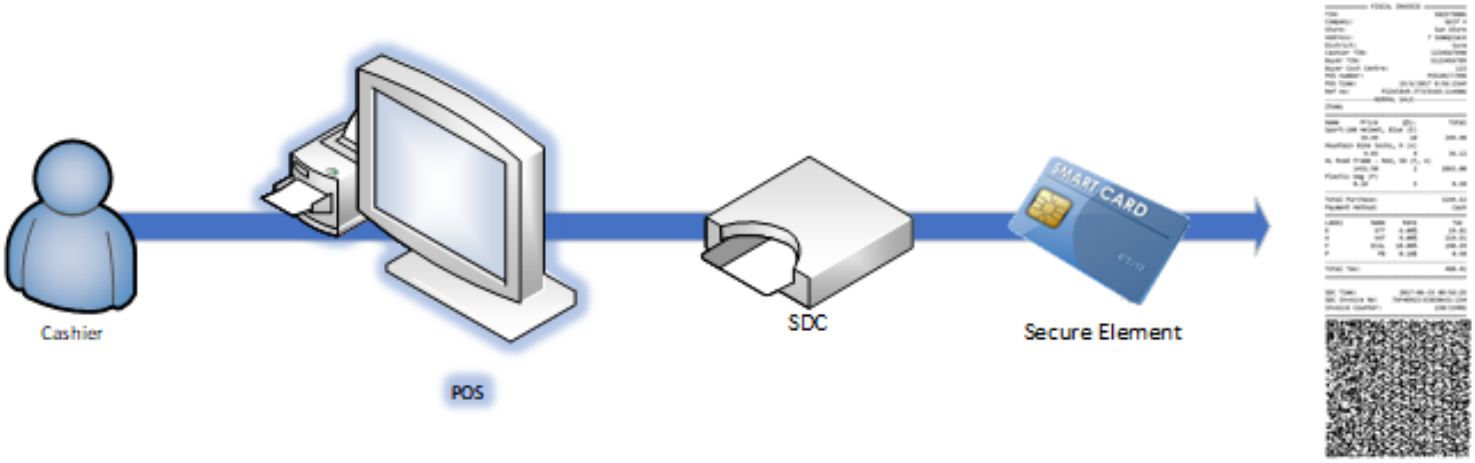
Electronic Fiscal Device (EFD) is composed of an Invoicing System (POS), an SDC and a Secure Element, all connected into one system. EFD produces fiscal receipts and reports Audit Data to a tax authority.

There are different options for a taxpayer's EFD setup. Every taxpayer can decide which EFD setup best suits their business needs.

Each EFD consists of three logically separated components.

POS, SDC, and Secure Element must communicate using protocols published as part of this documentation. Protocols are public and subject to change. Please familiarize yourself with Versioning and End of Life policies.

POS (or Invoicing System)

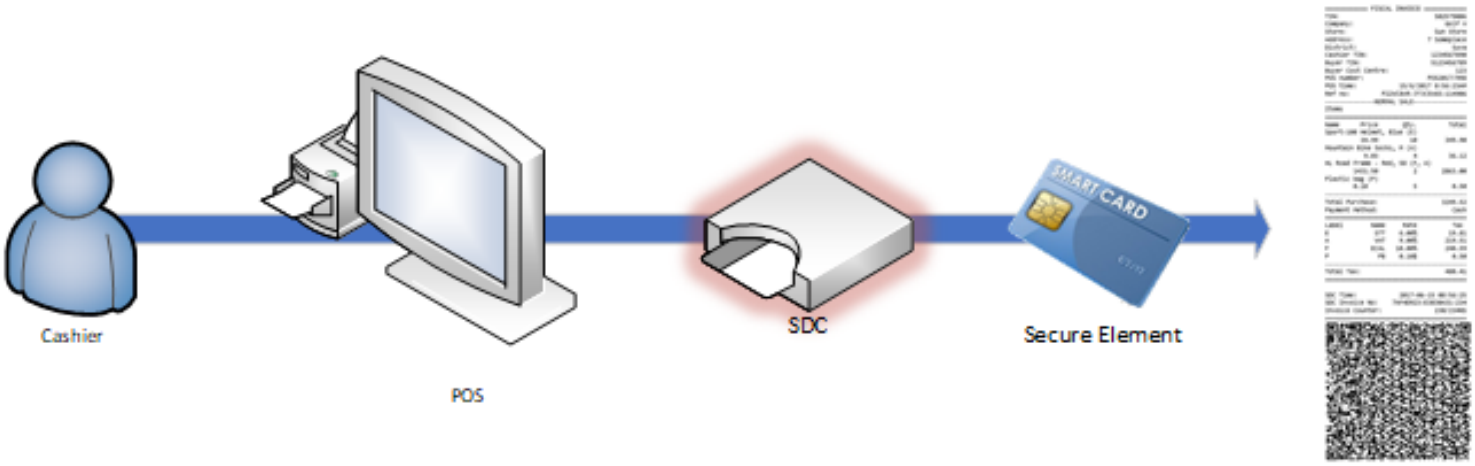


Invoicing System or POS means a point-of-sale invoicing device or software used by a business for:

- management control in the areas of sales analysis and stock control
- entering the transaction data for each transaction made by the business
- submitting the transaction data to an SDC service
- issuing fiscal invoices to customers after receiving the fiscalized data back from the SDC.

Invoicing System or POS are provided by 3rd Party Vendors and are subject to accreditation.

SDC



Sales Data Controller (SDC) is the hardware or software component of an EFD that:

- receives transaction data from a POS component of the EFD;
- analyzes the transaction data into fiscal data;
- formats the fiscal data as a fiscal invoice, creates the digital signature for the EFD, and records the digital signature on the fiscal invoice;
- transmits the fiscal invoice to the POS;
- preserves the transaction data and fiscal data in an irrevocable and secure manner;
- transmits the fiscal data to the Authority's system;
- communicates with Secure Element, TaxCore.API, and POS to configure itself, fiscalize invoice and transmit audit data.

V-SDC

Virtual Sales Data Controller (V-SDC) is a web service operated by the tax authority that enables authorized

taxpayers to use SDC functionality via the Internet.

V-SDC is limited to Connected Scenarios, which means that any invoicing system that targets V-SDC must have a reliable internet connection at the moment of sale.

Check V-SDC for more details.

E-SDC

External SDC (E-SDC) is a 'black box' type of software or hardware that communicates with a smart card secure element and enables semi-connected fiscalization scenarios (enables issuing fiscal invoice when the internet is down).

It resides at taxpayers' business locations and communicates with a Secure Element issued to taxpayers on a smart card (every smart card has its own secure element and a taxpayer can have more than one smart card for issuing fiscal invoices). In other words, E-SDC uses the taxpayer's secure element to place a digital signature on the fiscal invoice.

E-SDC are provided by 3rd Party vendors and are subject to accreditation.

Dev E-SDC

Development E-SDC is a software version of an E-SDC used by POS developers. It is built according to the latest technical specification for E-SDC devices and is used to develop, test, and accredit invoicing solutions.

Development E-SDC is available to all registered vendors via the Developer Portal.

It simulates the operation of an E-SDC on a local network in the production environment, so vendors can upgrade their applications or devices without obtaining any physical E-SDC device or smart card.

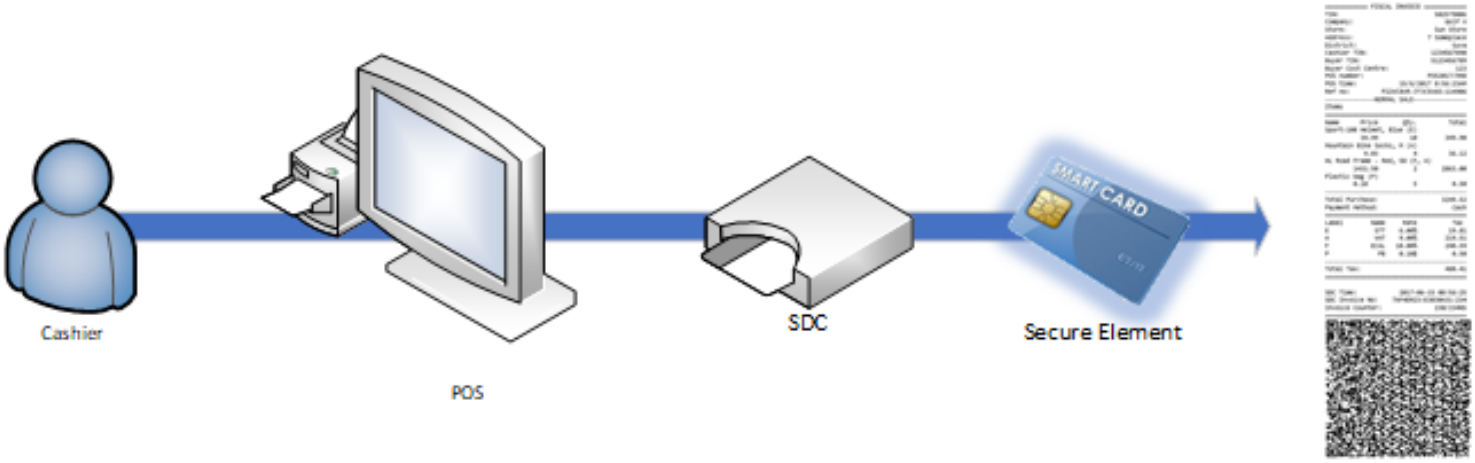
Development E-SDC is also used during the accreditation process to check whether an invoicing system is functional.

Secure Element

Secure Element (SE) is a software and hardware used by an EFD and the Authority to prevent tampering and unauthorized use of fiscal data transmitted to the Authority’s system, as well as to maintain the integrity of the fiscal data.

SEs are provided to Taxpayers by [*TaxCore.PublicConfiguration.Organization*]]. Authorized persons of Taxpayers may request additional secure elements via [Taxpayer Admin Portal.

It may be implemented in two formats: SmartCards and PKCS12 files.



Building POS and E-SDC as one device or application

These are the accreditation conditions you need to be aware of if you are planning to develop an all-in-one POS + E-SDC solution:

- The solution must pass all the tests with the SDC Analyzer Win App during the accreditation process. It means that endpoints documented in POS to SDC protocol related to E-SDC must be accessible to external callers during accreditation.
- once accreditation is completed, the POS to SDC protocol may be omitted from final documentation and made inaccessible to external callers.

What can you find in this documentation?

The documentation consists of a *Getting Started With Accreditation* article describing basic facts and steps related to accrediting a POS or E-SDC solution. There is also *General Information* section, common for all EFD vendors.

Finally, there are two separate sections with specific information for POS and E-SDC developers.

Please see below for lists of documentation sections according to vendor type.

Sections for POS vendors

- Getting Started With Accreditation
- General information
- For POS developers

Sections for E-SDC vendors

- Getting Started With Accreditation
- General information
- For E-SDC developers

Changelog

All notable changes to technical instructions for EFD vendors will be documented on this page.

2025-08-25

Added

- From **SE applet version 3.2.10**, if the SDC Time on an invoice is outside of the secure element's certificate validity period (i.e., before the NotBefore limit or after the NotAfter limit), the secure element returns error code 0x6308 - Secure Element Applet API and Secure Element Specific APDU Error Codes

Updated

- Examples of invoices response for Refund-type invoices now contain the invoice filed **Total Refund** instead of **Total Purchase** - see Normal Refund, Advance Refund, Copy Refund, Proforma Refund or Training Refund

2024-12-17

Added

- Instructions for extracting certificate expiration date - Extracting Expiration Date from Digital Certificate
- Instruction for E-SDC to verify the current date/time against certificate expiration date/time when fiscalizing invoices - see Fiscalization of an invoice
- New error code in POS-SDC communication protocol used by E-SDC to report that invoice can not be fiscalized with an expired certificate - see Status and Error codes

Updated

- Instructions for E-SDC not to execute the PIN Verify APDU command again as long as the communication session between the E-SDC and the smart card is open - see Enter PIN to Unlock the Secure Element
- Instructions for Invoicing System (POS) to preferably obtain configuration parameters from an SDC rather than TaxCore.API. See the pages below form more details:
    - Configuration
    - Environments
    - Tax Rates

2024-02-20

Updated

- Redesigned SE applet documentation and added commands for different versions with examples. See the pages below for more details:
    - Secure-Element-Applet-API
    - General-Commands
    - Fiscalization
    - Audit

2023-09-19

Updated

- Displayed price in fields **unitPrice** and **totalAmount** - submitted by an invoicing system to an SDC, as part of the invoice request - should show the price after the discount is applied (if it exists). It must not include the value of the discount - see Create Invoice.

2023-03-14

Updated

- Description for the **expiresAt** property as a part of TaxCore.API response when requesting authentication tokens. The description now clearly states that the property is in UTC time - see Request Authentication Token

2022-12-22

Added

- New statuses for TaxCore API response to an E-SDC after submitting audit packages - see Submit Audit Package

2022-11-02

Added

- Note regarding possible implementation of non-standard strategy for issuing Proof of Audit, in certain tax jurisdictions - see Proof of Audit

2022-09-16

Added

- Explanation of the case when **0x6A80** error code is received after sending the ***Sign Invoice*** APDU command to the Secure Element Applet API - see Secure Element Specific APDU Error Codes.

2022-09-02

Added

- Minimum and maximum values for date and time - see Create Invoice and Date and Time

2022-07-27

Added

- Notes regarding audit package deletion from E-SDC's local storage, after receiving a status response from TaxCore.API - see Creating an Audit Package and Submit Audit Package

2022-06-16

Changed

- Character encoding standard for the `buyerId` field in invoice request has changed from ASCII to Unicode - see Create Invoice
- Exact number of bytes (with different options) for all fields in Create Verification URL

2022-05-18

Changed

- Updated instructions for submitting payment amounts in the `payment` field of the invoice request - see Create Invoice
- *Export Internal Data* APDU command was replaced by the *Export Audit Data* command - see Audit

2022-04-21

Changed

- Updated the description of the secure element error code **0x63FF** in Secure Element Specific APDU Error Codes

2022-04-11

Changed

- clarified the character encoding standard for fields `referentDocumentNumber` and `mrc` in Create Invoice.

2022-04-07

Changed

- **UTC** standard is used only in the `sdcDateTime` value when creating an audit package to be sent to TaxCore.API.

2022-04-04

Changed

- Added clarification that the Reference time (field `referentDocumentDT`) must be submitted as the local date and time in ISO 8601 format - see Create Invoice

2022-02-08

Changed

- Updated the description of the content of the `{UID}.arp` file in E-SDC Stores Audit Files on SD card or USB Drive

2022-02-04

Changed

- Properties in the Audit Proof Request are now mandatory - see Format of the Audit-Proof Request

2022-01-31

Added

- RequestId in invoice request header is limited to 32 characters - see Create Invoice
- Note that the values for SDC time and all other fields used during invoice creation and submission must match in the audit package (Submit Audit Package), data forwarded to the Secure Element for digital signing (Fiscalization) and the Verification URL Create Verification URL

2022-01-24

Added

- List of important endpoints for development and production environments in Serbia in Identification of Environments and Important Endpoints

2022-01-14

Changed

- Updated minimal QR code size in Create a QR Code.

2021-12-20

Added

- Added an extra column in Secure Element Specific APDU Error Codes (`Error Code to POS`) to create mapping between APDU error codes and the general error codes in Status and Error Codes

Changed

- Enriched the description of status and error codes in Verify PIN service

2021-12-15

Changed

- Fields `omitQRCodeGen` and `omitTextualRepresentation` in invoice requests must use case case - see Create Invoice

2021-11-26

Changed

- Updated the recommendation for Audit Start frequency and enriched the explanation of different Audit cases in Proof of Audit

2021-11-18

Added

- A note that audit packages should be submitted continuously, regardless of the Audit cycle - see Proof of Audit

2021-11-09

Added

- Structure of files/folders for different file transfers during `Local Audit` in File-Based Communication protocol

Changed

- Enriched instructions for storing `last signed invoice(s)` when using E-SDC with multiple POSs and multiple Secure Elements - see Get Last Signed Invoice

2021-10-28

Changed

-

Enriched instructions for dumping audit files from E-SDC to external memory units in E-SDC Stores Audit Files on SD Card or USB Drive

Added

- Specification for AES encryption in Creating an Audit Package

2021-10-22

Changed

- Updated the number of bytes for the ARP file in Audit Process

2021-10-19

Changed:

- Updated maximum length for the `buyerCostCenterId` field in Create Invoice
- Renamed command `Get Api Version` into `Get Secure Element Version` in General Commands
- Updated description of the format for the fields `Taxpayer ID` and `Buyer ID` in Fiscalization commands

Removed

- Designation ***optional*** for model elements in Submit Audit Request ARP. All elements are now described as mandatory.
- `Get Audit Version` command from General Commands

2021-10-14

Changed:

- Added `64bit unsigned` designation for `sum` and `limit` integer values in Format of the Audit Proof Request and Submit Audit Request Payload - ARP
- Moved the content of the article ***Commands Execution Result*** into its parent article Commands
- Renamed all child articles in Commands for better clarification

2021-10-01

Changed:

- Redefined `cashier` as an optional field for Invoice Request in Create Invoice
- Removed `Content-Type` element from HTTP header for all GET operations

2021-09-22

Changed:

Updates in Verify PIN service:

- Quotation marks are removed from example

2021-09-20

Added:

- Note about different scenarios of POS submitting RequestID to E-SDC and its connection to using the Get Last Signed Invoice command
- *Export Internal Data* APDU command in Audit

2021-09-16

Added:

- Changelog page
- New status code (2110) as a possible response for Verify PIN service
- Updates in Get Status service:
    - o new response fields in the table (*currentTaxRates* and *allTaxRates*)
    - o response model
    - o response example with E-SDC
- Added definitions for minimum and maximum field length for Invoice Request in Create Invoice
- Request model in Submit Audit Request Payload - ARP
- Table with mandatory and optional usage of the Reference Number

Changed:

- Fixed broken links on the EFD Vendors page
- Property examples in Error Messages Format
- Updates in Create Invoice:
    - o V-SDC endpoint example
    - o description of the *Accept-Language* header for Invoice Request
- Updated explanation and examples for tax calculation algorithm in Calculate Taxes
- Updated field descriptions in Format of the Audit Package and Format of the Audit-Proof Request

Removed:

- *Export Internal Data* APDU command in Audit

Differences Between TaxCore v2 and v3

This article describes the main **differences between TaxCore v2 and TaxCore v3**. Its goal is to help the vendors/developers of EFD components to adapt their solutions that currently operate on the v2 system or create new ones, specifically for v3.

Specific jurisdiction requirements are described in the *Technical Guideline* document, published and updated by each jurisdiction independently.

Updates affecting both Invoicing Systems (POS) and E-SDCs

- New invoice types - **Advance Sale** and **Advance Refund**
- E-SDC should support new culture parameters (en-US) for localization of the invoice journal returned to the POS - affects the date format in the invoice journal
- Two endpoints added:
  - **api/v3/tax-rates** - returns information about defined tax rates for the environment
  - **api/v3/encryption-certificate** - returns the public key of the TaxCore.API which can be used for encryption
- POS to SDC protocol updates
  - **api/Status/GetTaxAuthorityParams** replaced with **api/v3/environment-parameters**
    - returns information about the configuration of the environment
    - can be used with both E-SDC and V-SDC
  - **api/Sign/SignInvoice** replaced with **api/v3/invoices**
    - new optional invoice field in the request - Reference Time (referentDocumentDT)
    - multiple payment types on one invoice
    - additional HTTP headers
    - updated response model from SDC
  - **api/Status/GetStatus** replaced with **api/v3/status**
  - **api/Status/VerifyPin** replaced with **api/v3/pin**
  - **api/Status/Attention** replaced with **api/v3/attention**
  - **api/Sign/GetSignedInvoice** replaced with **api/v3/invoices/{requestId}**
  - updated model for **Error message format**

Updates specific to Invoicing Systems (POS)

- New functional requirements such as **Transaction report (daily report)** and support for **Invoice cancelation** - specific content and requirements are more closely described in each jurisdiction's

Technical Guideline document

- POS to V-SDC client authentication
    - Updated **authentication to V-SDC service** - secure element PAC is submitted in the HTTP request header

Updates specific to E-SDC

- New testing app - **SE SignInvoice**
    - Used for testing sending APDU commands to smart card secure elements
    - Available via the Developer Portal

- E-SDC to TaxCore.API protocol updates
    - **api/SDC/RequestAuthenticationToken** replaced with **api/v3/sdc/token**
    - **api/SDC/GetInitializationCommands** replaced with **api/v3/sdc/commands**
    - **api/SDC/NotifyOnlineStatus** replaced with **api/v3/sdc/status**
        - ♠ the request must contain the string “true”
    - **api/SDC/NotifyCommandProcessed** replaced with **api/v3/sdc/commands/{commandId}**
        - ♠ the request must contain the string “true”
    - **api/SDC/SubmitAuditData** replaced with **api/v3/sdc/audit**
        - ♠ values for *sdcDateTime* and all other fields must match the values submitted to the Secure Element for digital signing and the values in the verification URL
        - ♠ extended the list of invoice statuses returned as part of the response from TaxCore.API
        - ♠ if status 4 (Invoice is verified) is received from TaxCore.API, that audit package should immediately be deleted from the E-SDC's local storage. Otherwise, the audit package must not be deleted.
        - ♠ the E-SDC should try to resubmit an audit package to TaxCore.API. only if it receives status 1 (Invoice cannot be stored) from the TaxCore.API. Otherwise, the audit package should not be resubmitted.
    - **api/SDC/StartProofOfAudit** replaced with **api/v3/sdc/audit**
        - ♠ updated model for request – added “sum” and “limit”

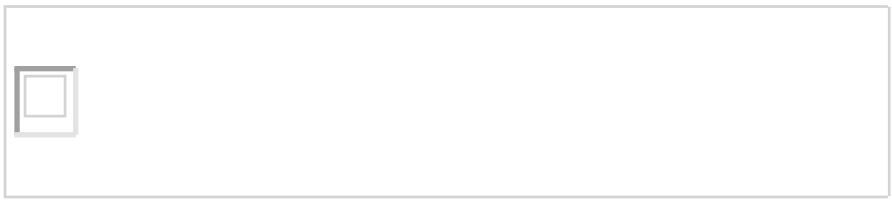
- E-SDC to Secure Element protocol updates
    - a **cyclic redundancy check (CRC)** can optionally be used to eliminate accidental communication errors
    - updated **General commands**
        - ♠ added new commands: **Forward Secure Element Directive**, **Get Last Signed Invoice** and **Get PIN tries left from SE Applet**
    - updated **Fiscalization commands**
        - ♠ PIN is sent in ASCII hex format for the **PIN Verify** command
    - updated **Audit commands**
    - updated Secure **Element Specific APDU Error Codes**
        - ♠ added new error codes

- File-based Communication (offline)
    - updated instructions for storing files on the removable memory unit

        - ♠ **E-SDC stores audit packages on a removable unit**

♠ **E-SDC receives commands from a removable unit**

♠ **E-SDC stores command execution results on a removable unit**



- Data structure updates
  - Audit
    - ♠ updated **Format of Audit Package** – added field ***invoiceNumber*** (SDC invoice number)
    - ♠ added documentation pages for **Format of Audit Data** and **Format of Audit Proof Request**
  - Commands (TaxCore.API to E-SDC)
    - ♠ **Set tax rates** – updated content (added historical tax rate groups) and format
    - ♠ added documentation for the **Set TaxCore Configuration** command and Forward Secure Element Directive command

- Standard E-SDC operation updates
  - updated instructions for **Create Verification Url**
  - updated min QR code size in **Create QR code**
  - updated **Create a textual representation of an invoice**
    - ♠ added - invoice journal must be in the audit package
  - added documentation pages for:
    - ♠ extracting Taxpayer Identification Number (TIN) from Digital Certificate
    - ♠ extracting secure element UID from Digital Certificate
    - ♠ extracting taxpayer information from a digital certificate
    - ♠ E-SDC execution of commands from VMS
    - ♠ syncing Date and Time on E-SDC

Getting Started With Accreditation

Introduction

According to fiscalization regulations, all fiscalized taxpayers must use an Electronic Fiscal Device (EFD) to issue

fiscal invoices.

By definition, each EFD setup consists of three components:

- Accredited invoicing system - POS
- A version of SDC service - V-SDC or an accredited E-SDC
- Secure element

Accreditation is the process through which EFD components become accredited in a specific tax jurisdiction. Only accredited EFD components can be sold to, and used by, taxpayers.

NOTE:
Beside this introductory *Getting started* document, make sure you also read the General Information section with useful information for both POS and E-SDC vendors.

For examples of different types of EFD setupst - see Electronic Fiscal Device.

For useful initial information specific for POS or E-SDC development, see For POS Developers and For E-SDC Developers.

Which EFD components are the subject of accreditation?

Only POS and E-SDC products are subject to accreditation. They can be developed by any registered vendor and, upon successful accreditation, sold to taxpayers as official fiscalization tools.

NOTE:
Taxpayers can also accredit their own POS or E-SDC product, but those product can not be offered and sold to other taxpayers. For more information, see the distinction between **transferable** and **non-transferable** accreditation types in Accreditation.

The production of V-SDC service and secure elements is under the full control of the tax authority. These EFD components are not the subject of accreditation.

NOTE:
POS and E-SDC components can also be developed as *all-in-one* product/device, incorporating functionalities of both components into one solution. However, in this case the solution must enable full review of both POS and E-SDC features - i.e. the reviewers must be able to test all required POS and E-SDC functionalities.

Accreditation Process From the Perspective of an EFD Vendor

The process of accrediting a POS or E-SDC product consists of the following steps:

1. EFD vendor registers on the designated development environment. During registration the vendor receives the Developer Authentication Certificate (in digital file format) which is used to log into the Developer Portal.
2. The EFD Vendor uses the Developer Portal to request additional secure elements (smart cards or digital files) if those are required for development and testing purposes.
3. The EFD vendor uses the Developer Portal resources (documentation and applications) to develop and test a POS or E-SDC product to confirm that it operates according to the technical instructions.
4. After completing the development and testing, the EFD vendor applies for accreditation via the Developer Portal.
5. Technical review authority checks the technical part of the application, i.e. performs the technical review.
    - If there are any issues with the answers in the application, a technical reviewer requests amendments from the vendor.
    - The vendor amends the required answers and resubmits the application.
6. When all the answers in the technical part are satisfactory, the technical review authority approves the technical part of the application.

NOTE:
Unless stated otherwise, the technical review is jurisdiction-agnostic. That means that one version of one POS or E-SDC product needs to pass the technical review only once. When it passes, the technical review is valid for all jurisdictions. However, every new product or a new version of an existing product has to go through the same technical review from the start.

7. EFD vendor applies for accrediting the product in a specific jurisdiction(s), i.e. chooses where the product will be sold/used.
8. Accreditation authority checks the administrative part of the application, i.e. performs the administrative review.
    - If there are any issues with the answers in the application, an administrative reviewer requests amendments from the vendor.
    - The vendor amends the required answers and resubmits the application.
9.

When all the answers in the administrative part are satisfactory, the accreditation authority approves the administrative part of the application.

NOTE:
Administrative review is always jurisdiction-specific. That means that every product and every new product version has to pass a separate administrative review process for each jurisdiction.

10. The details of the accreditation process are forwarded to the accreditation authority's Technical Committee for final review and confirmation.
11. If the Technical Committee's final review is positive, the product is accredited for that jurisdiction.

Development vs. production environment

All development and testing are performed in the designated testing environment. This environment also used to apply and obtain accreditation for your product.

Once accredited, your product(s) will be used by taxpayers in the production environment.

NOTE:
For more information about the environments, see Identification of Environments and Important Endpoints.

General Information

This section contains information relevant to all EFD vendors.

After you read it, you can continue to vendor-type-specific sections for E-SDC or POS vendors.

1. Development Environment
   The Sandbox environment is accessible to all registered EFD vendors. The Sandbox Environment exposes the same APIs and uses the same protocols as the Production environment.

2. Obtaining RCA and ICA Certificates From a PFX Certificate
   After you complete the process of EFD vendor registration in the Sandbox environment, you will receive a Developer Authentication Certificate (PFX file) for logging into the Developer Portal.

3.

Identification of Environments and Important Endpoints
After the process of registration, all EFD vendors have access to the development and testing environment - Sandbox - via the Developer Portal.

4. Data Structures
This Section contains specifications and an explanation of different data types and formats shared between E-SDCs and V-SDCs

5. Protocols
This section contains the specification of protocols used by both E-SDCs and POSes

Development Environment

The Sandbox environment is accessible to all registered EFD vendors. The Sandbox Environment exposes the same APIs and uses the same protocols as the Production environment.

Obtaining Additional Test Certificates

EFD Vendors can apply to the tax authority and additional testing certificates in the form of smart cards or digital certificates (PKCS 11 format). They can use these certificates for development, integration and testing purposes. With additional certificates, users can test both successful and failing scenarios (like trying to fiscalize an invoice with an expired certificate).

The process of requiring additional certificates involves these steps:

1. EFD vendor registers with tax authority through an official application form (see Registration to Developer Portal)
2. EFD vendor uses the developer certificate obtained during registration to log in to their profile on the Developer Portal.
3. EFD vendor uses the Developer Portal to request the desired number of additional certificates (see Requesting additional certificates) for each registered location.
4. tax authority officers review the request and if everything is ok, approve issuing additional certificates.
5. EFD vendor picks up the additional smart card certificate or receives the additional digital file certificate via email.

Obtaining Smart Cards

Accredited POS Vendors shall apply to the tax authority 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 authority using an application form published on web site (see Registration to Developer Portal)
2. tax authority verifies the application and sends the enrolment request to the primary contact by e-mail or SMS

3. Primary contact enters the desired PIN code for their smart card
4. tax authority delivers the smart card to the Primary contact
5. If additional smart cards are required, the Primary contact can submit a request using the Developer Portal on the Sandbox environment.

Identification of Environment

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

A Sandbox is a testing environment that isolates untested code changes and experimentations 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
4. Serbia - SUF
5. Rebulic of Srpska - SUF
6. Vanuatu - VSMS
7. Papua New Guinea - GMS

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

Obtaining RCA and ICA Certificates From a PFX Certificate

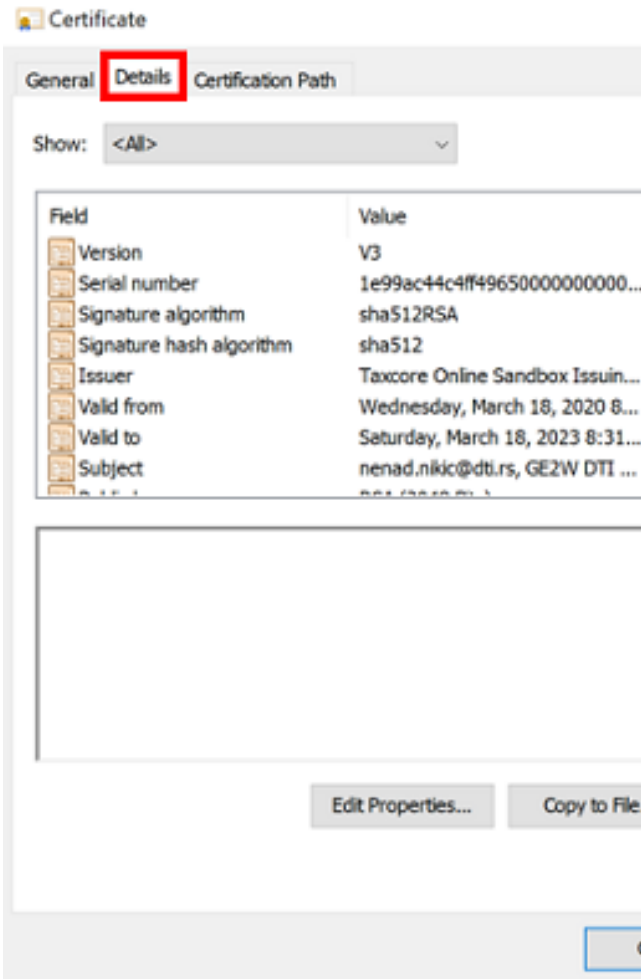
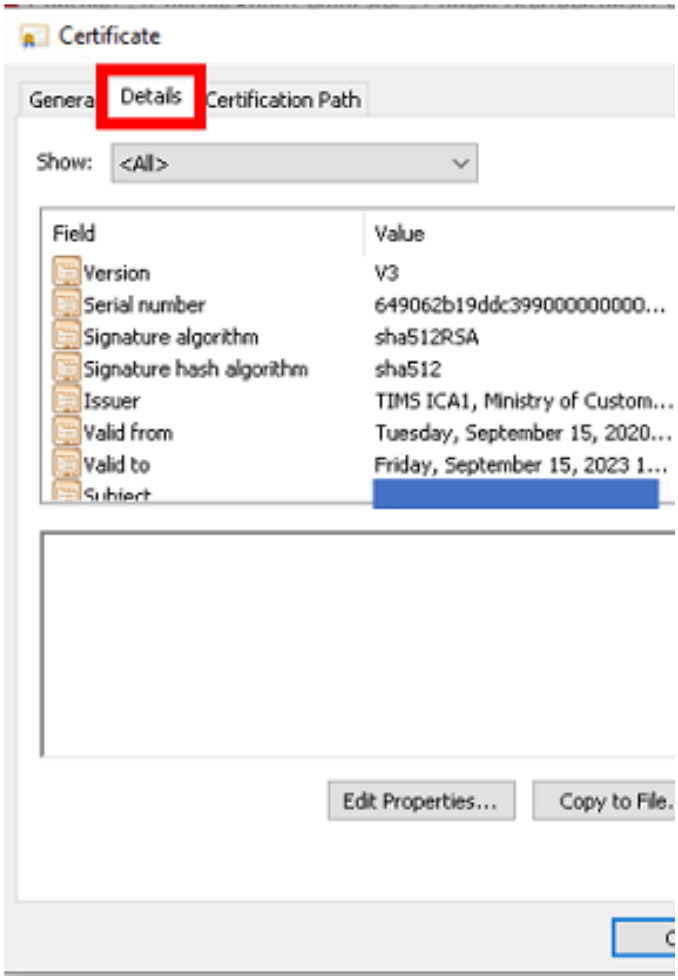
After you complete the process of EFD vendor registration in the Sandbox environment, you will receive a Developer Authentication Certificate (PFX file) for logging into the Developer Portal.

Also, taxpayers operating in the Production environment will use PFX files as additional certificates for issuing invoices.

If you wish to obtain the RCA and ICA certificates which were used to issue a PFX certificate, follow these steps:

1. Open the PFX certificate (check the installation instructions if you need a reminder of where you installed

the certificate) and click on the **Details** tab.

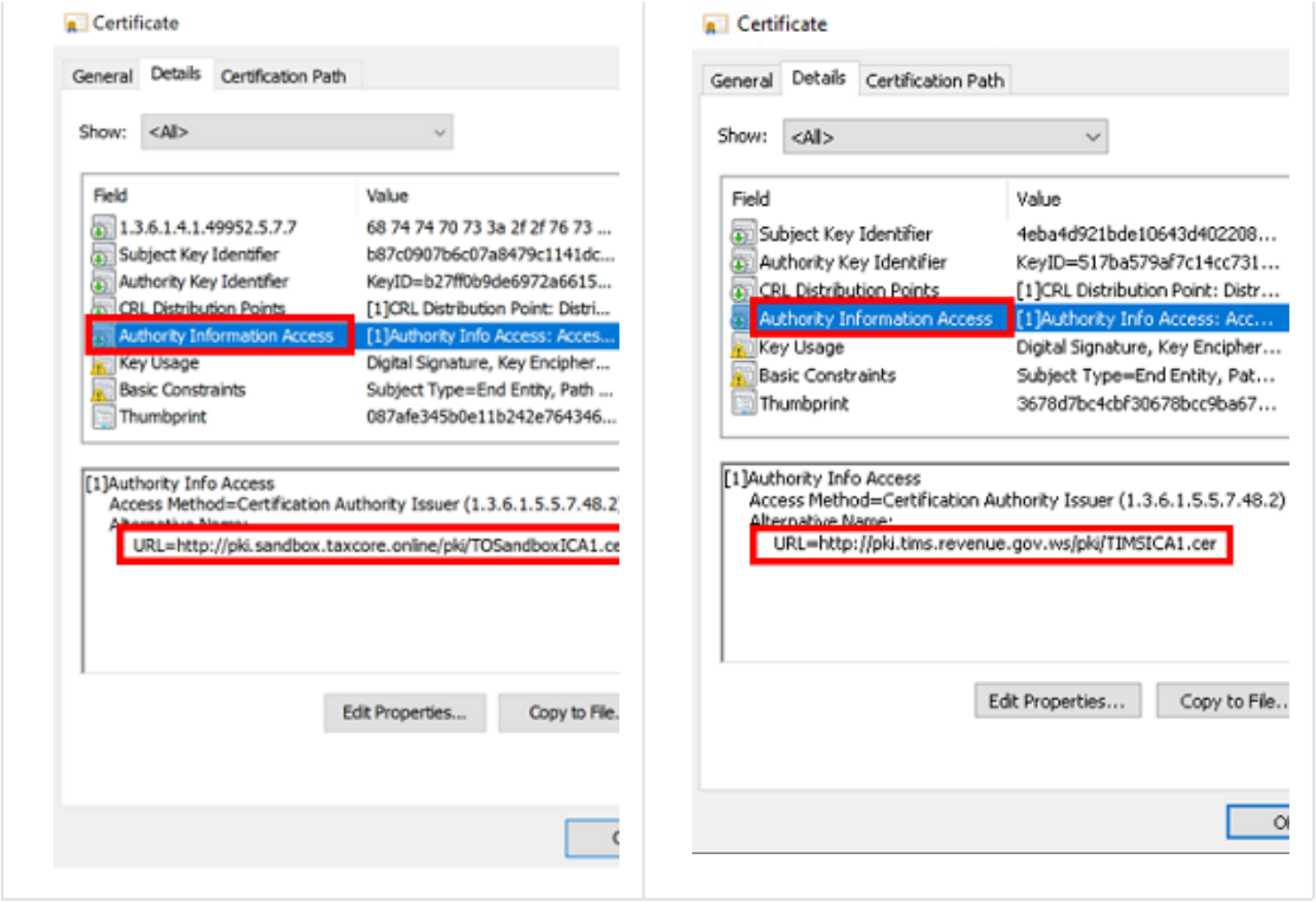
Sandbox environment	Production environment
	

Obtaining RCA and ICA Certificates From A PFX Certificate – Image showing the Details tab of the PFX certificate

NOTE:
All the images in this article display TIMS as the name of the certificate authority. This is used only as an example. Every jurisdiction will have a different certificate authority name.

2. Scroll down and click on **Authority Information Access**. The text box below will display the URL for downloading the ICA certificate which signed the PFX certificate.

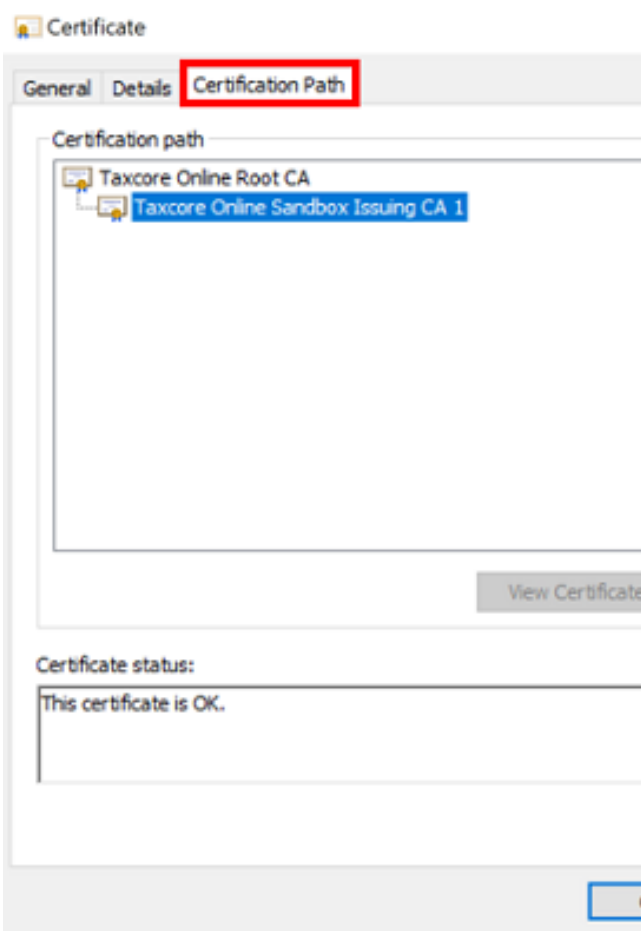
Sandbox environment	Production environment

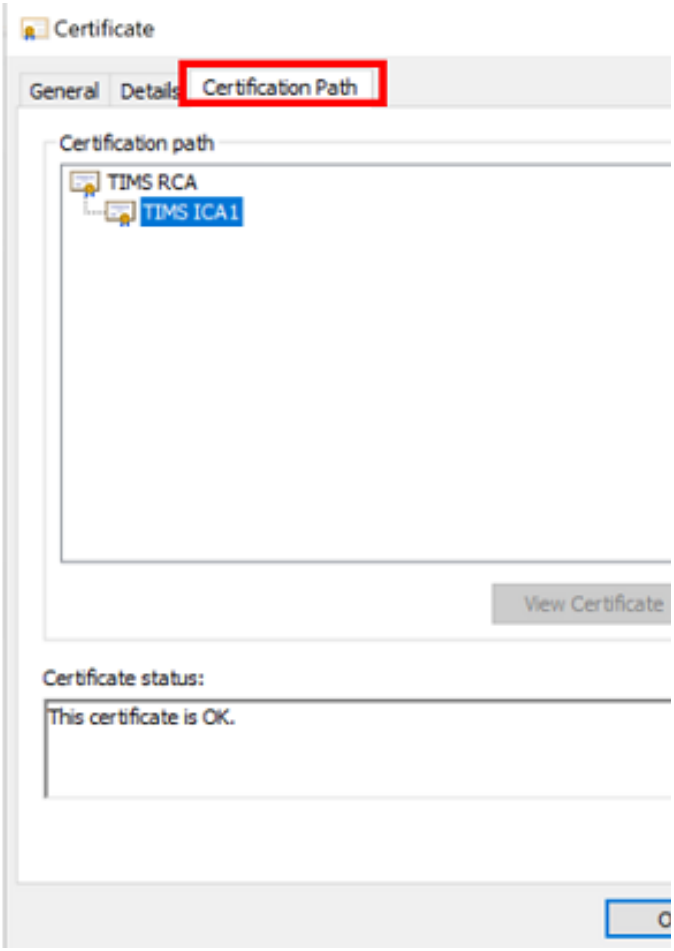


Obtaining RCA and ICA Certificates From A PFX Certificate – Image showing the Authority Information Access of the details tab in the Certificate section

3. Download and install the ICA certificate according to the instructions.
4. After you download the ICA certificate, you can check its status by opening it and clicking on **Certification Path**. It will be displayed hierarchically under the RCA certificate which signed that ICA certificate.

Sandbox environment	Production environment

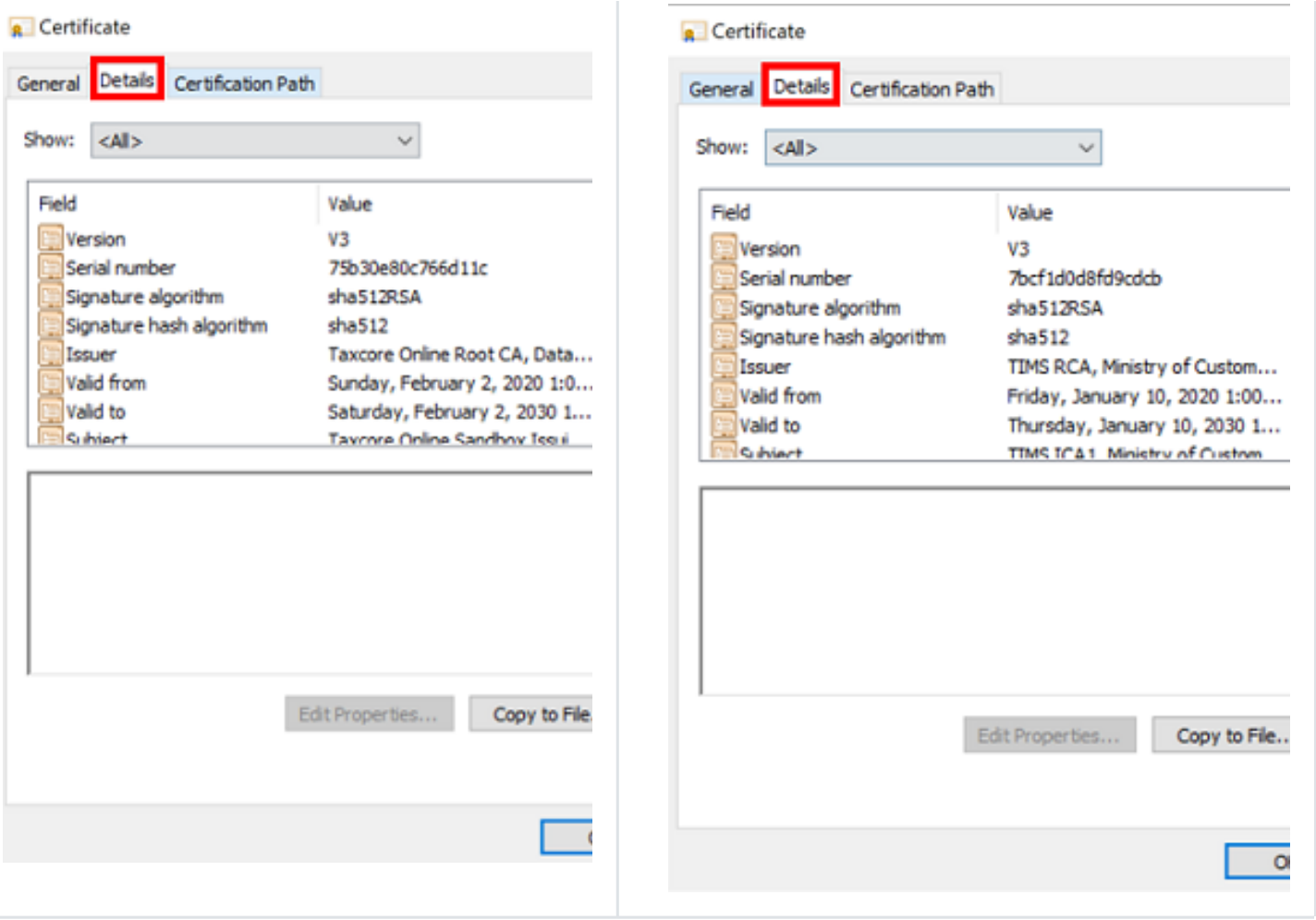




Obtaining RCA and ICA Certificates From A PFX Certificate – Image showing the Certification Path tab of the PFX certificate

5. To download that RCA certificate, open the ICA certificate and click on **Details**.

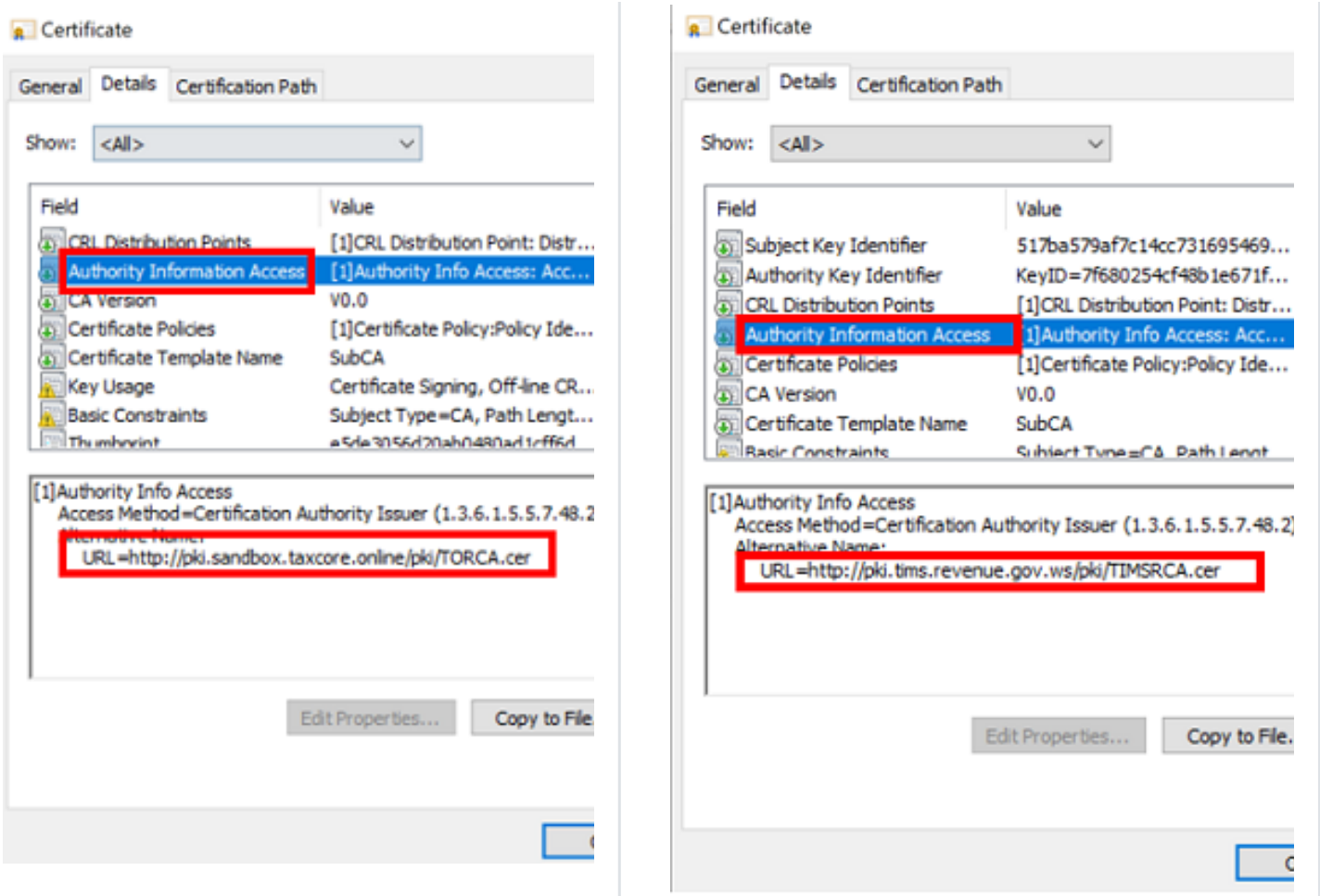
Sandbox environment	Production environment



Obtaining RCA and ICA Certificates From A PFX Certificate – Image showing the Details tab of the PFX certificate

6. Again, scroll down and click on **Authority Information Access**. The text box below will display the URL for downloading the RCA certificate which signed this ICA certificate.

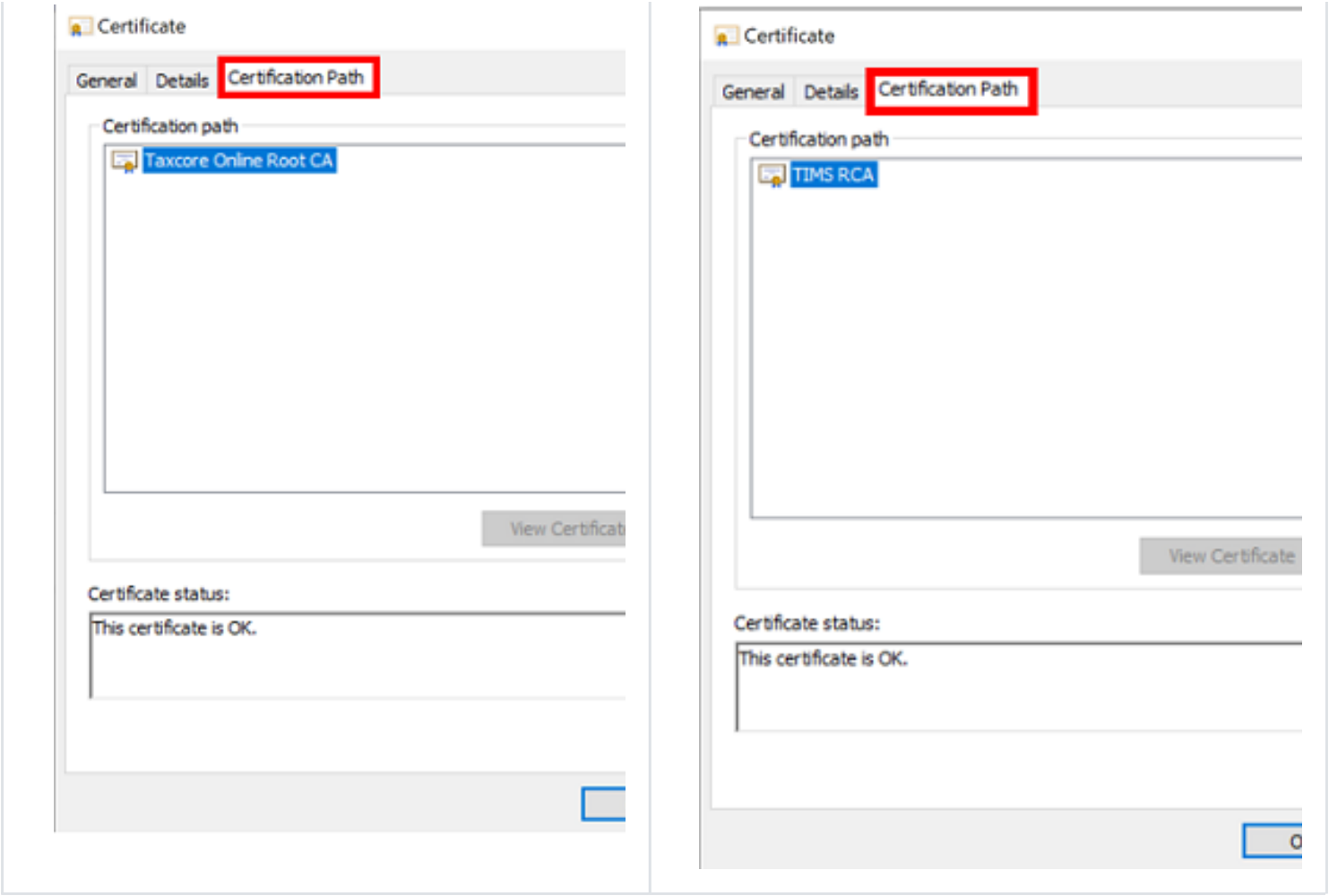
Sandbox environment	Production environment



Obtaining RCA and ICA Certificates From A PFX Certificate – Image showing the Authority Information Access of the details tab in the Certificate section

7. Use this URL to download and install the RCA certificate according to the instructions.
8. After you download the RCA certificate, you can check its status by opening it and clicking on **Certification Path**. It will be the only certificate displayed because it is self-signed.

Sandbox environment	Production environment



Obtaining RCA and ICA Certificates From A PFX Certificate – Image showing the Certification Path tab of the PFX certificate

Identification of Environments and Important Endpoints

Development environment

After the process of registration, all EFD vendors have access to the development and testing environment - Sandbox - via the Developer Portal.

Use this environment to perform all your testing, apply for accreditation, and track the status of your application review.

NOTE:
For more information about the development environment and obtaining certificates/smart cards for testing, see Development Environment.

Production environment

However, after successful accreditation, your product will be used in the Production environment(s) of the countries for which it is accredited.

Be mindful that different countries have different system names, and every environment has its own URL. In addition, Production environments might use different tax rates than the development environment. However, the logic behind applying tax rates is the same in all environments, regardless of their values.

System name by country/state:

- Republic of Srpska - `SUF`
- Serbia - `SUF`
- Fiji - `VMS`
- Samoa – `TIMS`
- Washington – `TaxCore`

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

Useful endpoints per EFD component and environment

Please refer to the below list of useful links for different EFD vendors and environments:

NOTE:
Client certificate authentication to web services, such as V-SDC or PKI, requires bypassing the proxy.

For POS products

The sections below offer two useful lists of endpoints, depending on whether your POS works only with V-SDC or with both V-SDC and E-SDC.

POS that works only with V-SDC

- **Development environment**
    - `Sandbox.taxcore.online` (Samoa)
        - V-SDC API - https://vsdc.sandbox.taxcore.online
    - `Sandbox.SUF` (Republic of Srpska)
        - V-SDC API - https://vsdc.sandbox.suf.poreskaupravars.org
    - `Sandbox.SUF` (Serbia)

        - V-SDC API - https://vsdc.sandbox.suf.purs.gov.rs
    - `Sandbox.VMS` (Fiji)
        - V-SDC API - https://vsdc.sandbox.vms.frcs.org.fj

- **Production environment**

    - `SUF` (Serbia)
        - V-SDC API - https://vsdc.suf.purs.gov.rs/
        - PKI - http://pki.suf.purs.gov.rs/
    - `SUF` (Republic of Srpska)
        - V-SDC API - https://vsdc.suf.poreskaupravars.org/
        - PKI - http://pki.suf.poreskaupravars.org/
    - `VMS` (Fiji)
        - VSDC API - https://vsdc.vms.frcs.org.fj
        - PKI - https://pki.vms.frcs.org.fj/pki
    - `TIMS` (Samoa)
        - VSDC API - https://vsdc.tims.revenue.gov.ws
        - PKI - https://pki.tims.revenue.gov.rs
    - `TaxCore` (Washington)
        - VSDC API - https://vsdc.wa.us.taxcore.online

NOTE:
URLs for the V-SDC API service can be obtained from certificates - for both the Development and the Production environemnts.

POS that works with both V-SDC and E-SDC

- **Development environment**

    - `Sandbox.SUF` (Serbia)
        - V-SDC API - https://vsdc.sandbox.suf.purs.gov.rs/Swagger
        - Secure access URL for DevSDC - (example: http://devesdc.sandbox.suf.purs.gov.rs:8888/59a718bb-00fa-44e8-b4f3-6e2987a1e53b/api) - must be obtained via the Developer Portal
    - `Sandbox.SUF` (Republic of Srpska)
        - V-SDC API - https://vsdc.sandbox.suf.poreskaupravars.org/Swagger
        - Secure access URL for DevSDC - (example: http://devesdc.sandbox.suf.poreskaupravars.org:8888/59a718bb-00fa-44e8-b4f3-6e2987a1e53b/api) - must be obtained via the Developer Portal
    - `Sandbox.taxcore.online` (Samoa)
        - V-SDC API - https://vsdc.sandbox.taxcore.online/Swagger -
        - Secure access URL for DevSDC - (example: http://devesdc.sandbox.taxcore.online:8888/59a718bb-00fa-44e8-b4f3-6e2987a1e53b/api) - must be obtained via the Developer Portal
    - `Sandbox.VMS` (Fiji)
        - V-SDC API - https://vsdc.sandbox.vms.frcs.org.fj/Swagger -
        - Secure access URL for DevSDC - (example:

http://devesdc.sandbox.vms.frcs.org.fj:8888/59a718bb-00fa-44e8-b4f3-6e2987a1e53b/api) - must be obtained via the Developer Portal

- **Production environment**
  - `SUF` (Serbia)
    - V-SDC API - https://vsdc.suf.purs.gov.rs/
    - PKI - http://pki.suf.purs.gov.rs/
  - `SUF` (Republic of Srpska)
    - V-SDC API - https://vsdc.suf.poreskaupravars.org/
    - PKI - http://pki.suf.poreskaupravars.org/
  - `VMS` (Fiji)
    - V-SDC API - https://vsdc.vms.frcs.org.fj
    - PKI - https://pki.vms.frcs.org.fj/pki
  - `TIMS` (Samoa)
    - V-SDC API - https://vsdc.tims.revenue.gov.ws
    - PKI - https://pki.tims.revenue.gov.rs
  - `TaxCore` (Washington)
    - V-SDC API - https://vsdc.wa.us.taxcore.online

NOTE:
In the Production environment, connection to an E-SDC is established by providing the E-SDC's IP address, according to the E-SDC manufacturer's instructions.

For E-SDC products

- **Development environment**
  - `Sandbox.taxcore.online` (Samoa)
    - TaxCore API - https://api.sandbox.taxcore.online
    - Get environment configuration - https://api.sandbox.taxcore.online/api/Configuration/GetConfiguration - does not require presenting your authorization certificate
  - `Sandbox.SUF` (Serbia)
    - TaxCore API - https://api.sandbox.suf.purs.gov.rs
    - Get environment configuration - https://api.sandbox.suf.purs.gov.rs/api/v3/configuration - does not require presenting your authorization certificate
  - `Sandbox.SUF` (Republic of Srpska)
    - TaxCore API - https://api.sandbox.suf.poreskaupravars.org
    - Get environment configuration -

https://api.sandbox.suf.poreskaupravars.org/api/v3/configuration - does not require presenting your authorization certificate

- `Sandbox.VMS` (Fiji)
  - TaxCore API - https://api.sandbox.vms.frcs.org.fj
  - Get environment configuration - https://api.sandbox.vms.frcs.org.fj/api/v3/configuration - does not require presenting your authorization certificate

- **Production**
  - `SUF` (Serbia)
    - TaxCore API - https://api.suf.purs.gov.rs/
    - Get environment configuration - https://api.suf.purs.gov.rs/api/v3/configuration
    - PKI - http://pki.suf.purs.gov.rs/
  - `SUF` (Republic of Srpska)
    - TaxCore API - https://api.suf.poreskaupravars.org/
    - Get environment configuration - https://api.suf.poreskaupravars.org/api/v3/configuration
    - PKI - http://pki.suf.poreskaupravars.org/
  - `VMS` (Fiji)
    - TaxCore API - https://api.vms.frcs.org.fj/
    - Get environment configuration - https://api.vms.frcs.org.fj/api/Configuration/GetConfiguration
    - PKI - https://pki.vms.frcs.org.fj/pki
  - `TIMS` (Samoa)
    - TaxCore API - https://api.tims.revenue.gov.ws/
    - Get environment configuration - https://api.tims.revenue.gov.ws/api/Configuration/GetConfiguration
    - PKI - https://pki.tims.revenue.gov.rs
  - `TaxCore` (Washington)
    - TaxCore API - https://api.wa.us.taxcore.online/

NOTE:
URLs for the TaxCore API service can be obtained from certificates in runtime, using Export Certificate APDU command to get TaxCore certificate in DER format for the Develpment and the Production environments.

List of Accreditation Authorities per Jurisdiction

Introduction

This section lists all environments and related resources for POS and E-SDC developers to use for technical review, administrative review, and accreditation for specific environments.

Accreditation authorities per jurisdiction

Accreditation authorities are responsible for **both** approving the administrative part of the application **and** for giving the final accreditation for a POS or E-SDC product.

Jurisdiction	Authority	Sandbox Environment
Fiji	Fiji Revenue and Customs Service	https://tap.sandbox.vms.frcs.org.fj
Samoa	Ministry of Customs and Revenue	https://tap.sandbox.taxcore.online
Serbia	Ministry of Finance of The Republic of Serbia - Tax Administration	https://tap.sandbox.suf.purs.gov.rs
Washington	Data Tech International	https://tap.sandbox.taxcore.online
Republic of Srpska	Ministry of Finance of The Republic of Srpska - Tax Administration	https://tap.sandbox.suf.poreskaupravars.(
Vanuatu	Department of Customs and Inland Revenue	https://tap.sandbox.taxcore.online
Papua New Guinea	The Internal Revenue Commission	https://tap.sandbox.taxcore.online

Technical review authorities per jurisdiction

Technical review authorities are responsible for approving the technical part of the application for accreditation.

Jurisdiction	Authority	Sandbox Environment
Fiji	Fiji Revenue and Customs Service	https://tap.sandbox.vms.frcs.org.fj
Samoa	Ministry of Customs and Revenue	https://tap.sandbox.taxcore.online
Serbia	Ministry of Finance of The Republic of Serbia - Tax Administration	https://tap.sandbox.suf.purs.gov.rs
Washington	Data Tech International	https://tap.sandbox.taxcore.online
Republic of Srpska	Ministry of Finance of The Republic of Srpska - Tax Administration	https://tap.sandbox.suf.poreskaupravars.(

Vanuatu	Department of Customs and Inland Revenue	https://tap.sandbox.taxcore.online
Papua New Guinea	The Internal Revenue Commission	https://tap.sandbox.taxcore.online

Data Structures

This Section contains specifications and an explanation of different data types and formats shared between E-SDCs and V-SDCs

1. UID
   UID is a Unique Identifier (8 alphanumeric characters) assigned to each Smart card and embedded in the Subject field of a digital certificate as the SERIALNUMBER parameter.

2. JSON Request Property Types
   In JSON invoice request and invoice response, values must be one of the following data types:

3. Amounts
   TaxCore works with amounts presented as decimal numbers with 4 decimal places. For the sake of simplicity, all binary-based protocols, E-SDC to Secure element - APDU commands, use integers representing the amount multiplied by 10,000.

4. Tax Rates
   E-SDC receives tax groups via *Set Tax Rates Command* on the E-SDC Initialization process, or as an information if taxes have been changed. If more than one tax group takes effect from the same moment, the group with greater groupId must be used.

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

6. Unique Identification of a Fiscal Invoice
   A fiscal invoice is uniquely identified by Invoice number - the combination of the invoice ordinal number and the Secure Element identification number (UID). Invoice number is defined in the following format:

7. Anatomy of a 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 detailed explanation below. POS is free to print any content (coupons, logos, etc.) before the beginning and after the ending mark of

the fiscal invoice.

UID

UID is a Unique Identifier (8 alphanumeric characters) assigned to each Smart card and embedded in the Subject field of a digital certificate as the SERIALNUMBER parameter.

UID can be obtained from the Secure Element Applet using the *Export Certificate* APDU command.

JSON Request Property Types

In JSON invoice request and invoice response, values must be one of the following data types:

- a string
- a number
- an object (JSON object)
- an array
- a boolean
- null

Please refer to the below document:

https://tools.ietf.org/html/rfc7159

Examples:

- "buyerId" : ""

API will see this as an empty string and an error message (code) will be returned (*FieldTooShort = 2802* or *FieldLengthInvalid = 2803*)

- "totalAmount" : "10.25"

API will see this as a string and an error message (code) will be returned (*FieldValueInvalid = 2805*)

Amounts

TaxCore works with amounts presented as decimal numbers with 4 decimal places. For the sake of simplicity, all binary-based protocols, E-SDC to Secure element - APDU commands, use integers representing the amount

multiplied by 10,000.

For example, the amount of 123,4567 will be converted into 1234567.

JSON based protocols use the standard decimal format.

Tax Rates

Tax rate: a sales tax expressed as a percentage on the sale of goods and services (for a proportional tax), or fixed tax amount, as imposed by the government. One tax rate is uniquely identified by the tax label – a string (usually only one letter) which represents one tax rate. One label is related to one category, and it will never reappear within another category.

Tax category: one or more tax rates are grouped into a tax category – a group of different rates for the same tax type (e.g. VAT, Consumption, ECAL, etc.). If a tax rate (all rates under a category) is applied on the net price, then field *Type* for a category shall be set to 0 (proportional tax). If a tax rate (all rates under a category) is applied on the total amount (with other taxes included), the field *Type* for a category shall be set to 1 (proportional tax-on-total). If a tax (all rates under a category) is a fixed tax amount, the field *Type* for a category shall be set to 2 (amount-per-quantity).

Tax group: one or more tax categories are grouped into a tax group – a set of all taxes imposed by the government applied on any points of sale within a period of time. A tax group consists of a groupId, which represents tax revision, and a date/time which defines the moment when the new taxes shall take effect.

E-SDC receives tax groups via *Set Tax Rates Command* on the E-SDC Initialization process, or as an information if taxes have been changed. If more than one tax group takes effect from the same moment, the group with greater groupId must be used.

Model and Example

TaxRateGroup

Field	Type	Description
validFrom	Date time	Date when a tax rates group shall enter into force
groupId	32 bit integer	Revision number for all taxes under a tax rates group
taxCategories	Array of TaxCategory	All tax categories under one tax rates group

TaxCategory

Field	Type	Description
name	Unicode String	Name of a tax (tax category)
type	32 bit integer	One of the following tax category types: 0 (tax-on-net) - all tax rates from this category are proportional, and shall be applied on the net price; 1 (tax-on-total) - all tax rates from this category are proportional, and shall be applied on the total amount; 2 (amount-per-quantity) - all tax rates from this category are fixed tax amounts, which shall be multiplied with item quantity.
orderId	32 bit integer	Order number for a tax category. It uniquely identifies the tax. It is related ¬to the category name meaning even if the name changes for the tax category, it's orderId will remain the same, pointing to the same tax. It is crucial for Sign Invoice APDU command
raxRates	Array of TaxRate	All tax rates for a tax (category)

TaxRate

Field	Type	Description
label	Unicode String	Label for a tax rate, unique within a tax group, always belongs to one tax category
rate	Decimal	Rate percentage for a proportional tax, or tax amount for a fixed tax.

Example


```
{
  "validFrom": "2020-08-03T12:56:56",
  "groupId": 2,
  "taxCategories": [
□{
□  "name": "ECAL",
□  "categoryType": 0,
□  "taxRates": [
□□{
□□  "rate": 5,
□□  "label": "D"
□□},
□□{
□□  "rate": 11,
□□  "label": "F"
□□}
□  ],
□  "orderId": 1
□},
□{
□  "name": "N-TAX",
```

```
    "categoryType": 0,
    "taxRates": [
      {
        "rate": 0,
        "label": "N"
      }
    ],
    "orderId": 2
  },
  {
    "name": "PBL",
    "categoryType": 2,
    "taxRates": [
      {
        "rate": 0.5,
        "label": "P"
      }
    ],
    "orderId": 3
  },
  {
    "name": "STT",
    "categoryType": 0,
    "taxRates": [
      {
        "rate": 6,
        "label": "E"
      }
    ],
    "orderId": 4
  },
  {
    "name": "VAT",
    "categoryType": 0,
    "taxRates": [
      {
        "rate": 9,
        "label": "A"
      },
      {
        "rate": 0,
        "label": "B"
      }
    ],
    "orderId": 5
  },
  {
    "name": "VAT-EXCL",
    "categoryType": 1,
    "taxRates": [
      {
        "rate": 0,
        "label": "C"
      }
    ],
    "orderId": 6
  }
 ]
}
```


Tax Amounts

It is essential to note, that **POS never uses other taxes except the ones received from an SDC**. POS displays the total prices and only the tax values received from an 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 x (**Rate/100**))	For each item with this label.
Tax on Total	**Tax Amount** = ∑(Total price x (**Rate/100**))	For each item with this label (Total price here is item base price with all other regular taxes included).
Tax on Quantity	**Tax Amount** = ∑(Fixed amount x quantity)	For each item with this label, item quantity is multiplied with fixed amount as defined by tax authority. If other tax labels are defined on the item, those taxes are calculated on the remainder.

Unique Identification of a Fiscal Invoice

A fiscal invoice is uniquely identified by Invoice number - the combination of the invoice ordinal number and the Secure Element identification number (UID). Invoice number is defined in the following format:

UID1-UID2-Ordinal_Number

Where:

- **UID1** and **UID2** are identical for the invoice issued by E-SDC and
- **Ordinal_Number** is a number generated by the Secure Element, after each invoice signing, representing a total count of invoices signed by that Secure Element.

Anatomy of a 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 detailed explanation below. POS is free to print any content (coupons, logos, etc.) before the beginning and after the ending mark of the fiscal invoice.

Elements of a fiscal receipt

Title line – marks the beginning of the fiscal part of a receipt

```
============ FISCAL INVOICE ============
```

Header data is provided by E-SDC during fiscalization of the invoice and returned to POS as part of the InvoiceFiscalizationResult object (explained in section Create Invoice). Values are extracted from the subject field of the digital certificate stored in the Secure Element Applet.

```
              502579006
               Golf V
              Sun Store
             7 Someplace
                Suva
```

Cashier identification is mandatory **only** in tax jurisdictions where local regulations mandate POS to send particular data, such as Employee ID or some other information, that uniquely identifies the POS cashier. Otherwise, it is optional information.

```
Cashier TIN:                   1234567890
```

Buyer TIN is mandatory **only** if there is a legal obligation for buyer identification. **Buyer Cost Center** is mandatory **only** if there is a legal obligation for invoice purpose identification. Make sure you are familiar with the legal requirements in each tax jurisdiction.

POS number and **POS time** are optional fields. However, tax legislation in certain jurisdictions might mandate including POS number and POS time on invoices - so make sure you become familiarized with the requirements in each jurisdiction.

```
Buyer TIN:                     5123456789
Buyer Cost Centre:                    123
POS number:                   POS2017/998
POS time:            15/6/2017 8:56:23AM
```

Reference Number is always mandatory for Refund or Copy transactions, and some Normal Sale and Advance Sale transactions. **Ref No** is printed on the receipt, containing the **SDC Invoice No** of the document which is being referenced in the format of *RequestedBy-SignedBy-Integer*.

For more information about mandatory and optional reference combinations, see Reference Number.

Optional reference may be mandatory if requested by a Tax Authority (obligation may apply to certain business activities).

All Reference Numbers are created for connected transactions, and if a **Ref No** is sent to an SDC, it must be displayed on the receipt journal.

If a Copy or Refund is issued for the transaction that was recorded before the introduction of fiscalization, POS should send XXXXXXXX-XXXXXXXX-1 as the value of **Ref No** field.

```
Ref no:          P22VC8VR-JTJC5V65-114906
```

Reference Time is always optional information. It contains the **SDC Time** of the document (invoice) that is being referenced. For more information about reference combinations, see Reference Time.

```
Ref DT:              03/09/2021 21:45:48
```

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 Invoice and Transaction Types.

```
--------------NORMAL SALE---------------
```

List of items with **gross price**, **tax labels**, **unit price** and **quantity**. Tax Labels and their validity dates are published by the tax authority and they are mandatory for each item, even when the price is 0.00.

When applying discounts, **Unit price** of the line item displayed on the journal (Price column) shows the discounted price, after **all** discounts have been applied. The displayed total price for an item is also calculated considering the unit price with the discount already applied.

```
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
```

Total Purchase, **Tax items** and **Total Tax** are calculated by 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. The taxpayer's tax liability is based on these tax amounts, calculated by E-SDC. The calculation is explained in the section Calculate Taxes. In case of Refund receipts, **Total Refunded** should be printed instead of **Total Purchase**.

```
---------------------------------------
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
```

Fiscal metadata is added to the invoice through fiscalization. **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.

```
=======================================
```

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

QR Code contains Invoice verification URL. QR Code also contains Internal data and the digital signature used for invoice verification. The invoice is verifiable by the customer immediately after fiscalization. In case an invoice/receipt is delivered as an electronic document (email), the QR Code shall be substituted with a verification URL in (clickable) hyperlink format.

NOTE:
This is just a sample QR code image, not an actual URL.

Title line – marks the end of the fiscal part of a receipt

```
======= END OF FISCAL INVOICE ========
```

Custom message returned from E-SDC

```
 This is a custom message.
```

Key Elements

A fiscal invoice must contain the following parts (it may also contain additional data if it is required by a specific industry):

Invoice Request

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 the standard, publicly available protocol for communication to E-SDC and the preferred technology of the POS system.

Invoice Response

Invoice Response is generated by 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.

Signature

A digital signature applied to the content of an electronic invoice by the Secure Element.

Internal Data

Internal data contains encrypted fiscal data. The content of the internal data is readable by the tax authority system only.

Verification URL

URL of verification service is used to verify the authenticity of the particular fiscal invoice for customer convenience. It shall be represented as a QR code on a printed receipt or as a hyperlink in an electronic document (e.g. an email).

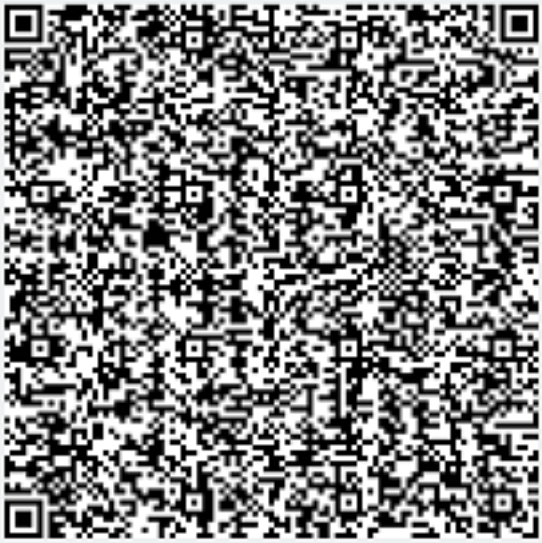
Normal Refund Receipt

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 Refunded. Tax Items are displayed as positive values.

For Refund transaction type, the **Ref no** (reference to the original Normal Sale) element is mandatory, and in some cases the same applies to the **Ref DT** (SDC Time (server time zone) of the original referenced document). Make sure you are familiar with the local legal requirements.

Example:

```
============ FISCAL INVOICE ============
TIN:                            RS3434343434
Company:                        Popravke doo
Store:                          Popravke doo
Address:                  Sime Vlahovica 65
District:                             Lestane
Cashier TIN:                             QA
POS Number:                     145345-2021
POS Time:              14.07.2021 15:59:33
Ref No:               FQ38SHWP-FQ38SHWP-73
Ref DT:                14.07.2021 15:59:33
-------------NORMAL REFUND--------------
                   Items
========================================
Name    Price         Qty.           Total
Hleb (A)
        10,00           2           -20,00
----------------------------------------
Total Purchase:                      20,00
----------------------------------------
Mobile money:                        20,00
========================================
Label        Name      Rate            Tax
A             VAT    9,00%           1,65
----------------------------------------
Total Tax:                           1,65
========================================
SDC Time:              14.07.2021 15:59:33
SDC Invoice No:      FQ38SHWP-FQ38SHWP-75
Invoice Counter:                    8/75NR
========================================
```



```
======== 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.

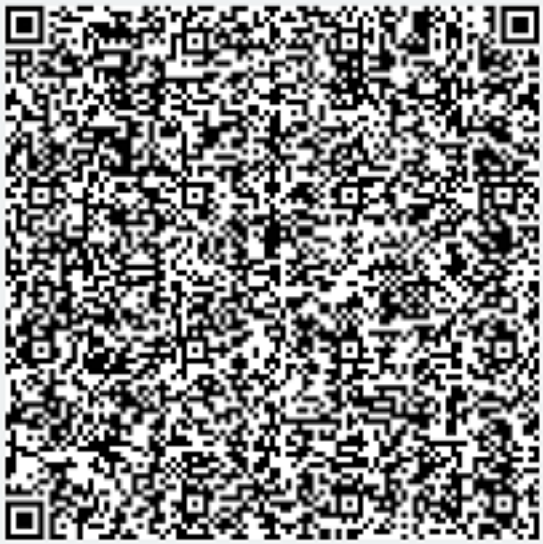
Receipt must also contain **THIS IS NOT A FISCAL INVOICE** below the total amount payable. Font size is at least twice the size of the text on the receipt that specifies the total amount payable.

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

Copy type invoices usually require the use of the **Ref no** (reference to the original Normal Sale) element. Make sure you are familiar with the local legal requirements.

Example:

```
===== THIS IS NOT A FISCAL RECEIPT =====
TIN:                          RS3434343434
Company:                      Popravke doo
Store:                        Popravke doo
Address:                 Sime Vlahovica 65
District:                          Lestane
Cashier TIN:                           QA
POS Number:                   145345-2021
POS Time:             14.07.2021 15:59:33
Ref No:               FQ38SHWP-FQ38SHWP-73
Ref DT:               14.07.2021 15:59:33
---------------COPY SALE----------------
                  Items
========================================
Name    Price         Qty.           Total
Hleb (A)
        10,00           2            20,00
----------------------------------------
Total Purchase:                      20,00
----------------------------------------
Mobile money:                        20,00
========================================
      THIS IS NOT A FISCAL INVOICE
========================================
Label        Name     Rate             Tax
A             VAT    9,00%           1,65
----------------------------------------
Total Tax:                           1,65
========================================
SDC Time:             14.07.2021 15:59:33
SDC Invoice No:     FQ38SHWP-FQ38SHWP-74
Invoice Counter:                   8/74CS
========================================
```



```
===== THIS IS NOT A FISCAL RECEIPT =====
```

Protocols

This section contains the specification of protocols used by both E-SDCs and POSes

1. TaxCore.Api
   TaxCore.API is designed to expose its functionalities to all external systems.

2. POS to SDC Protocol
   This section describes API exposed by V-SDC and all accredited E-SDC devices and applications to POS and invoicing systems.

TaxCore.Api

Introduction

TaxCore.API is designed to expose its functionalities to all external systems.

It enables integration of POSes, E-SDCs, accreditation tools and other clients that makes TaxCore ecosystem.

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

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.

Content

1. Obtain a URL of the TaxCore.API Service from Digital Certificate
   TaxCore.API service provides the configuration information of the current environment. In order to access it, one must obtain URL of the TaxCore.API service from Secure Element digital certificate. This procedure applies to any type of Secure Element.

2. Configuration
   Every TaxCore.API instance provides services anyone can use to gather information about current environment and all available environments from the current node.

Obtain a URL of the TaxCore.API Service from Digital Certificate

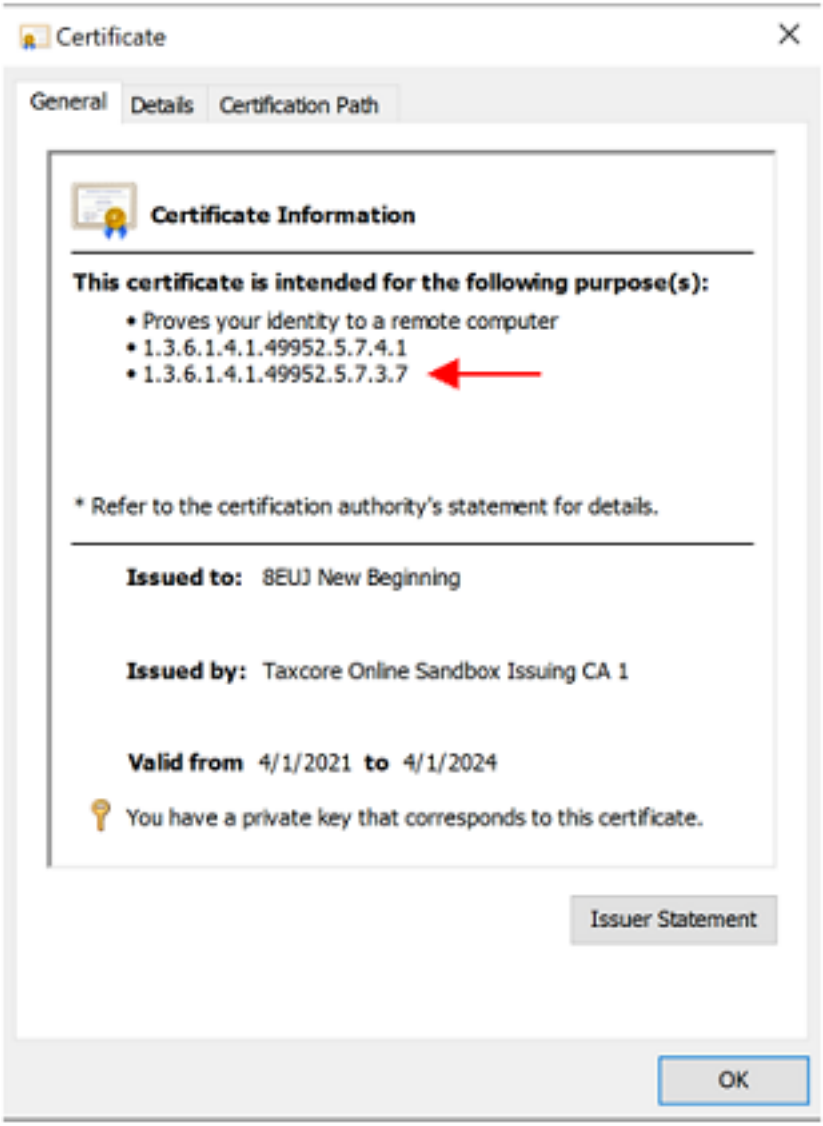
Introduction

TaxCore.API service provides the configuration information of the current environment. In order to access it, one must obtain URL of the TaxCore.API service from Secure Element digital certificate. This procedure applies to any type of Secure Element.

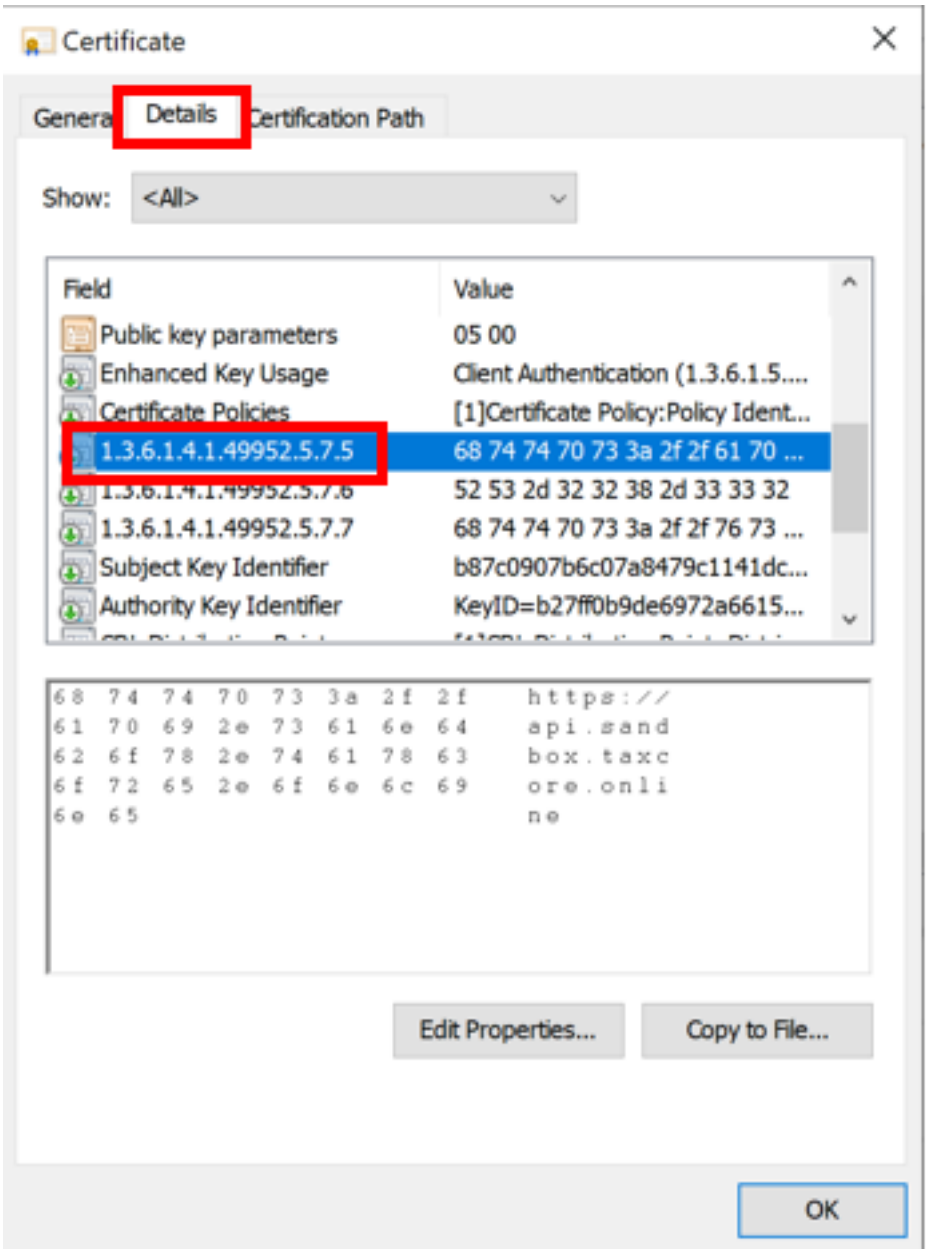
How to obtain TaxCore.API URL

TaxCore.API URL is stored in a digital certificate of Secure Element as the value of OID. OID is dynamically created when Secure Element is issued and depends on the target environment. For example, Test and Production environments will have different OIDs, so that digital certificate issued for one environment cannot be used for any purpose in another environment. Follow the steps to read the OID value from the digital certificate installed on your computer:

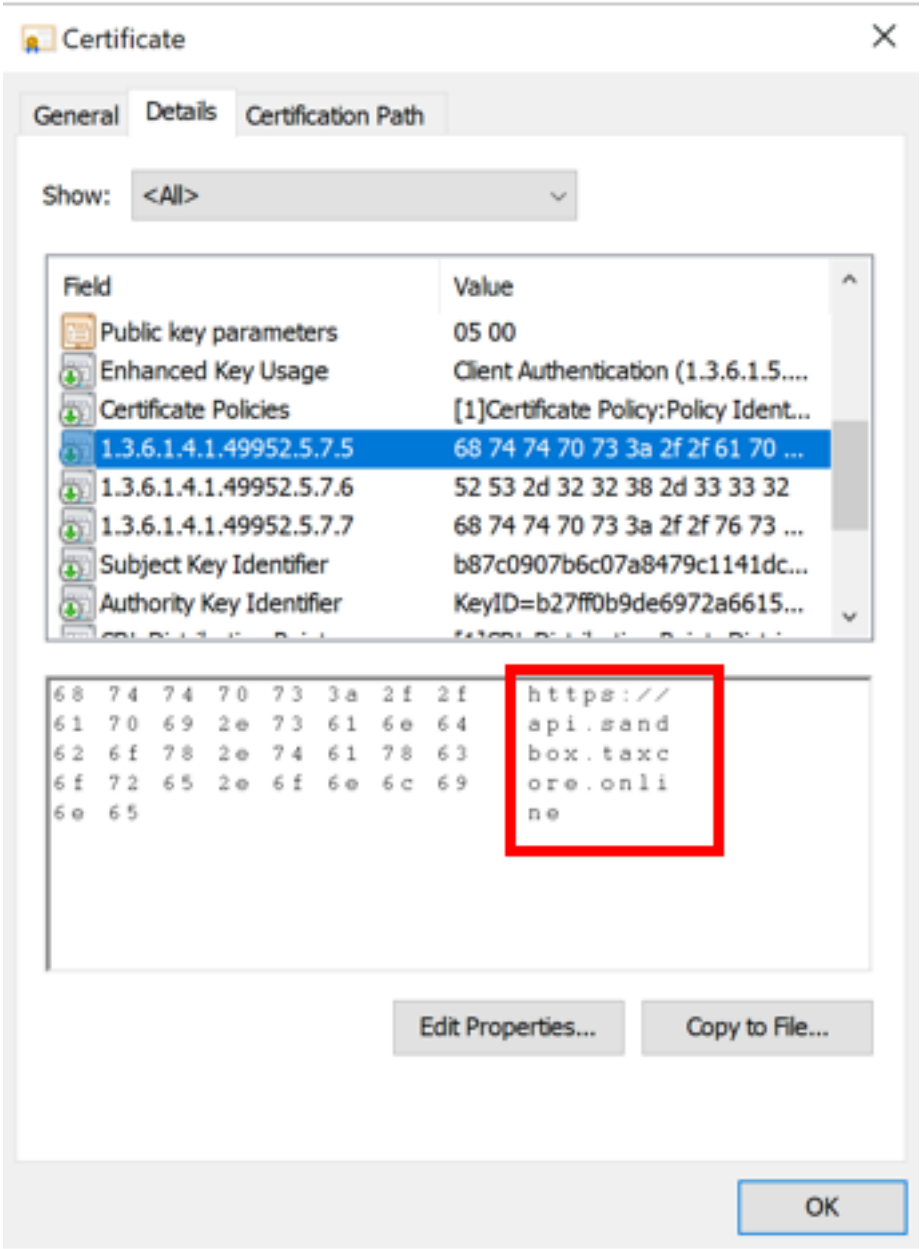
1. Open the certificate installed on your computer
2. In the **General** tab, your certificate should have OID in this format **1.3.6.1.4.1.49952.X.Y.3.7**
3. **X and Y parameters identify the environment** and their **values change according to each environment**.



4. Click on the **Details** tab and find the line with an OID in this format - **1.3.6.1.4.1.49952.X.Y.5**



5. Again, **X and Y parameters identify the environment**, and the **values will vary according to different environments**, but they will be the same as on the **General** tab
6. Number **5** at the end identifies the TaxCore.API URL
7. Read the value of this OID containing root URL of TaxCore.API Service



In case Secure Element is in Smart Card format, URL of TaxCore.API services can be obtained at runtime using [Export Certificate APDU command], which returns digital certificate in DER format.

Configuration

Introduction

Every TaxCore.API instance provides services anyone can use to gather information about current environment and all available environments from the current node.

These services should be used by all POS and E-SDC implementations to gather environment information and to simplify technical review, accreditation, deployment and usage in various jurisdictions.

Content

1. Configuration
   This service returns information about environment configuration.

2. Tax Rates
   This service returns information about defined tax rates for the environment.

3. Environments
   This service returns a list of all available environments from the TaxCore.API.

4. Encryption Certificate
   This service returns the public key of the TaxCore.API which can be used for encryption.

Configuration

This service returns information about environment configuration.

NOTE:
The same parameters can also be obtained from any SDC device (E-SDC or V-SDC) which is the preferred option for obtaining this information - see POS to SDC Protocol.

Endpoint

Endpoint	Example
<TaxCore_API_URL_obtained_from_certificate_as_e: here>/api/v3/configuration	`https://api.sandbox.suf.purs.gov.rs/api/v`

NOTE:
Development and production environments, as well the environments in different countries, have different URLs. For this reason, URLs and names in your documentation, code and UI should not be hardcoded but configurable or extracted from a digital certificate.

HTTP Method

GET

Authentication

No authentication

Response

Field	Description
organizationName	Name of the tax authority organization
serverTimeZone	Time zone of the server location
street	Tax authority street address
city	Tax authority city
country	Tax authority country
endpoints	List of available endpoints
environmentName	Name of the environment
logo	Link to the tax authority's official logo
ntpServer	NTP server used for time synchronization
supportedLanguages	List of language-culture strings supported by TaxCore

Example


```
{
  "organizationName": "Data Tech International",
  "serverTimeZone": "Fiji Standard Time",
  "street": "Kruzni put 7",
  "city": "Belgrade",
  "country": "RS",
  "endpoints": {
    "taxpayerAdminPortal": "https://tap.sandbox.taxcore.dti.rs:443/",
    "taxCoreApi": "https://api.sandbox.taxcore.dti.rs:443/",
    "vsdc": "https://vsdc.sandbox.taxcore.dti.rs:443/",
    "root": "https://frontendui.sandbox.taxcore.dti.rs:443/v/?vl="
  },
  "environmentName": "SANDBOX",
  "logo": "https://i.imgur.com:443/tg5ad6O.png?hash=59421698",
  "ntpServer": "http://0.europe.pool.ntp.org:80/",
  "supportedLanguages": [
```

```
    "en-US",
    "sr-Cyrl-RS"
  ]
}
```

Tax Rates

This service returns information about defined tax rates for the environment.

NOTE:
The same parameters can also be obtained from any SDC device (E-SDC or V-SDC) which is the preferred option for obtaining this information - see POS to SDC Protocol.

Endpoint

Endpoint	Example
<TaxCore_API_URL_obtained_from_certificate_as_e: here>/api/v3/tax-rates	`https://api.sandbox.suf.purs.gov.rs/api rates`

NOTE:
Development and production environments, as well the environments in different countries, have different URLs. For this reason, URLs and names in your documentation, code and UI should not be hardcoded but configurable or extracted from a digital certificate.

HTTP Method

`GET`

Authentication

No authentication

Response

Field	Description
currentTaxRates	The currently active tax rate group.

allTaxRates	All tax rate groups ever used for the environment - can be used for reference when issuing refunds. Also includes all approved future tax rate groups, with the date of their activation.

Example


```
{
  "currentTaxRates": [
    {
      "validFrom": "2020-08-03T12:56:56",
      "groupId": 2,
      "taxCategories": [
        {
          "name": "ECAL",
          "categoryType": 0,
          "taxRates": [
            {
              "rate": 5,
              "label": "D"
            },
            {
              "rate": 11,
              "label": "F"
            }
          ],
          "orderId": 1
        },
        {
          "name": "N-TAX",
          "categoryType": 0,
          "taxRates": [
            {
              "rate": 0,
              "label": "N"
            }
          ],
          "orderId": 2
        },
        {
          "name": "PBL",
          "categoryType": 2,
          "taxRates": [
            {
              "rate": 0.5,
              "label": "P"
            }
          ],
          "orderId": 3
        },
        {
          "name": "STT",
          "categoryType": 0,
          "taxRates": [
            {
              "rate": 6,
              "label": "E"
            }
          ],
          "orderId": 4
        },
        {
          "name": "VAT",
          "categoryType": 0,
```

```
        "taxRates": [
          {
            "rate": 9,
            "label": "A"
          },
          {
            "rate": 0,
            "label": "B"
          }
        ],
        "orderId": 5
      },
      {
        "name": "VAT-EXCL",
        "categoryType": 1,
        "taxRates": [
          {
            "rate": 0,
            "label": "C"
          }
        ],
        "orderId": 6
      }
    ]
  }
],
"allTaxRates": [
  {
    "validFrom": "2020-08-02T21:04:00",
    "groupId": 1,
    "taxCategories": [
      {
        "name": "ECAL",
        "categoryType": 0,
        "taxRates": [
          {
            "rate": 5,
            "label": "D"
          },
          {
            "rate": 11,
            "label": "F"
          }
        ],
        "orderId": 1
      },
      {
        "name": "N-TAX",
        "categoryType": 0,
        "taxRates": [
          {
            "rate": 0,
            "label": "N"
          }
        ],
        "orderId": 2
      },
      {
        "name": "PBL",
        "categoryType": 2,
        "taxRates": [
          {
            "rate": 0.2,
            "label": "P"
          }
        ],
```

```
      "orderId": 3
    },
    {
      "name": "STT",
      "categoryType": 0,
      "taxRates": [
        {
          "rate": 6,
          "label": "E"
        }
      ],
      "orderId": 4
    },
    {
      "name": "VAT",
      "categoryType": 0,
      "taxRates": [
        {
          "rate": 9,
          "label": "A"
        },
        {
          "rate": 0,
          "label": "B"
        }
      ],
      "orderId": 5
    },
    {
      "name": "VAT-EXCL",
      "categoryType": 1,
      "taxRates": [
        {
          "rate": 0,
          "label": "C"
        }
      ],
      "orderId": 6
    }
  ]
},
{
  "validFrom": "2022-01-01T00:00:00Z",
  "groupId": 3,
  "taxCategories": [
    {
      "name": "ECAL",
      "categoryType": 0,
      "taxRates": [
        {
          "rate": 10,
          "label": "F"
        }
      ],
      "orderId": 5
    },
    {
      "name": "PB",
      "categoryType": 2,
      "taxRates": [
        {
          "rate": 0.1,
          "label": "P"
        }
      ],
      "orderId": 6
```

```
        },
        {
          "name": "STT",
          "categoryType": 0,
          "taxRates": [
            {
              "rate": 1,
              "label": "E"
            }
          ],
          "orderId": 2
        },
        {
          "name": "VAT",
          "categoryType": 0,
          "taxRates": [
            {
              "rate": 9,
              "label": "A"
            }
          ],
          "orderId": 1
        }
      ]
    }
  ]
}
```


Environments

This service returns a list of all available environments from the TaxCore.API.

NOTE:
The same parameters can also be obtained from any SDC device (E-SDC or V-SDC) which is the preferred option for obtaining this information - see POS to SDC Protocol.

Endpoint

Endpoint	Example
<TaxCore_API_URL_obtained_from_certificate_as_e: here>/api/v3/environments	`https://api.sandbox.suf.purs.gov.rs/api/v`

NOTE:
Development and production environments, as well the environments in different countries, have different URLs. For this reason, URLs and names in your documentation, code and UI should not be hardcoded but configurable or extracted from a digital certificate.

HTTP Method

`GET`

Authentication

No authentication

Response data

Field	Description
name	Name of the TaxCore environment
url	Root Url of the TaxCore environment. You can construct other well-known Urls using this value. For Example, `api.sandbox.taxcore.online` and `vsdc.sandbox.taxcore.online`
description	Description of the environment. For eaxample, "for accreditation purposes only"
environmentName	For internal use only

Example (not specific for any environment)

```
[
  {
    "name": "Sandbox",
    "url": "sandbox.taxcore.online",
    "description": "",
    "environmentName": "SANDBOX"
  }
]
```

Encryption Certificate

This service returns the public key of the TaxCore.API which can be used for encryption.

Endpoint

Endpoint	Example
<TaxCore_API_URL_obtained_from_certificate_as_e:	`https://api.sandbox.suf.purs.gov.rs/api/v`

here>/api/v3/encryption-certificate	`certificate`

NOTE:
Development and production environments, as well the environments in different countries, have different URLs. For this reason, URLs and names in your documentation, code and UI should not be hardcoded but configurable or extracted from a digital certificate.

HTTP Method

`GET`

Authentication

No authentication

Response data

Response contains Base64 encoded raw data of TaxCore.API X509 public key certificate used for encryption.

Example


```
"MIIF8TCCBNmgAwIBAgIOH2Jcu+Xv1l8AAAAAJN4wDQYJKoZIhvcNAQELBQAwP
jELMAkGA1UEBhMCUlMxDDAKBgNVBAoTA0RUSTEhMB8GA1UEAxMYVGF4Q29yZSB
JQ0ExIERldmVsb3BtZW50MB4XDTIwMDYxNzA4MjAxNVoXDTIzMDYxNzA4MzAxN
VowczELMAkGA1UEBhMCUlMxDzANBgNVBAgTBlNlcmJpYTERMA8GA1UEBxMIQmV
sZ3JhZGUxDDAKBgNVBAoTA0RUSTENMAsGA1UECxMEQ29yZTERMA8GA1UEBRMIZ
mYwN2RlMGYxEDAOBgNVBAMTB1RheENvcmUwggEiMA0GCSqGSIb3DQEBAQUAA4I
BDwAwggEKAoIBAQCP6DtitC7kP3EwPwES6xjPD7Yn2S0mUzTy2vAsHmSeW4iHs
ZfSY24kzlEpUKqHJhukmg7tJwT+L4IJDaQzFdK+PxlgYAFiOGeAu+WS3Ka1dY0
2/bS+rFvzLbuh29Jhd5ESPqP39X7DSA9fSB8bms/SmaI0h5qT3NWL7leU9tiG/
77srpVhzPP4/CI+oXYOQ5V7MvztGUXjsRJBcg8o2nHJa4nDzz4kuyY5Mb4k6Nr
OTPMcTs8HgR2HE1NTIplKEA46R8Rv9QD4l5oBXmL+bEiKR68xXjoAxbBLJcEfV
CTIkFU0ImuyHK1RRV3JCSbg1fdRWcqIbkeCEYdeRE0HPMAbAgMBAAGjggK2MII
CsjAOBgNVHQ8BAf8EBAMCBDAwDwYDVR0TAQH/BAUwAwIBADAXBgNVHSUEEDAOB
gwrBgEEAYOGIAEBAwYwggGJBgNVHSAEggGAMIIBfDCBuwYMKwYBBAGDhiABAQQ
BMIGqMC4GCCsGAQUFBwIBFiJodHRwOi8vcGtpLnRlc3QudGF4Y29yZS5kdGkuc
nMvcGtpMHgGCCsGAQUFBwICMGweagBEAFQASQAgAFQAYQB4AEMAbwByAGUAIAB
pAG4AdABlAHIAbgBhAGwAVQBzAGUAIABwAEsASQBUAGUAcwB0ACAAaQBzAHMAd
QBhAG4AYwBlAFAAbwBsAGkAYwB5ACAAYwBsAGEAcwBzADEwgbsGDCsGAQQBg4Y
gAQEEATCBqjAuBggrBgEFBQcCARYiaHR0cDovL3BraS50ZXN0LnRheGNvcmUuZ
HRpLnJzL3BraTB4BggrBgEFBQcCAjBsHmoARABUAEkAIABUAGEAeABDAG8AcgB
lACAAaQBuAHQAZQByAG4AYQBsAFUAcwBlACAAcABLAEkAVABlAHMAdAAgAGkAc
wBzAHUAYQBuAGMAZQBQAG8AbABpAGMAeQAgAGMAbABhAHMAcwAxMB0GA1UdDgQ
WBBQCMogZokiv0fsLYwd+h2HFU9SBejAfBgNVHSMEGDAWgBR96opdCPQ+lAPgN
GyM3NFn5WwubjBOBgNVHR8ERzBFMEOgQaA/hj1odHRwOi8vcGtpLnRlc3QudGF
4Y29yZS5kdGkucnMvcGtpL1RheENvcmVJQ0ExRGV2ZWxvcG1lbnQuY3JsMFkGC
CsGAQUFBwEBBE0wSzBJBggrBgEFBQcwAoY9aHR0cDovL3BraS50ZXN0LnRheGN
vcmUuZHRpLnJzL3BraS9UYXhDb3JlSUNBMURldmVsb3BtZW50LmNlcjANBgkqh
kiG9w0BAQsFAAOCAQEAi9pj/BFcQ7i1i/577kcCP2/0W8hi3WrO7UyESvSn+z7
5jdBG1r1v3tkSswqYhoQmZsDifuviO2cnSEt0QOyyBTUYS499xsIbGVZVzuEop
nFzqsgytarPtITho4DeLCnjtollxVh9PPzAT1fsz7xXHqfaex6x+HCkWFkUuFE
bze59ib5MJoiz5zE9WxYZwdXy7mOxgdjyjTaNQTGReqDWeT1bA2l9/UZxHGBpr
TALfU7uAxdnCexPxim24sDkO3/ZsVLqhRFPuTFqlEY8K9YY9L9aWxIWFuJFKsh
```

rHc3fQND9BzjT5Hj6T2CxqfEUUxgUkpFPkeOYEtmvE/rm03jSvA=="

POS to SDC Protocol

This section describes API exposed by V-SDC and all accredited E-SDC devices and applications to POS and invoicing systems.

Create Invoice command is the same for both V-SDCs and E-SDC thus simplifying implementation of POSes. Some other services defined in this section are specific to either V-SDC or E-SDC implmentations because they cater for differences between Connected and Semi-Connected modes of operation and specifics of authentication and authorization.

Network Connectivity of E-SDCs

E-SDC device shall 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 shall be equipped with RJ-45 plug male connectors since an E-SDC is equipped with a female RJ-45 connector.

E-SDC shall 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 non-volatile memory during the manufacturing.

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

Content

This service is used to get status information from SDC.

4. Verify PIN
   This service is used to verify a PIN entered by a cashier on a POS. Once the PIN is entered, it is validated using the Secure element on a smart card. The request is sent as plain text and the response is plain text (still application/json HTTP header must be provided).

5. Create Invoice
   This section contains the description of the **Create Invoice** command.

6. Get Last Signed Invoice
   Get the last signed invoice by `RequestId`. It is used in case POS did not get a response from E-SDC after the Create Invoice service was invoked. The response is the same as for the Create Invoice service. `RequestId` field from the request is used to determine whether the last invoice was successfully created.

7. Error Messages Format
   This section describes the structure and format of error messages that SDC returns to POS.

8. Status and Error Codes
   All protocols share the same Info, Error and Warning codes.

Get Environment Parameters

This service returns information about environment configuration. The same parameters can also be obtained directly from TaxCore.API with more accuracy, as SDCs' accuracy depends on their synchronization frequency. The fields list in Response can grow, adding additional information, so integrators should keep that in mind.

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 environment names, and every environment has its own URL (see Environments). For this reason, URLs and names in your documentation, code and UI should not be hardcoded but configurable or extracted from a digital certificate.

SDC	Endpoint	Example
V-SDC	<V-SDC_API_URL_obtained_from_certificate_as_exp	`[[_TaxCore.PublicConfiguration.VsdcAp parameters`

	here>/api/v3/environment-parameters	
E-SDC	http://<ESDC_ip_address>:<ESDC_port>/api/parameters	`http://192.168.88.112:8888/api/v3/env parameters`

HTTP Method

`GET`

Header

Add following HTTP headers to each request

- `Accept: application/json`
- `PAC: <PAC_value>` (PAC value for the provided client authentication certificate - **only for V-SDC**)

Authentication

V-SDC

Use client digital certificate for authentication with each request to V-SDC.Api.

E-SDC

E-SDC does not require client authentication.

Response

Field	Description
organizationName	Name of the tax authority organization
serverTimeZone	Time zone of the server location
street	Tax authority street address
city	Tax authority city
country	Tax authority country
endpoints	List of available endpoints

environmentName	Name of the environment
logo	Link to the tax authority's official logo
ntpServer	NTP server used for time synchronization
supportedLanguages	Supported language(s) for the environment

Example (not specific for any environment)


```
{
  "organizationName": "Data Tech International",
  "serverTimeZone": "Fiji Standard Time",
  "street": "Kruzni put 7",
  "city": "Belgrade",
  "country": "RS",
  "endpoints": {
    "taxpayerAdminPortal": "https://tap.sandbox.taxcore.dti.rs:443/",
    "taxCoreApi": "https://api.sandbox.taxcore.dti.rs:443/",
    "vsdc": "https://vsdc.sandbox.taxcore.dti.rs:443/",
    "root": "https://frontendui.sandbox.taxcore.dti.rs:443/v/?vl="
  },
  "environmentName": "SANDBOX",
  "logo": "https://i.imgur.com:443/tg5ad6O.png?hash=59421698",
  "ntpServer": "http://0.europe.pool.ntp.org:80/",
  "supportedLanguages": [
    "en-US",
  ]
}
```


Attention

This command is used by POS to verify if E-SDC is available. The command is 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 immediately sends the next command.

Endpoint

SDC	Endpoint	Example
V-SDC	<V-SDC_API_URL_obtained_from_certificate_as_exp here>/api/v3/attention	`[[_TaxCore.PublicConfiguration.VSDCAp`
E-SDC	http://<ESDC_ip_address>:<ESDC_port>/api/	`http://192.168.88.112:8888/api/v3/att`

NOTE:
Development and production environments, as well the environments in different countries, have different URLs. For this reason, URLs and names in your documentation, code and UI should not be hardcoded but configurable or extracted from a digital certificate.

Method

`GET`

Header

Add the following HTTP headers to each request

- `Accept: application/json`

Authentication

V-SDC

V-SDC.API does not require client authentication.

E-SDC

E-SDC does not require client authentication.

Request

N/A

Response

`HTTP Code 200`

Get Status

This service is used to get status information from SDC.

Endpoint

SDC	Endpoint	Example
V-SDC	<V-SDC_API_URL_obtained_from_certificate_as_exp here>/api/v3/status	`[[_TaxCore.PublicConfiguration.VSDCAp`
E-SDC	http://<E-SDC_ip_address>:<E-SDC_port>/api/v3/status	`http://192.168.88.112:8888/api/v3/sta`

NOTE:
Development and production environments, as well the environments in different countries, have different URLs. For this reason, URLs and names in your documentation, code and UI should not be hardcoded but configurable or extracted from a digital certificate.

Method

`GET`

Header

Add following HTTP headers to each request

- `Accept: application/json`
- `PAC: <PAC_value>` (PAC value for the provided client authentication certificate - **only for V-SDC**)

Authentication

V-SDC

Use client digital certificate for authentication with each request to V-SDC.Api.

E-SDC

E-SDC does not require client authentication.

Request

N/A

Response

JSON formatted data in accordance with the below table.

SDC	Field	Description	Example
E-SDC	**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
E-SDC	**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 using *Amount Status* APDU command	False if Total Amount is 1554879 Maximum Limit is 9000000 True If Total Amount is 7504899 Maximum Limit is 10000000
E-SDC, V-SDC	**sdcDateTime**	Current **Local Date and Time** in ISO 8601 format	2017-08-30T11:53:05+13:00
E-SDC	**lastInvoiceNumber**	Invoice number of the last invoice signed by this E-SDC.	ORG674J1-ORG674J1-98637
E-SDC	**protocolVersion**	Always return 1.0.0.0	1.0.0.0
E-SDC	**secureElementVersion**	Version obtained from the Secure element using *Get Secure Element Version* APDU command	1.1.1
E-SDC	**hardwareVersion**	Manufacturer-specific hardware version, if applicable	1.2.7.21
E-SDC	**softwareVersion**	Manufacturer-specific software version	1.7.6.5
E-SDC	**deviceSerialNumber**	Manufacturer-specific serial number of the E-SDC device	0108-0012-ASDJKA1SDJL2A

		in format of the Manufacturer Registration Code - MRC	
E-SDC	**make**	Name of the manufacturer	Acme
E-SDC	**model**	Manufacturer-specific Model Name	The Device 442
E-SDC	**mssc**	Manufacturer-specific Errors, Warnings and info messages	Array of error codes
E-SDC	**gsc**	General Errors, Warnings and info messages defined in the Status and Error Codes section	Array of error codes
E-SDC, V-SDC	**supportedLanguages**	List of language-culture strings supported by SDC and intercepted with languages supported by specific environment, as per Get Environment Parameters. They define the language and data formats on the invoice journal.	en-US
E-SDC, V-SDC	**uid**	UID of the client certificate (V-SDC) or inserted smart card (E-SDC)	TK7SV2AY
E-SDC, V-SDC	**taxCoreApi**	Root URL of the TaxCore.API targeted by SDC	https://api.sandbox.taxcore.dt
E-SDC, V-SDC	**currentTaxRates**	An object of currently active tax rates	An object containing validFrom date, group Id and the array of currently active tax categories
E-SDC, V-SDC	**allTaxRates**	An array of all tax rates	An array containing validFrom date, group Id and tax categories for each group

Model

```
GetStatusResponse {
sdcDateTime (string, optional),
```

supportedLanguages (Array[string], optional),

uid (string, optional),

taxCoreApi (string, optional),

currentTaxRates (TaxRateGroup, optional),

allTaxRates (Array[TaxRateGroup], optional)
}

Example for E-SDC


```
{
"isPinRequired": true,
"auditRequired": false,
"sdcDateTime": "2021-07-14T18:21:11.3320963+02:00",
"lastInvoiceNumber": "CL4KBJCE-CL4KBJCE-1",
"protocolVersion": "1.0.0.0",
"hardwareVersion": "1.0.0.0",
"softwareVersion": "1.0.0.0",
"mssc": [
"6001"
],
"gsc": [
"0100"
],
"deviceSerialNumber": "99-3010-22222222",
"make": "ExampleSDC",
"model": "ExampleSDC",
"supportedLanguages": [
"en-US",
],
"uid": "TK7SV2AY",
"taxCoreApi": "https://api.sandbox.taxcore.dti.rs:443/",
"currentTaxRates": {
"validFrom": "2020-08-03T12:56:56",
"groupId": 2,
"taxCategories": [
{
"name": "ECAL",
"categoryType": 0,
"taxRates": [
{
"rate": 5,
"label": "D"
},
{
"rate": 11,
"label": "F"
}
],
"orderId": 1
},
{
"name": "N-TAX",
"categoryType": 0,
"taxRates": [
{
"rate": 0,
"label": "N"
}
],
```

```
□□□□"orderId": 2
□□□},
□□□{
□□□□"name": "PBL",
□□□□"categoryType": 2,
□□□□"taxRates": [
□□□□□{
□□□□□□"rate": 0.5,
□□□□□□"label": "P"
□□□□□}
□□□□],
□□□□"orderId": 3
□□□},
□□□{
□□□□"name": "STT",
□□□□"categoryType": 0,
□□□□"taxRates": [
□□□□□{
□□□□□□"rate": 6,
□□□□□□"label": "E"
□□□□□}
□□□□],
□□□□"orderId": 4
□□□},
□□□{
□□□□"name": "VAT",
□□□□"categoryType": 0,
□□□□"taxRates": [
□□□□□{
□□□□□□"rate": 9,
□□□□□□"label": "A"
□□□□□},
□□□□□{
□□□□□□"rate": 0,
□□□□□□"label": "B"
□□□□□}
□□□□],
□□□□"orderId": 5
□□□},
□□□{
□□□□"name": "VAT-EXCL",
□□□□"categoryType": 1,
□□□□"taxRates": [
□□□□□{
□□□□□□"rate": 0,
□□□□□□"label": "C"
□□□□□}
□□□□],
□□□□"orderId": 6
□□□}
□□]
□},
□"allTaxRates": [
□□{
□□□"validFrom": "2020-08-02T21:04:00",
□□□"groupId": 1,
□□□"taxCategories": [
□□□□{
□□□□□"name": "ECAL",
□□□□□"categoryType": 0,
□□□□□"taxRates": [
□□□□□□{
□□□□□□□"rate": 5,
□□□□□□□"label": "D"
□□□□□□},
□□□□□□{
```

```
        "rate": 11,
        "label": "F"
      }
    ],
    "orderId": 1
  },
  {
    "name": "N-TAX",
    "categoryType": 0,
    "taxRates": [
      {
        "rate": 0,
        "label": "N"
      }
    ],
    "orderId": 2
  },
  {
    "name": "PBL",
    "categoryType": 2,
    "taxRates": [
      {
        "rate": 0.2,
        "label": "P"
      }
    ],
    "orderId": 3
  },
  {
    "name": "STT",
    "categoryType": 0,
    "taxRates": [
      {
        "rate": 6,
        "label": "E"
      }
    ],
    "orderId": 4
  },
  {
    "name": "VAT",
    "categoryType": 0,
    "taxRates": [
      {
        "rate": 9,
        "label": "A"
      },
      {
        "rate": 0,
        "label": "B"
      }
    ],
    "orderId": 5
  },
  {
    "name": "VAT-EXCL",
    "categoryType": 1,
    "taxRates": [
      {
        "rate": 0,
        "label": "C"
      }
    ],
    "orderId": 6
  }
]
```

```
  },
  {
    "validFrom": "2022-01-01T00:00:00Z",
    "groupId": 3,
    "taxCategories": [
      {
        "name": "ECAL",
        "categoryType": 0,
        "taxRates": [
          {
            "rate": 10,
            "label": "F"
          }
        ],
        "orderId": 5
      },
      {
        "name": "PB",
        "categoryType": 2,
        "taxRates": [
          {
            "rate": 0.1,
            "label": "P"
          }
        ],
        "orderId": 6
      },
      {
        "name": "STT",
        "categoryType": 0,
        "taxRates": [
          {
            "rate": 1,
            "label": "E"
          }
        ],
        "orderId": 2
      },
      {
        "name": "VAT",
        "categoryType": 0,
        "taxRates": [
          {
            "rate": 9,
            "label": "A"
          }
        ],
        "orderId": 1
      }
    ]
  }
]
}
```


Example for V-SDC


```
{
  "sdcDateTime": "2021-07-14T18:21:11.3320963+02:00",
  "supportedLanguages": [
    "en-US",
    "sr-Cyrl-RS"
  ],
  "uid": "TK7SV2AY",
  "taxCoreApi": "https://api.sandbox.taxcore.dti.rs:443/",
```

```
"currentTaxRates": {
"validFrom": "2020-08-03T12:56:56",
"groupId": 2,
"taxCategories": [
{
"name": "ECAL",
"categoryType": 0,
"taxRates": [
{
"rate": 5,
"label": "D"
},
{
"rate": 11,
"label": "F"
}
],
"orderId": 1
},
{
"name": "N-TAX",
"categoryType": 0,
"taxRates": [
{
"rate": 0,
"label": "N"
}
],
"orderId": 2
},
{
"name": "PBL",
"categoryType": 2,
"taxRates": [
{
"rate": 0.5,
"label": "P"
}
],
"orderId": 3
},
{
"name": "STT",
"categoryType": 0,
"taxRates": [
{
"rate": 6,
"label": "E"
}
],
"orderId": 4
},
{
"name": "VAT",
"categoryType": 0,
"taxRates": [
{
"rate": 9,
"label": "A"
},
{
"rate": 0,
"label": "B"
}
],
"orderId": 5
```

```
    },
    {
      "name": "VAT-EXCL",
      "categoryType": 1,
      "taxRates": [
        {
          "rate": 0,
          "label": "C"
        }
      ],
      "orderId": 6
    }
  ]
},
"allTaxRates": [
  {
    "validFrom": "2020-08-02T21:04:00",
    "groupId": 1,
    "taxCategories": [
      {
        "name": "ECAL",
        "categoryType": 0,
        "taxRates": [
          {
            "rate": 5,
            "label": "D"
          },
          {
            "rate": 11,
            "label": "F"
          }
        ],
        "orderId": 1
      },
      {
        "name": "N-TAX",
        "categoryType": 0,
        "taxRates": [
          {
            "rate": 0,
            "label": "N"
          }
        ],
        "orderId": 2
      },
      {
        "name": "PBL",
        "categoryType": 2,
        "taxRates": [
          {
            "rate": 0.2,
            "label": "P"
          }
        ],
        "orderId": 3
      },
      {
        "name": "STT",
        "categoryType": 0,
        "taxRates": [
          {
            "rate": 6,
            "label": "E"
          }
        ],
        "orderId": 4
```

```
        },
        {
          "name": "VAT",
          "categoryType": 0,
          "taxRates": [
            {
              "rate": 9,
              "label": "A"
            },
            {
              "rate": 0,
              "label": "B"
            }
          ],
          "orderId": 5
        },
        {
          "name": "VAT-EXCL",
          "categoryType": 1,
          "taxRates": [
            {
              "rate": 0,
              "label": "C"
            }
          ],
          "orderId": 6
        }
      ]
    },
    {
      "validFrom": "2022-01-01T00:00:00Z",
      "groupId": 3,
      "taxCategories": [
        {
          "name": "ECAL",
          "categoryType": 0,
          "taxRates": [
            {
              "rate": 10,
              "label": "F"
            }
          ],
          "orderId": 5
        },
        {
          "name": "PB",
          "categoryType": 2,
          "taxRates": [
            {
              "rate": 0.1,
              "label": "P"
            }
          ],
          "orderId": 6
        },
        {
          "name": "STT",
          "categoryType": 0,
          "taxRates": [
            {
              "rate": 1,
              "label": "E"
            }
          ],
          "orderId": 2
        },
```

```
□□□□{
□□□□□"name": "VAT",
□□□□□"categoryType": 0,
□□□□□"taxRates": [
□□□□□□{
□□□□□□□"rate": 9,
□□□□□□□"label": "A"
□□□□□□}
□□□□□],
□□□□□"orderId": 1
□□□□}
□□□]
□□}
□]
}
```


Verify PIN

This service is used to verify a PIN entered by a cashier on a POS. Once the PIN is entered, it is validated using the Secure element on a smart card. The request is sent as plain text and the response is plain text (still application/json HTTP header must be provided).

If the command is successfully executed and the PIN is correct, E-SDC shall return "0100" and remember this PIN in its volatile memory (RAM) for all subsequent operations, until the smart card is removed from the reader or the E-SDC is switched off. Therefore there is no need for POS to send a PIN before every operation, only once it receives the response "0100".

However, if POS receives the error code "1500" during its regular operation, it must use this service again to provide a valid PIN (ask the user for the PIN or other way as per the POS internal policy). This case can be due to a card being removed, a card reader or E-SDC malfunctioning, etc.

In order to verify PIN, E-SDC invokes *Pin Verify* APDU command and passes PIN to the Secure element.

Endpoint

SDC	Endpoint	Example
V-SDC	`N/A`	`N/A`
E-SDC	`http://<ESDC_ip_address>:<ESDC_port`	`http://192.168.88.112:8888/api/v3/p`

Method

`POST`

Header

Add following HTTP headers to each request

- `Accept: application/json`
- `Content-Type: application/json`

Authentication

V-SDC

N/A

E-SDC

E-SDC does not require client authentication.

Request

String with PIN code sent from POS. PIN can contain only 4 ASCII digits 0-9.

Example

```
1234
```

Response Data

A string representation of one of the general Status and Error Codes:

- "0100" - command executed successfully and PIN is correct for the inserted card;
- "2100" - PIN code is wrong for the inserted card. Secure element responded with 0x6302 or 0x6303;
- "2110" - The number of allowed PIN entries exceeded. The card is locked for further use and has to be returned to Tax Office. Secure element responded with 0x6310;
- "1300" - E-SDC detects that the smart card is not inserted (or E-SDC fails to connect to it due to card or card reader issues or some other internal malfunction);
- "2220" - E-SDC cannot connect to the Secure Element applet;
- ~~"2210" - Secure Element is locked due to the card limit set by the Tax Service. No additional invoices can be signed before the audit (local or remote) is completed;~~
- ~~"2230" - Secure Element does not support requested protocol version (reserved for later use);~~
- "2400" - SDC device is not fully configured for invoice signing (i.e. tax rates or verification URL are missing etc.)
- "2806" - If provided PIN is in invalid format, other than ASCII digits 0-9.
- "1999" - E-SDC could not verify PIN, but specific case reason is not determined by error codes. Field *message* may contain more description. Manufacturer can use manufacturer-specific codes to describe

warning in more details.

For more information consult Status and Error Codes sections.

Example

```
"0100"
```

Create Invoice

This section contains the description of the **Create Invoice** command.

This command enables applying a digital signature on the transaction data received from the taxpayer's invoicing system. It provides non-repudiation of the signed invoice, i.e. the taxpayer cannot deny the content of the invoice nor that the invoice was signed using his/her secure element.

Endpoints

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 environment names, and every environment has its own URL. See Environments.

SDC	Endpoint	Example
V-SDC	<V-SDC_API_URL_obtained_from_certificate_as_exp here>/api/v3/invoices	`[[_TaxCore.PublicConfiguration.VSDCAp`
E-SDC	http://<ESDC_ip_address>:<ESDC_port>/api/	`http://192.168.88.112:8888/api/v3/inv`

Method

`POST`

Authentication

V-SDC

Use the client digital certificate for authentication with each request to V-SDC.Api.

E-SDC

E-SDC does not require client authentication.

Request

Headers

Add the following HTTP headers to each request

- `Accept: application/json`
- `Content-Type: application/json`
- `RequestId: <RequestId_value>` (Unique identifier of the request, generated by the POS system. It is used only for later request search, in case a response was not received. - **optional**). MaxLength: 32
- `Accept-Language: <Accept-Language_value>` (The list of one or more languages returned through the field **supportedLanguages** in the Get Status service response, ordered in POS language preference and separated by semi-colon. This affects the language and data formats on the invoice journal.)
    - SDC returns Invoice Result in the first supported language from the list
    - if none of the languages from the list are supported by SDC, it returns HTTP status code 406
    - if POS does not submit this parameter, SDC returns Invoice Result in the first language from the list obtained through the Get Status service
- `PAC: <PAC_value>` (PAC value for the provided client authentication certificate - **only for V-SDC**)

Invoice

Field	Optional/Mandatory	Description
dateAndTimeOfIssue	Optional	Current **Local Date and Time** in ISO 8601 format. This is an optional element on the API level. However, tax legislation in certain jurisdictions might mandate including it on invoices - so **make sure you become familiarized with the requirements in each jurisdiction**.
invoiceType	Mandatory	Invoice Type enumeration value: 0 - Normal, 1 - Proforma, 2 - Copy, 3 - Training, 4 - Advance
transactionType	Mandatory	Transaction Type enumeration value: 0 - Sale, 1 - Refund
payment	Mandatory	List of payments for the invoice,

		where each payment defines its method and amount. Each invoice must contain at least one payment element. In cases when the cashier needs to give change to the customer, the amount submitted in the `payment` field **must be the actual amount that stays in the cash register** (not the total amount that was given by the customer to the cashier).
cashier	Optional	Cashier's identification.
buyerId	Optional	Unique identification of the buyer/customer. It is **mandatory only if there is a legal obligation for buyer identification** (in both B2C and B2B transactions); otherwise, it's optional.
buyerCostCenterId	Optional	Cost Center ID provided by the buyer to the cashier or an identification of the invoice purpose. It is **mandatory only if there is a legal obligation for invoice purpose identification** (in both B2C and B2B transactions); otherwise, it's optional.
invoiceNumber	Optional	Invoice number generated by a POS **or** the software version of the accredited POS. This is an optional element on the API level. However, tax legislation in certain jurisdictions might mandate including it on invoices. **Make sure you become familiarized with specific jurisdiction requirements regarding its use and content.**
referentDocumentNumber	Mandatory for all Copies and Refunds and some Normal Sale or Advance Sale invoices	Mandatory only in case Invoice Type is **Refund**, **Copy** or **Advance Sale** connected to an Advance Sale (other jurisdiction-specific rules may apply). In all cases, this field must contain SDC Invoice Number of the previously issued invoice. In any other case, this field is optional.

		Must be in the **requestedBy-signedBy-Ordinal_Number** format. Unicode MaxLength: 50
referentDocumentDT	Optional	**SDC date and time** (local date and time in ISO 8601 format) of the document referenced in the `referentDocumentNumber` field. It is used to calculate taxes on the date of issue of the original document that is refunded or copied. If it is not provided in the request, SDC uses the currently active tax rates. It can be used only when there is a Reference Number (`referentDocumentNumber`) on the same invoice.
items (n)	Mandatory	Each invoice contains at least one item in the Items collection (E-SDC should support a minimum of 250, recommended up to 500)
options	Optional	Key/value collection defines the output of 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 to POS. E-SDC **MUST NOT** submit QR code to the tax authority as part of an audit package. Key: **omitTextualRepresentation** Value: "1" to omit generation of textual representation by E-SDC and "0" to generate return textual representation to POS. E-SDC **MUST ALWAYS** submit the textual representation of an invoice (journal) to the tax authority as part of an audit package.

Item

Each Item represents one line item on the invoice.

Field	Optional/Mandatory	Description
gtin	Optional	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	Mandatory	Human-readable name of the product or service.
quantity	Mandatory	The quantity of an item, with a maximum of 3 decimals. Example: 2 (pieces), 0.100 (grams).
unitPrice	Mandatory	Unit price of the line item. It does not take part in tax calculation. If there is a discount, the displayed unit price should show the price after the discount is applied. It must not include the value of the discount.
labels	Mandatory	The array of labels. Each Label represents one of the Tax Rates applied on the invoice item. Tax Items are calculated based on `totalAmount` and applied labels as described in the Calculate Taxes section. This field is mandatory (i.e. the caller must submit a non-empty collection) for each item, even when the price is 0.00.
totalAmount	Mandatory	Gross price for the line item - `unit price x quantity`. If there is a discount, the displayed amount should show the calculated price after the discount is applied to the line item. It must not include the value of the discount.

Payment

Field	Optional/Mandatory	Description
amount	Mandatory	Decimal amount of the payment
paymentType	Mandatory	Payment Type enumeration value: 0 - Other, 1 - Cash, 2 - Card, 3 - Check, 4 - Wire Transfer, 5 - Voucher, 6 - Mobile Money

Model

```
InvoiceRequest {
```

```
  dateAndTimeOfIssue (string, optional) MinValue: 1900-01-01T00:00:01Z, MaxValue: 9999-12-31T
  cashier (string) Unicode MaxLength:50,
  buyerId (string, optional) Unicode MaxLength:20,
  buyerCostCenterId (string, optional) Unicode MaxLength:50,
  invoiceType (string) = ['Normal', 'ProForma', 'Copy', 'Training', 'Advance'] (int) = [0,1,2
  transactionType (string) = ['Sale', 'Refund'] (int) = [0,1],
  payment (Array[Payment]),
  invoiceNumber (string, optional) Unicode MaxLength:60,
  referentDocumentNumber (string, optional) Unicode MaxLength: 50,
  referentDocumentDT (string, optional) MinValue: 1900-01-01T00:00:01Z, MaxValue: 9999-12-31T
  options (inline_model, optional),
  items (Array[Item])
}
payment {
  amount (number) MaxLength:29,
  paymentType (string) = ['Other', 'Cash', 'Card', 'Check', 'WireTransfer', 'Voucher', 'Mobil
}
inline_model {
omitQRCodeGen (string, optional) = ["0", "1"],
omitTextualRepresentation (string, optional) = ["0", "1"]
}
items {
  gtin (string, optional) MinLength:8 MaxLength:14,
  name (string) Unicode MaxLength:2048,
  quantity (number) Decimal(14,3) MinValue:0.001,
  labels (Array[string]) MinLength:1,
  unitPrice (number) Decimal(14,2),
  totalAmount (number) Decimal(14,2) MaxLength:29
}
```

Response

Headers

The following HTTP headers shall be returned in response

- `RequestId: <RequestId from Request HTTP Headers>`

Data Fields

Field	Description
requestedBy	UID of client's Secure Element digital certificate.
signedBy	UID of SDC's Secure Element digital certificate.
sdcDateTime	Local date and time in ISO 8601 format provided by E-SDC.
invoiceCounter	Invoice Counter in format **transactionTypeCounter/totalCounter invoiceCounterExtension** For Example: 14/17NS

invoiceCounterExtension	First letters of Invoice Type and Transaction Type of the invoice. NS for Normal Sale, CR – Copy Refund, TS – Training Sale, etc.
invoiceNumber	SDC Invoice Number in format **requestedBy-signedBy-totalCounter**
verificationUrl	VerificationURL generated in the *Create Verification URL* process
verificationQRCode	Base64 encoded byte array of GIF image created in the **Create QR Code** process
journal	Textual Representation of the invoice created in the *Create a Textual Representation of an Invoice (Receipt)* process
totalCounter	Total number of invoices signed by Secure Element. Returned by *Sign Invoice* APDU command
transactionTypeCounter	Total number of invoices for a requested type. Returned by *Sign Invoice* APDU command
totalAmount	Sum of all Items – total payable by the customer
encryptedInternalData	Base64 encoded byte array returned by *Sign Invoice* APDU command
signature	Base64 encoded byte array returned by *Sign Invoice* APDU command
taxItems	Array of TaxItem entities
businessName	Taxpayer Business Name obtained from digital certificate subject field **O**, as explained here
locationName	Location Name obtained from digital certificate subject field **OU**, as explained here
address	Street address obtained from digital certificate subject field **STREET**, as explained here
tin	Taxpayer's tax identification number obtained taxpayer's digital certificate, as explained here
district	District obtained from digital certificate subject field **S**, as explained here
taxGroupRevision	Revision of taxes used in the calculation
mrc	Manufacturer Registration Code is mandatory for an audit package sent to the tax authority database, but it's optional for invoice response sent to POS. It always has the format **ProductCode-ProductVersionCode-**

	DeviceSerialNumber - all 3 elements of MRC are mandatory. Explanation: **ProductCode** - unique 4 characters received from the tax authority during accreditation - case sensitive. **ProductVersionCode** - unique 4 characters received from the tax authority during accreditation - case sensitive. **DeviceSerialNumber** - serial number assigned by the manufacturer for each E-SDC installation - case sensitive (max 32 characters).
messages (optional)	Custom human-readable message that shall be printed or displayed by POS.

TaxItem

TaxItem represents tax liability on the invoice per one tax category.

Field	Description
label	Tax Label (A, F, G, N, P...)
categoryName	Tax Category Name (e.g. VAT, Consumption)
categoryType	Tax Category Type (0 - Tax on net, 1 - Tax on total, 2 - Amount per quantity)
rate	Tax rate percentage for Label (i.e. 12.50%)
amount	Tax amount calculated by E-SDC during invoice fiscalization

Model

```
InvoiceResult {
requestedBy (string, optional),
sdcDateTime (string),
invoiceCounter (string, read only),
invoiceCounterExtension (string, optional),
invoiceNumber (string, read only),
taxItems (Array[TaxItem], optional),
verificationUrl (string, optional),
verificationQRCode (string, optional),
journal (string, optional),
messages (string, optional),
signedBy (string),
encryptedInternalData (string),
signature (string, optional),
totalCounter (integer, optional),
transactionTypeCounter (integer, optional),
totalAmount (number, optional) MaxLength:29,
taxGroupRevision (integer, optional),
businessName (string, optional),
tin (string, optional),
locationName (string, optional),
address (string, optional),
district (string, optional),
mrc (string, optional)
}
```

```
TaxItem {
categoryType (integer, optional),
label (string),
amount (number) MaxLength: 29,
rate (number),
categoryName (string)
}
```

Examples

Below are examples of invoice requests and responses for all supported invoice and transaction types:

1. Mapping
   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 must use the following element mappings:

2. Normal Sale
   This is an example of Normal Sale Invoice

3. Normal Refund
   This is an example of Normal Refund

4. Advance Sale
   This is an example of Advance Sale Invoice

5. Advance Refund
   This is an example of Advance Refund.

6. Copy Sale
   This is an example of Copy Sale

7. Copy Refund
   This is an example of Copy Refund

8. Proforma Sale
   This is an example of Proforma Sale

9. Proforma Refund
   This is an example of Proforma Refund

10.

Training Sale
This is an example of Training Sale

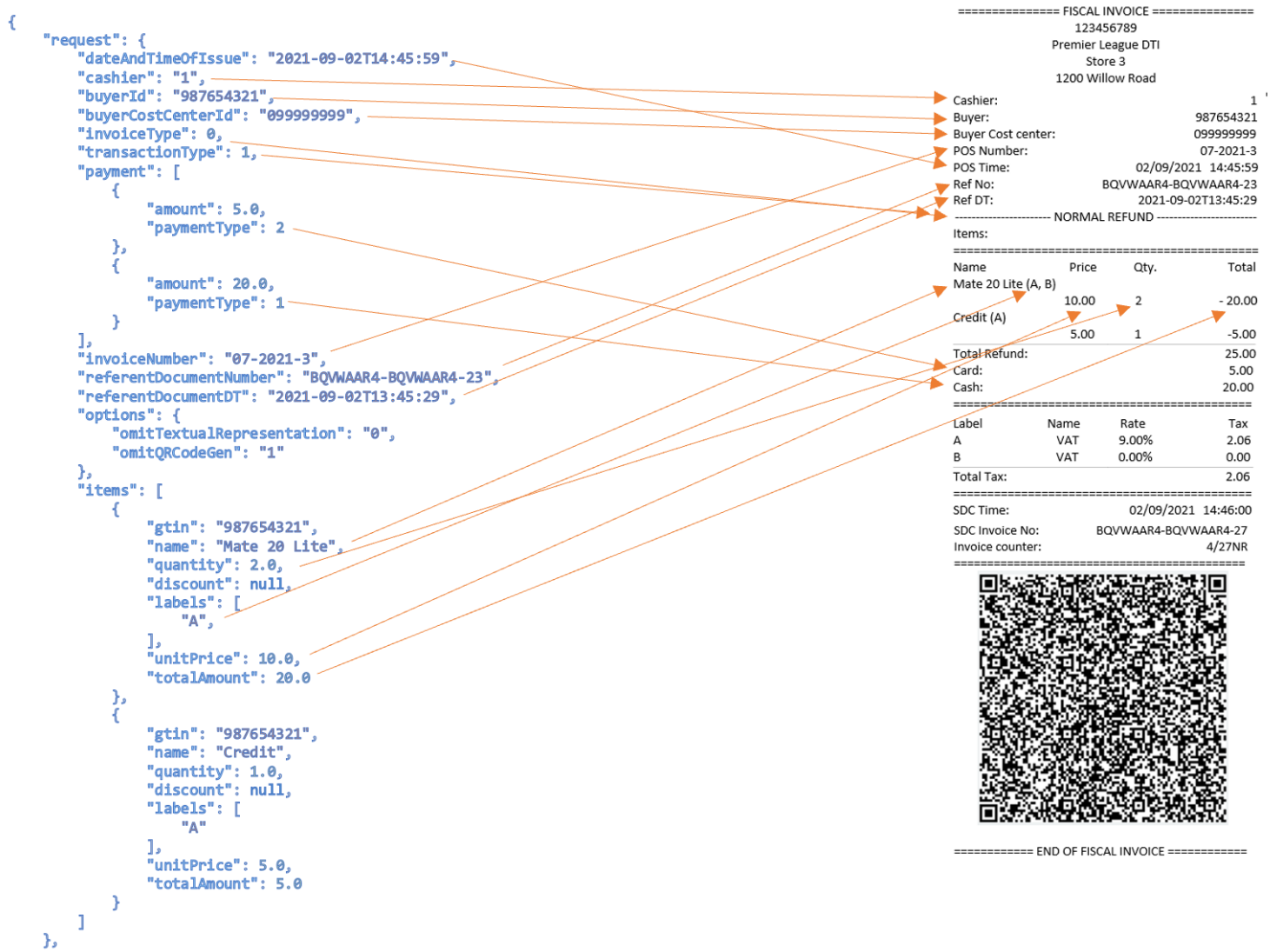
11.

Training Refund
This is an example of Training Refund

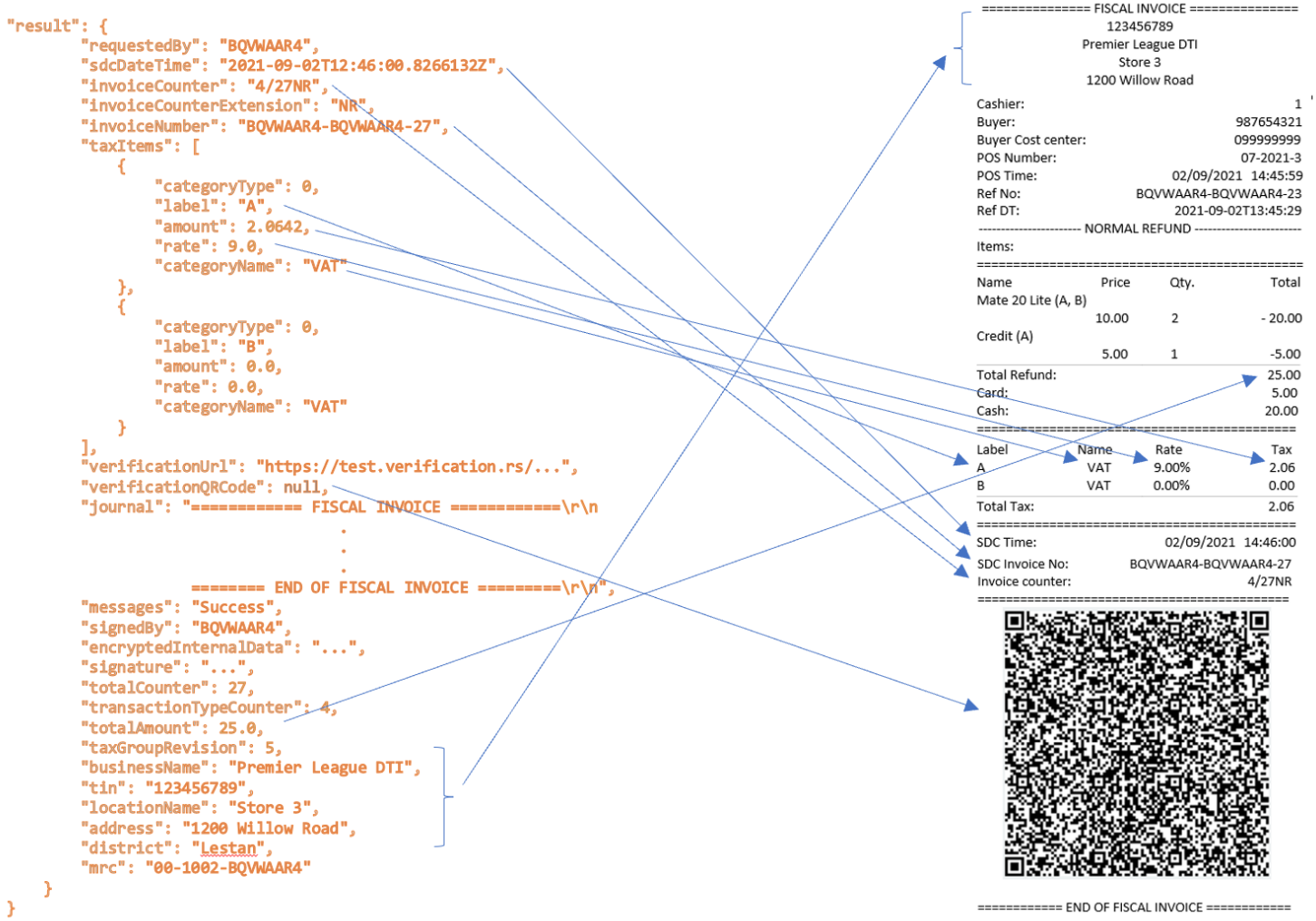
Mapping

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 must use the following element mappings:

Invoice request



Invoice response



Normal Sale

This is an example of Normal Sale Invoice

Request

Header

```
RequestId: 4c5730b50eb7b50e7a0292600a67ce16
```

Body

```
{
  "dateAndTimeOfIssue": "2020-12-08T08:55:23.286Z",
  "cashier": "123456789",
  "buyerId": "RS34564565",
  "buyerCostCenterId": "567546",
  "invoiceType": "Normal",
  "transactionType": "Sale",
  "payment": [
```

```
    {
      "amount": 68.46,
      "paymentType": "Cash"
    }
  ],
  "invoiceNumber": "POS2017/998",
  "referentDocumentNumber": "",
  "options": {
    "omitQRCodeGen" : "1" ,
    "omitTextualRepresentation" : "0"
  },
  "items": [
    {
      "name": "Sport-100 Helmet, Blue",
      "quantity": 2,
      "unitPrice": 34.23,
      "labels": [
        "A"
      ],
      "totalAmount": 68.46
    }
  ]
}
```


Response

Header

```
RequestId: 4c5730b50eb7b50e7a0292600a67ce16
```

Body


```
{
  "requestedBy": "WHKV8WJH",
  "sdcDateTime": "2020-12-09T01:51:43.1847161+12:00",
  "invoiceCounter": "6/16NS",
  "invoiceCounterExtension": "NS",
  "invoiceNumber": "WHKV8WJH-GFB38TO0-16",
  "taxItems": [
    {
      "categoryType": 0,
      "label": "A",
      "amount": 5.6527,
      "rate": 9,
      "categoryName": "VAT"
    }
  ],
  "verificationUrl": "https://frontendui.test.taxcore.dti.rs/v/?vl=AldIS1Y4V0pIR0ZCMzhUTzAQAA
  "verificationQRCode": null,
  "journal": "============ FISCAL INVOICE ============\r\n
TIN:                            HIHKLIKJ\r\n
Company:                Fiskalizacija doo\r\n
Store:                  Fiskalizacija doo\r\n
Address:           Milinka Milinkovica 44\r\n
District:                           Vinca\r\n
Cashier TIN:                    123456789\r\n
POS Number:                   POS2017/998\r\n
POS Time:             08/12/2020 08:55:23\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
---------------------------------------\r\n
Cash:                               68.46\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:            09/12/2020 01:51:43\r\n
SDC Invoice No:     WHKV8WJH-GFB38TO0-16\r\n
Invoice Counter:                 6/16NS\r\n
=======================================\r\n
======== END OF FISCAL INVOICE =========\r\n",
  "messages": "Success",
  "signedBy": "GFB38TO0",
  "encryptedInternalData": "aoo+9OCMgbe+a1OQyeJt44DqbE8KyBh9fj49saQNr3XcMRgHz+IE1oagmebqVucjN
  "signature": "YE6eOSCgZaz/eCVzeLQHvD02zWBBEOxFnQhFRiY1jNmaEbHi+7j91O/vXd47ui6qvKtYp+pahbbRZ
  "totalCounter": 16,
  "transactionTypeCounter": 6,
  "totalAmount": 68.46,
  "taxGroupRevision": 2,
  "businessName": "Shtelovanje doo",
  "tin": "HIHKLIKJ",
  "locationName": "Shtelovanje doo",
  "address": "Milinka Milinkovica 44",
  "address": "Vinca",
  "mrc": "0099-0100-GFB38TO0"
}
```


Normal Refund

This is an example of Normal Refund

Request

Header

```
RequestId: 4feb64edffca4ccdb19b9c44214a4c66
```

Body


```
{
  "dateAndTimeOfIssue": "2020-12-08T08:55:23.286Z",
  "cashier": "123456789",
  "buyerId": "RS34564565",
  "buyerCostCenterId": "567546",
  "invoiceType": "Normal",
```

```
  "transactionType": "Refund",
  "payment": [
    {
      "amount": 68.46,
      "paymentType": "Cash"
    }
  ],
  "invoiceNumber": "POS2017/998",
  "referentDocumentNumber": "YTSS6KKG-HJHR9EO0-305",
"referentDocumentDT": "2021-07-15T15:57:49.7423334+02:00",
  "options": {
    "omitQRCodeGen" : "1" ,
    "omitTextualRepresentation" : "0"
  },
  "items": [
    {
      "name": "Sport-100 Helmet, Blue",
      "quantity": 2,
      "unitPrice": 34.23,
      "labels": [
        "A"
      ],
      "totalAmount": 68.46
    }
  ]
}
```


Response

Header

```
RequestId: 4feb64edffca4ccdb19b9c44214a4c66
```

Body


```
{
  "requestedBy": "YTSS6KKG",
  "sdcDateTime": "2021-07-15T16:20:55.6856444+02:00",
  "invoiceCounter": "30/319NR",
  "invoiceCounterExtension": "NR",
  "invoiceNumber": "YTSS6KKG-HJHR9EO0-319",
  "taxItems": [
    {
      "categoryType": 0,
      "label": "A",
      "amount": 5.6527,
      "rate": 9,
      "categoryName": "VAT"
    }
  ],
  "verificationUrl": "https://ft8.test.taxcore.dti.rs/v/?vl=AllUU1M2S0tHSEpIUjlFTzA%2FAQAAHgA
  "verificationQRCode": null,
  "journal": "============ FISCAL INVOICE ============\r\n
TIN:                        RS543456345\r\n
Company:                    PKBB Livada\r\n
Store:                      PKBB Livada\r\n
Address:              Leskovacki drum bb\r\n
District:                       Lestane\r\n
Cashier TIN:                  123456789\r\n
```

```
Buyer TIN:                    RS34564565\r\n
Buyers Cost Center:               567546\r\n
POS Number:                  POS2017/998\r\n
POS Time:            08/12/2020 08:55:23\r\n
Ref No:            YTSS6KKG-HJHR9EO0-305\r\n
Ref DT:              15/07/2021 15:57:49\r\n
-------------NORMAL REFUND--------------\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 Refund:                      68.46\r\n
----------------------------------------\r\n
Cash:                              68.46\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:            15/07/2021 16:20:55\r\n
SDC Invoice No:    YTSS6KKG-HJHR9EO0-319\r\n
Invoice Counter:                30/319NR\r\n
========================================\r\n
======== END OF FISCAL INVOICE =========\r\n",
  "messages": "Success",
  "signedBy": "HJHR9EO0",
  "encryptedInternalData": "C6LKBVy3PlWQeDOEPJFcvl1xbpbxR8r6Rw/o/kkU/PI54u2NYvPlXSfRad+kuAOX3
  "signature": "Nt5xSWtfD1LJuoph+qd9kBONG8uExt6FmWgDtvMjr2Dyahxn+ZfXBaMgJeSDH99BgZsY4l9DRMb+r
  "totalCounter": 319,
  "transactionTypeCounter": 30,
  "totalAmount": 68.46,
  "taxGroupRevision": 3,
  "businessName": "PKBB Livada",
  "tin": "RS543456345",
  "locationName": "PKBB Livada",
  "address": "Leskovacki drum bb",
  "district": "Lestane",
  "mrc": "0099-3000-HJHR9EO0"
}
```


Advance Sale

This is an example of Advance Sale Invoice

Request

Header

```
RequestId: 8b795cf980c54f1ca7b40d9f5fe5fda3
```

Body

```
{
 "Cashier": "Software QA Engineer",
 "buyerId": "DT12345",
 "BuyerCostCenterId": "Data Tech",
 "InvoiceNumber": "07-2021-3",
 "DateAndTimeOfIssue": "2021-09-03T21:44:59",
 "ReferentDocumentNumber" : "",
 "referentDocumentDT" : "",
 "invoiceType": "Advance",
 "transactionType": "Sale",
 "payment": [
        {
            "amount": 5,
            "paymentType": "Card"
        },
        {
            "amount": 20,
            "paymentType": "Cash"
        }
    ],
 "Options": {
     "omitTextualRepresentation": 0,
     "omitQRCodeGen": 1
     },
 "Items": [
     {
     "GTIN": "987654321",
     "Name": "Mate 20 Lite",
     "Quantity": 2,
     "Labels": ["A","B"],
     "UnitPrice": 10,
     "TotalAmount": 20
     },
    {
     "GTIN": "987654321",
     "Name": "Credit",
     "Quantity": 1,
     "Labels": ["A"],
     "UnitPrice": 5,
     "TotalAmount": 5
   }
 ]
}
```


Response

Header

```
RequestId: 4c5730b50eb7b50e7a0292600a67ce16
```

Body


```
{
    "requestedBy": "BQVWAAR4",
    "sdcDateTime": "2021-09-03T21:45:48.5070512+02:00",
    "invoiceCounter": "3/30AS",
    "invoiceCounterExtension": "AS",
    "invoiceNumber": "BQVWAAR4-BQVWAAR4-30",
```

```
    "taxItems": [
        {
            "categoryType": 0,
            "label": "A",
            "amount": 2.0642,
            "rate": 9.00,
            "categoryName": "VAT"
        },
        {
            "categoryType": 0,
            "label": "B",
            "amount": 0.0000,
            "rate": 0.00,
            "categoryName": "VAT"
        }
    ],
    "verificationUrl": "https://ft8.test.taxcore.dti.rs/v/?vl=A0JRVldBQVI0QlFWV0FBUjQeAAAAAwA
    "verificationQRCode": "R0lGODlhhAGEAfcAAAAAAAAAMwAAZgAAmQAAzAAA/wArAAArMwArZgArmQArzAAr/v
    "journal": "============ FISCAL INVOICE ============\r\n
TIN:                            RS654321\r\n
Company:              Premier League DTI\r\n
Store:                Premier League DTI\r\n
Address:                    Kruzni put 7\r\n
District:                        Lestane\r\n
Cashier TIN:        Software QA Engineer\r\n
Buyer TIN:                       DT12345\r\n
Buyers Cost Center:            Data Tech\r\n
POS Number:                    07-2021-3\r\n
POS Time:            03/09/2021 21:44:59\r\n
--------------ADVANCE SALE--------------\r\n
Items\r\n
========================================\r\n
Name    Price        Qty.          Total\r\n
Mate 20 Lite (A, B)                    \r\n
        10.00          2           20.00\r\n
Credit (A)                             \r\n
         5.00          1            5.00\r\n
----------------------------------------\r\n
Total Purchase:                     25.00\r\n
Card:                                5.00\r\n
Cash:                               20.00\r\n
========================================\r\n
Label        Name     Rate           Tax\r\n
A             VAT    9.00%          2.06\r\n
B             VAT    0.00%          0.00\r\n
----------------------------------------\r\n
Total Tax:                           2.06\r\n
========================================\r\n
SDC Time:            03/09/2021 21:45:48\r\n
SDC Invoice No:     BQVWAAR4-BQVWAAR4-30\r\n
Invoice Counter:                  3/30AS\r\n
========================================\r\n
======== END OF FISCAL INVOICE =========\r\n",
    "messages": "Success",
    "signedBy": "BQVWAAR4",
    "encryptedInternalData": "Q/I4fZy0VicsCVoRN0kt7j3V4NkgDtc/L4okDB96nP0l27oOHPiPDgRNjwKdRfJ
    "signature": "UMNE9LKgn4Bn6R0HDlNVeRD2E+E9Ry4s+A/esxbDKSCXBxlv0LCuJ+OHXLhprbyTKx9N8h4JC2+
    "totalCounter": 30,
    "transactionTypeCounter": 3,
    "totalAmount": 25.0,
    "taxGroupRevision": 5,
    "businessName": "Premier League DTI",
    "tin": "RS654321",
    "locationName": "Premier League DTI",
    "address": "Kruzni put 7",
    "district": "Lestane",
```

```
    "mrc": "0001-1002-BQVWAAR4"
}
```

Advance Refund

This is an example of Advance Refund.

Request

Header

```
RequestId: 80b0ea559aff4058a030dc5c8ca01f31
```

Body

```
{
 "Cashier": "Software QA Engineer",
 "buyerId": "DT12345",
 "BuyerCostCenterId": "Data Tech",
 "InvoiceNumber": "07-2021-3",
 "DateAndTimeOfIssue": "2021-09-03T21:44:59",
 "ReferentDocumentNumber" : "BQVWAAR4-BQVWAAR4-30",
 "referentDocumentDT" : "2021-09-03T21:45:48",
 "invoiceType": "Advance",
 "transactionType": "Refund",
 "payment": [
        {
            "amount": 5,
            "paymentType": "Card"
        },
        {
            "amount": 20,
            "paymentType": "Cash"
        }
    ],
 "Options": {
     "omitTextualRepresentation": 0,
     "omitQRCodeGen": 1
     },
 "Items": [
     {
     "GTIN": "987654321",
     "Name": "Mate 20 Lite",
     "Quantity": 2,
     "Labels": ["A","B"],
     "UnitPrice": 10,
     "TotalAmount": 20
     },
    {
     "GTIN": "987654321",
     "Name": "Credit",
     "Quantity": 1,
     "Labels": ["A"],
     "UnitPrice": 5,
```

```
     "TotalAmount": 5
   }
 ]
}
```

Response

Header

```
RequestId: 4c5730b50eb7b50e7a0292600a67ce16
```

Body


```
{
    "requestedBy": "BQVWAAR4",
    "sdcDateTime": "2021-09-03T21:48:27.3129874+02:00",
    "invoiceCounter": "2/31AR",
    "invoiceCounterExtension": "AR",
    "invoiceNumber": "BQVWAAR4-BQVWAAR4-31",
    "taxItems": [
        {
            "categoryType": 0,
            "label": "A",
            "amount": 2.0642,
            "rate": 9.00,
            "categoryName": "VAT"
        },
        {
            "categoryType": 0,
            "label": "B",
            "amount": 0.0000,
            "rate": 0.00,
            "categoryName": "VAT"
        }
    ],
    "verificationUrl": "https://ft8.test.taxcore.dti.rs/v/?vl=A0JRVldBQVI0QlFWV0FBUjQfAAAAAgA
    "verificationQRCode": "R0lGODlhlAGUAfcAAAAAAAAAMwAAZgAAmQAAzAAA/wArAAArMwArZgArmQArzAAr/v
    "journal": "============ FISCAL INVOICE ============\r\n
TIN:                            RS654321\r\n
Company:               Premier League DTI\r\n
Store:                 Premier League DTI\r\n
Address:                     Kruzni put 7\r\n
District:                         Lestane\r\n
Cashier TIN:         Software QA Engineer\r\n
Buyer TIN:                       DT12345\r\n
Buyers Cost Center:             Data Tech\r\n
POS Number:                     07-2021-3\r\n
POS Time:             03/09/2021 21:44:59\r\n
Ref No:              BQVWAAR4-BQVWAAR4-30\r\n
Ref DT:               03/09/2021 21:45:48\r\n
-------------ADVANCE REFUND-------------\r\n
Items\r\n
=========================================\r\n
Name    Price        Qty.          Total\r\n
Mate 20 Lite (A, B)                    \r\n
        10.00          2         -20.00\r\n
Credit (A)                             \r\n
         5.00          1          -5.00\r\n
```

```
----------------------------------------\r\n
Total Refund:                       25.00\r\n
Card:                                5.00\r\n
Cash:                               20.00\r\n
========================================\r\n
Label        Name     Rate           Tax\r\n
A             VAT    9.00%          2.06\r\n
B             VAT    0.00%          0.00\r\n
----------------------------------------\r\n
Total Tax:                           2.06\r\n
========================================\r\n
SDC Time:            03/09/2021 21:48:27\r\n
SDC Invoice No:     BQVWAAR4-BQVWAAR4-31\r\n
Invoice Counter:                 2/31AR\r\n
========================================\r\n
======== END OF FISCAL INVOICE =========\r\n",
    "messages": "Success",
    "signedBy": "BQVWAAR4",
    "encryptedInternalData": "DSx6K4/kXJ576hf4xUvtNgIb28+Q/Yja72U6IaNmYN9UlDZ9BaP/S0U7cRv78C+
    "signature": "ZSkjT5VRY9vXwfk8h5cwA0n1p7DPNDBHLS7kVCYRJ2zq8FCy8DvVDcMfnKWoJzInvNLBk78jdyI
    "totalCounter": 31,
    "transactionTypeCounter": 2,
    "totalAmount": 25.0,
    "taxGroupRevision": 5,
    "businessName": "Premier League DTI",
    "tin": "RS654321",
    "locationName": "Premier League DTI",
    "address": "Kruzni put 7",
    "district": "Lestane",
    "mrc": "0001-1002-BQVWAAR4"
}
```


Copy Sale

This is an example of Copy Sale

Request

Header

```
RequestId: cf40377f2d1a4b5593f80008876dc016
```

Body


```
{
  "dateAndTimeOfIssue": "2020-12-08T08:55:23.286Z",
  "cashier": "123456789",
  "buyerId": "RS34564565",
  "buyerCostCenterId": "567546",
  "invoiceType": "Copy",
  "transactionType": "Sale",
  "payment": [
    {
      "amount": 68.46,
```

```
      "paymentType": "Cash"
    }
  ],
  "invoiceNumber": "POS2017/998",
  "referentDocumentNumber": "YTSS6KKG-HJHR9EO0-305",
"referentDocumentDT": "2021-07-15T15:57:49.7423334+02:00",
  "options": {
    "omitQRCodeGen" : "1" ,
    "omitTextualRepresentation" : "0"
  },
  "items": [
    {
      "name": "Sport-100 Helmet, Blue",
      "quantity": 2,
      "unitPrice": 34.23,
      "labels": [
        "A"
      ],
      "totalAmount": 68.46
    }
  ]
}
```


Response

Header


```
RequestId: cf40377f2d1a4b5593f80008876dc016
```


Body


```
{
  "requestedBy": "YTSS6KKG",
  "sdcDateTime": "2021-07-15T16:03:35.173426+02:00",
  "invoiceCounter": "31/307CS",
  "invoiceCounterExtension": "CS",
  "invoiceNumber": "YTSS6KKG-HJHR9EO0-307",
  "taxItems": [
    {
      "categoryType": 0,
      "label": "A",
      "amount": 5.6527,
      "rate": 9,
      "categoryName": "VAT"
    }
  ],
  "verificationUrl": "https://ft8.test.taxcore.dti.rs/v/?vl=AllUU1M2S0tHSEpIUjlFTzAzAQAAHwAA
  "verificationQRCode": null,
  "journal": "===== THIS IS NOT A FISCAL RECEIPT =====\r\n
TIN:                        RS543456345\r\n
Company:                    PKBB Livada\r\n
Store:                      PKBB Livada\r\n
Address:              Leskovacki drum bb\r\n
District:                       Lestane\r\n
Cashier TIN:                  123456789\r\n
Buyer TIN:                   RS34564565\r\n
Buyers Cost Center:              567546\r\n
POS Number:                 POS2017/998\r\n
POS Time:           08/12/2020 08:55:23\r\n
```

```
Ref No:            YTSS6KKG-HJHR9EO0-305\r\n
Ref DT:              15/07/2021 15:57:49\r\n
---------------COPY 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
----------------------------------------\r\n
Cash:                              68.46\r\n
========================================\r\n
      THIS IS NOT A FISCAL INVOICE      \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:            15/07/2021 16:03:35\r\n
SDC Invoice No:    YTSS6KKG-HJHR9EO0-307\r\n
Invoice Counter:                31/307CS\r\n
========================================\r\n
===== THIS IS NOT A FISCAL RECEIPT =====\r\n",
  "messages": "Success",
  "signedBy": "HJHR9EO0",
  "encryptedInternalData": "dBgBvpa/5YOqhOlYDcXTsMw9IAY4O+pggnDtEv/Ua1pvEa6JU4LtQ/IakGMN3BG7y
  "signature": "OWzvSqz/TcvWsG6UsL3U1QdvcfN755/37x/So4SHhH/3PbcviT419uLnHtUIj9UaDX6uGzniEouW/
  "totalCounter": 307,
  "transactionTypeCounter": 31,
  "totalAmount": 68.46,
  "taxGroupRevision": 3,
  "businessName": "PKBB Livada",
  "tin": "RS543456345",
  "locationName": "PKBB Livada",
  "address": "Leskovacki drum bb",
  "district": "Lestane",
  "mrc": "0099-3000-HJHR9EO0"
}
```


Copy Refund

This is an example of Copy Refund

Request

Header

```
RequestId: a02a9746466e46a0837990b3d5569ba3
```

Body

```
{
```

```
  "dateAndTimeOfIssue": "2020-12-08T08:55:23.286Z",
  "cashier": "123456789",
  "buyerId": "RS34564565",
  "buyerCostCenterId": "567546",
  "invoiceType": "Copy",
  "transactionType": "Refund",
  "payment": [
    {
      "amount": 68.46,
      "paymentType": "Cash"
    }
  ],
  "invoiceNumber": "POS2017/998",
  "referentDocumentNumber": "YTSS6KKG-HJHR9EO0-305",
"referentDocumentDT": "2021-07-15T15:57:49.7423334+02:00",
  "options": {
    "omitQRCodeGen" : "1" ,
    "omitTextualRepresentation" : "0"
  },
  "items": [
    {
      "name": "Sport-100 Helmet, Blue",
      "quantity": 2,
      "unitPrice": 34.23,
      "labels": [
        "A"
      ],
      "totalAmount": 68.46
    }
  ]
}
```


Response

Header

```
RequestId: a02a9746466e46a0837990b3d5569ba3
```

Body


```
{
  "requestedBy": "YTSS6KKG",
  "sdcDateTime": "2021-07-15T16:25:49.9232384+02:00",
  "invoiceCounter": "38/390CR",
  "invoiceCounterExtension": "CR",
  "invoiceNumber": "YTSS6KKG-HJHR9EO0-390",
  "taxItems": [
    {
      "categoryType": 0,
      "label": "A",
      "amount": 5.6527,
      "rate": 9,
      "categoryName": "VAT"
    }
  ],
  "verificationUrl": "https://ft8.test.taxcore.dti.rs/v/?vl=AllUU1M2S0tHSEpIUjlFTzCGAQAAJgAA
  "verificationQRCode": null,
  "journal": "===== THIS IS NOT A FISCAL RECEIPT =====\r\n
TIN:                        RS543456345\r\n
```

```
Company:                      PKBB Livada\r\n
Store:                        PKBB Livada\r\n
Address:                Leskovacki drum bb\r\n
District:                          Lestane\r\n
Cashier TIN:                     123456789\r\n
Buyer TIN:                      RS34564565\r\n
Buyers Cost Center:                567546\r\n
POS Number:                    POS2017/998\r\n
POS Time:              08/12/2020 08:55:23\r\n
Ref No:              YTSS6KKG-HJHR9EO0-305\r\n
Ref DT:                15/07/2021 15:57:49\r\n
--------------COPY REFUND---------------\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 Refund:                      68.46\r\n
----------------------------------------\r\n
Cash:                              68.46\r\n
========================================\r\n
      THIS IS NOT A FISCAL INVOICE      \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:              15/07/2021 16:25:49\r\n
SDC Invoice No:    YTSS6KKG-HJHR9EO0-390\r\n
Invoice Counter:                38/390CR\r\n
========================================\r\n
===== THIS IS NOT A FISCAL RECEIPT =====\r\n",
  "messages": "Success",
  "signedBy": "HJHR9EO0",
  "encryptedInternalData": "hsiPUq7UvtxWXCLdkmZ04o3eAqGwoeb9Agf9x0DVdADJhwWaI9IAbImZnXh9cY+y
  "signature": "Aypg62GkNMFKuCosQ2LVtVk1WYeWQCwtxX1rIFCDdzmQIGd3eEe4yBs3piQWAs3qJD84EcMGALGS
  "totalCounter": 390,
  "transactionTypeCounter": 38,
  "totalAmount": 68.46,
  "taxGroupRevision": 3,
  "businessName": "PKBB Livada",
  "tin": "RS543456345",
  "locationName": "PKBB Livada",
  "address": "Leskovacki drum bb",
  "district": "Lestane",
  "mrc": "0099-3000-HJHR9EO0"
}
```


Proforma Sale

This is an example of Proforma Sale

Request

Header

RequestId: 3f980e694ce045a9814ac145f4374531

Body


```
{
  "dateAndTimeOfIssue": "2020-12-08T08:55:23.286Z",
  "cashier": "123456789",
  "buyerId": "RS34564565",
  "buyerCostCenterId": "567546",
  "invoiceType": "Proforma",
  "transactionType": "Sale",
  "payment": [
    {
      "amount": 68.46,
      "paymentType": "Cash"
    }
  ],
  "invoiceNumber": "POS2017/998",
  "referentDocumentNumber": "YTSS6KKG-HJHR9EO0-305",
"referentDocumentDT": "2021-07-15T15:57:49.7423334+02:00",
  "options": {
    "omitQRCodeGen" : "1" ,
    "omitTextualRepresentation" : "0"
  },
  "items": [
    {
      "name": "Sport-100 Helmet, Blue",
      "quantity": 2,
      "unitPrice": 34.23,
      "labels": [
        "A"
      ],
      "totalAmount": 68.46
    }
  ]
}
```


Response

Header

RequestId: 3f980e694ce045a9814ac145f4374531

Body


```
{
  "requestedBy": "YTSS6KKG",
  "sdcDateTime": "2021-07-15T16:41:30.1405248+02:00",
  "invoiceCounter": "39/393PS",
  "invoiceCounterExtension": "PS",
  "invoiceNumber": "YTSS6KKG-HJHR9EO0-393",
  "taxItems": [
    {
      "categoryType": 0,
      "label": "A",
      "amount": 5.6527,
```

```
      "rate": 9,
      "categoryName": "VAT"
    }
  ],
  "verificationUrl": "https://ft8.test.taxcore.dti.rs/v/?vl=AllUU1M2S0tHSEpIUjlFTzCJAQAAJwAA
  "verificationQRCode": null,
  "journal": "===== THIS IS NOT A FISCAL RECEIPT =====\r\n
TIN:                         RS543456345\r\n
Company:                     PKBB Livada\r\n
Store:                       PKBB Livada\r\n
Address:               Leskovacki drum bb\r\n
District:                        Lestane\r\n
Cashier TIN:                   123456789\r\n
Buyer TIN:                    RS34564565\r\n
Buyers Cost Center:               567546\r\n
POS Number:                  POS2017/998\r\n
POS Time:             08/12/2020 08:55:23\r\n
-------------PROFORMA 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
----------------------------------------\r\n
Cash:                               68.46\r\n
========================================\r\n
      THIS IS NOT A FISCAL INVOICE      \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:             15/07/2021 16:41:30\r\n
SDC Invoice No:    YTSS6KKG-HJHR9EO0-393\r\n
Invoice Counter:                39/393PS\r\n
========================================\r\n
===== THIS IS NOT A FISCAL RECEIPT =====\r\n",
  "messages": "Success",
  "signedBy": "HJHR9EO0",
  "encryptedInternalData": "NH6NC4ixqSHJXOs+s8TNeCRj9+WbTJT1t2LshfHxU3keo78alMvsRBEN7QhR9bev
  "signature": "CvGKrQEG4qCNHDStyViKYc3ogCHhAw0NJhDHJUzUX6nryxVEEpwmxFOAOJFRUVCsqF2pNElav1fw
  "totalCounter": 393,
  "transactionTypeCounter": 39,
  "totalAmount": 68.46,
  "taxGroupRevision": 3,
  "businessName": "PKBB Livada",
  "tin": "RS543456345",
  "locationName": "PKBB Livada",
  "address": "Leskovacki drum bb",
  "district": "Lestane",
  "mrc": "0099-3000-HJHR9EO0"
}
```


Proforma Refund

This is an example of Proforma Refund

Request

Header

```
RequestId: d0bdc3dcb8b442c0bd7231ccff59f027
```

Body


```
{
  "dateAndTimeOfIssue": "2020-12-08T08:55:23.286Z",
  "cashier": "123456789",
  "buyerId": "RS34564565",
  "buyerCostCenterId": "567546",
  "invoiceType": "Proforma",
  "transactionType": "Refund",
  "payment": [
    {
      "amount": 68.46,
      "paymentType": "Cash"
    }
  ],
  "invoiceNumber": "POS2017/998",
  "referentDocumentNumber": "YTSS6KKG-HJHR9EO0-305",
"referentDocumentDT": "2021-07-15T15:57:49.7423334+02:00",
  "options": {
    "omitQRCodeGen" : "1" ,
    "omitTextualRepresentation" : "0"
  },
  "items": [
    {
      "name": "Sport-100 Helmet, Blue",
      "quantity": 2,
      "unitPrice": 34.23,
      "labels": [
        "A"
      ],
      "totalAmount": 68.46
    }
  ]
}
```


Response

Header

```
RequestId: d0bdc3dcb8b442c0bd7231ccff59f027
```

Body


```
{
  "requestedBy": "YTSS6KKG",
  "sdcDateTime": "2021-07-15T16:44:44.0877791+02:00",
  "invoiceCounter": "37/394PR",
```

```
  "invoiceCounterExtension": "PR",
  "invoiceNumber": "YTSS6KKG-HJHR9EO0-394",
  "taxItems": [
    {
      "categoryType": 0,
      "label": "A",
      "amount": 5.6527,
      "rate": 9,
      "categoryName": "VAT"
    }
  ],
  "verificationUrl": "https://ft8.test.taxcore.dti.rs/v/?vl=AllUU1M2S0tHSEpIUjlFTzCKAQAAJQAA
  "verificationQRCode": null,
  "journal": "===== THIS IS NOT A FISCAL RECEIPT =====\r\n
TIN:                         RS543456345\r\n
Company:                     PKBB Livada\r\n
Store:                       PKBB Livada\r\n
Address:              Leskovacki drum bb\r\n
District:                        Lestane\r\n
Cashier TIN:                   123456789\r\n
Buyer TIN:                    RS34564565\r\n
Buyers Cost Center:               567546\r\n
POS Number:                  POS2017/998\r\n
POS Time:            08/12/2020 08:55:23\r\n
Ref No:            YTSS6KKG-HJHR9EO0-305\r\n
Ref DT:              15/07/2021 15:57:49\r\n
------------PROFORMA REFUND-------------\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 Refund:                       68.46\r\n
----------------------------------------\r\n
Cash:                               68.46\r\n
========================================\r\n
      THIS IS NOT A FISCAL INVOICE      \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:            15/07/2021 16:44:44\r\n
SDC Invoice No:    YTSS6KKG-HJHR9EO0-394\r\n
Invoice Counter:                37/394PR\r\n
========================================\r\n
===== THIS IS NOT A FISCAL RECEIPT =====\r\n",
  "messages": "Success",
  "signedBy": "HJHR9EO0",
  "encryptedInternalData": "AOF3Rh6qYd2E3vI2ub0mKqW9CbeFKF6br+dKl8COblPTlWBWP+Sr22PkgHjT9HI6
  "signature": "AMr8SA1WxDwUHC7UrKJMcGkXgJuEASwzRYa3XeHecNEMvxdOXZzKnpus9NAEDGU+I3TIyYPuCFKL
  "totalCounter": 394,
  "transactionTypeCounter": 37,
  "totalAmount": 68.46,
  "taxGroupRevision": 3,
  "businessName": "PKBB Livada",
  "tin": "RS543456345",
  "locationName": "PKBB Livada",
  "address": "Leskovacki drum bb",
  "district": "Lestane",
  "mrc": "0099-3000-HJHR9EO0"
}
```

Training Sale

This is an example of Training Sale

Request

Header

```
RequestId: a44cf40b1f2b4afabd1d9c4f1066b92d
```

Body

```
{
  "dateAndTimeOfIssue": "2020-12-08T08:55:23.286Z",
  "cashier": "123456789",
  "buyerId": "RS34564565",
  "buyerCostCenterId": "567546",
  "invoiceType": "Training",
  "transactionType": "Sale",
  "payment": [
    {
      "amount": 68.46,
      "paymentType": "Cash"
    }
  ],
  "invoiceNumber": "POS2017/998",
  "referentDocumentNumber": "YTSS6KKG-HJHR9EO0-305",
"referentDocumentDT": "2021-07-15T15:57:49.7423334+02:00",
  "options": {
    "omitQRCodeGen" : "1" ,
    "omitTextualRepresentation" : "0"
  },
  "items": [
    {
      "name": "Sport-100 Helmet, Blue",
      "quantity": 2,
      "unitPrice": 34.23,
      "labels": [
        "A"
      ],
      "totalAmount": 68.46
    }
  ]
}
```

Response

Header

```
RequestId: d0bdc3dcb8b442c0bd7231ccff59f027
```

Body


```
{
  "requestedBy": "YTSS6KKG",
  "sdcDateTime": "2021-07-15T16:47:48.3965089+02:00",
  "invoiceCounter": "39/395TS",
  "invoiceCounterExtension": "TS",
  "invoiceNumber": "YTSS6KKG-HJHR9EO0-395",
  "taxItems": [
    {
      "categoryType": 0,
      "label": "A",
      "amount": 5.6527,
      "rate": 9,
      "categoryName": "VAT"
    }
  ],
  "verificationUrl": "https://ft8.test.taxcore.dti.rs/v/?vl=AllUU1M2S0tHSEpIUjlFTzCLAQAAJwAA
  "verificationQRCode": null,
  "journal": "===== THIS IS NOT A FISCAL RECEIPT =====\r\n
TIN:                         RS543456345\r\n
Company:                     PKBB Livada\r\n
Store:                       PKBB Livada\r\n
Address:              Leskovacki drum bb\r\n
District:                        Lestane\r\n
Cashier TIN:                   123456789\r\n
Buyer TIN:                    RS34564565\r\n
Buyers Cost Center:               567546\r\n
POS Number:                  POS2017/998\r\n
POS Time:            08/12/2020 08:55:23\r\n
-------------TRAINING 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
----------------------------------------\r\n
Cash:                               68.46\r\n
========================================\r\n
      THIS IS NOT A FISCAL INVOICE      \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:            15/07/2021 16:47:48\r\n
SDC Invoice No:    YTSS6KKG-HJHR9EO0-395\r\n
Invoice Counter:                39/395TS\r\n
========================================\r\n
===== THIS IS NOT A FISCAL RECEIPT =====\r\n",
  "messages": "Success",
  "signedBy": "HJHR9EO0",
  "encryptedInternalData": "UKsZsoFmKMEW2PLX+Ex/P/H8El355WZPrVU+jrJYKjsm1dGOvvWxDVTc5pJkRltQ
  "signature": "QbFJICVTO2OWru4zViu2KKbLW7hPhpA/ED6AVD8H1SkRZBhNQdu/tXr61GuuOILBPuI1gal02MXs
  "totalCounter": 395,
  "transactionTypeCounter": 39,
  "totalAmount": 68.46,
  "taxGroupRevision": 3,
  "businessName": "PKBB Livada",
  "tin": "RS543456345",
  "locationName": "PKBB Livada",
```

```
  "address": "Leskovacki drum bb",
  "district": "Lestane",
  "mrc": "0099-3000-HJHR9EO0"
}
```

Training Refund

This is an example of Training Refund

Request

Header

```
RequestId: 8caa71f8dca849908872b0d14f4c33fb
```

Body

```
{
  "dateAndTimeOfIssue": "2020-12-08T08:55:23.286Z",
  "cashier": "123456789",
  "buyerId": "RS34564565",
  "buyerCostCenterId": "567546",
  "invoiceType": "Training",
  "transactionType": "Refund",
  "payment": [
    {
      "amount": 68.46,
      "paymentType": "Cash"
    }
  ],
  "invoiceNumber": "POS2017/998",
  "referentDocumentNumber": "YTSS6KKG-HJHR9EO0-305",
"referentDocumentDT": "2021-07-15T15:57:49.7423334+02:00",
  "options": {
    "omitQRCodeGen" : "1" ,
    "omitTextualRepresentation" : "0"
  },
  "items": [
    {
      "name": "Sport-100 Helmet, Blue",
      "quantity": 2,
      "unitPrice": 34.23,
      "labels": [
        "A"
      ],
      "totalAmount": 68.46
    }
  ]
}
```

Response

Header

RequestId: 8caa71f8dca849908872b0d14f4c33fb

Body


```
{
  "requestedBy": "YTSS6KKG",
  "sdcDateTime": "2021-07-15T17:01:53.0192859+02:00",
  "invoiceCounter": "37/396TR",
  "invoiceCounterExtension": "TR",
  "invoiceNumber": "YTSS6KKG-HJHR9EO0-396",
  "taxItems": [
    {
      "categoryType": 0,
      "label": "A",
      "amount": 5.6527,
      "rate": 9,
      "categoryName": "VAT"
    }
  ],
  "verificationUrl": "https://ft8.test.taxcore.dti.rs/v/?vl=AllUU1M2S0tHSEpIUjlFTzCMAQAAJQAA
  "verificationQRCode": null,
  "journal": "===== THIS IS NOT A FISCAL RECEIPT =====\r\n
TIN:                         RS543456345\r\n
Company:                     PKBB Livada\r\n
Store:                       PKBB Livada\r\n
Address:              Leskovacki drum bb\r\n
District:                        Lestane\r\n
Cashier TIN:                   123456789\r\n
Buyer TIN:                    RS34564565\r\n
Buyers Cost Center:               567546\r\n
POS Number:                  POS2017/998\r\n
POS Time:            08/12/2020 08:55:23\r\n
Ref No:            YTSS6KKG-HJHR9EO0-305\r\n
Ref DT:              15/07/2021 15:57:49\r\n
------------TRAINING REFUND-------------\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 Refund:                       68.46\r\n
----------------------------------------\r\n
Cash:                               68.46\r\n
========================================\r\n
      THIS IS NOT A FISCAL INVOICE      \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:            15/07/2021 17:01:53\r\n
SDC Invoice No:    YTSS6KKG-HJHR9EO0-396\r\n
Invoice Counter:                37/396TR\r\n
========================================\r\n
===== THIS IS NOT A FISCAL RECEIPT =====\r\n",
  "messages": "Success",
  "signedBy": "HJHR9EO0",
  "encryptedInternalData": "brqW02fw5qyqAC4AkSyc8K7zrN2K9BTZL5cofnNYMEEysyqsPVfmK6ipB4a6ie8o
```

```
  "signature": "lUfjvl/LqBOn7koT7qeKQqv6xTA+UQ3iBMUqz2UB13hjSrBRK/41axT3WTdMJSTeuIURUoxCLNIRI
  "totalCounter": 396,
  "transactionTypeCounter": 37,
  "totalAmount": 68.46,
  "taxGroupRevision": 3,
  "businessName": "PKBB Livada",
  "tin": "RS543456345",
  "locationName": "PKBB Livada",
  "address": "Leskovacki drum bb",
  "district": "Lestane",
  "mrc": "0099-3000-HJHR9EO0"
}
```


Get Last Signed Invoice

Get the last signed invoice by `RequestId`. It is used in case POS did not get a response from E-SDC after the Create Invoice service was invoked. The response is the same as for the Create Invoice service. `RequestId` field from the request is used to determine whether the last invoice was successfully created.

Endpoint

SDC	Endpoint	
V-SDC	<V-SDC_API_URL_obtained_from_certificate_as_exp here>/api/v3/invoices/{requestId}	`[[_TaxCore.PublicConfiguration.VSDCAp`
E-SDC	`http://<ESDC_ip_address>:<port>/api`	`http://192.168.88.112:8888/api/v3/inv`

NOTE:
Development and production environments, as well the environments in different countries, have different URLs. For this reason, URLs and names in your documentation, code and UI should not be hardcoded but configurable or extracted from a digital certificate.

Method

`GET`

Authentication

V-SDC

N/A

E-SDC

E-SDC does not require client authentication.

Request

N/A

Response

If found, same response structure as the response model for Create Invoice service. If not found, null is returned.

NOTE:
The main purpose of the command is to receive information about the last signed invoice, so that in case of failure, POS can decide whether it is necessary to repeat an invoice request. There are various alternative scenarios that may arise from this, which E-SDC should process in accordance with its technical capabilities and capacities, as well as depending on the intended implementation and the guarantees it can offer to POS:

1. If more than one POS is using the same E-SDC, the uniqueness of the RequestId should be ensured at the level of all acceding POS. In this regard, it is recommended to use a GUID with each request, or a combination of `POS_id` + `request_id`. Also, in this case, the last invoice according to the POS, or one of the last N requests, should be saved, depending on the capabilities of the E-SDC.
2. If one E-SDC is issuing invoices with multiple POSs and multiple Secure Elements, then the E-SDC should store at least 1 latest invoice response for each unique combination of `request_id` + secure element UID. Depending on the implementation, E-SDC can store more than 1 latest responses for each combination.
3. The E-SDC should store this data in a non-volatile memory so that in the event of a failure (e.g. power failure) it can read the stored data.
4. E-SDC should keep this data at least until the next request.

Error Messages Format

This section describes the structure and format of error messages that SDC returns 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	Array of objects containing key-value pairs of property-errorCodes .
property	Path to the property error codes are associated with. If some of the fields in the path is an array, index number shall be included in square brackets. If errorcode is not related to a specific JSON property in the request, leave empty string.
errors	Error codes associated with a particular property. Codes are from the section Status and Error Codes.

Model

```
ModelErrors {
message (string),
modelState (Array[ModelError])
}

ModelError {
property(string),
errors(Array[string])
}
```

Example

```
{
    "message": "The request is invalid.",
    "modelState": [
        {
            "property": "invoiceType",
            "errors": [
                "1234"
            ]
        },
        {
            "property": "items[0].labels[0]",
            "errors": [
                "1234"
            ]
        },
        {
            "property": "items[0].gtin",
            "errors": [
                "1234",
                "5678"
            ]
        }
    ]
}
```

Status and Error Codes

All protocols share the same Info, Error and Warning 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	V-SDC, 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	A field is required (a mandatory invoice request field is missing)	V-SDC, E-SDC
2801	Field Value Too Long	The length of the field value is longer than expected	V-SDC, E-SDC
2802	Field Value Too Short	The length of the field value is shorter than expected	V-SDC, 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	A field value out of expected range	VSDC, E-SDC
2805	Invalid Field Value	A 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 of items or the list of tax labels in the invoice request does not contain at least one element (item/label).	VSDC, E-SDC
2808	List Too Long	The list of items or the list of tax labels in the invoice request exceeds the maximum allowed number of elements (items/labels) or byte size. The maximum values depend on an SDC's capacity for processing invoice requests and can be manufacturer-specific.	VSDC, E-SDC
2809	Secure Element Expired	The digital certificate on the smart card secure element has expired.	E-SDC

Obsolete error codes

Code	Error	Description	Applies to
~~**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

For ESDC developers

E-SDCs are software applications or hardware devices whose main function is to:

- fiscalize invoices,
- enable issuing invoices with or without the internet (by safeguarding invoices in its memory)
- deliver fiscalized audit packages to the tax authority

- enable the audit of fiscalized invoices

From the technical point of view, E-SDC is a middleware component that connects an accredited invoicing system (POS) to a secure element and enables standardized communication with TaxCore.API.

This article offers useful initial information about technical requirements for accrediting E-SDC solutions.

NOTE:
Before reading the rest of this article, make sure you have read Getting Started With Accreditation
Also, before you start developing your solution, please read General Information for all vendors.

This article offers E-SDC vendors who are interested in accrediting an E-SDC solution the following insight:

- quick guide for E-SDC accreditation
- desired/needed professional experience for developing an E-SDC product
- how an E-SDC solution fits in with other EFD components?
- how to navigate the rest of the technical instructions in order to initialize development?

Quick step-by-step guide for E-SDC accreditation

1. Register as a vendor for the Sandbox environment
2. Receive a Developer Certificate and use it to access the Developer Portal
3. Use the Developer Portal to request additional certificates (smart cards) for testing purposes
4. Consult all the sections in these technical instructions to see understand all the requirements and how they should be implemented
5. Use the testing application SDC Analyzer on the Developer Portal to test your E-SDC's operation
6. Compile user documentation for your POS
7. Use the My Accreditations section on the Developer Portal to apply for accrediting your POS

Which professional experience do I need for developing an E-SDC?

Be mindful that developing E-SDCs is much more challenging than developing POS solutions.

For that reason, we have compiled a list of desired professional experiences in order to prepare developers for the challenges that an E-SDC development puts forwards:

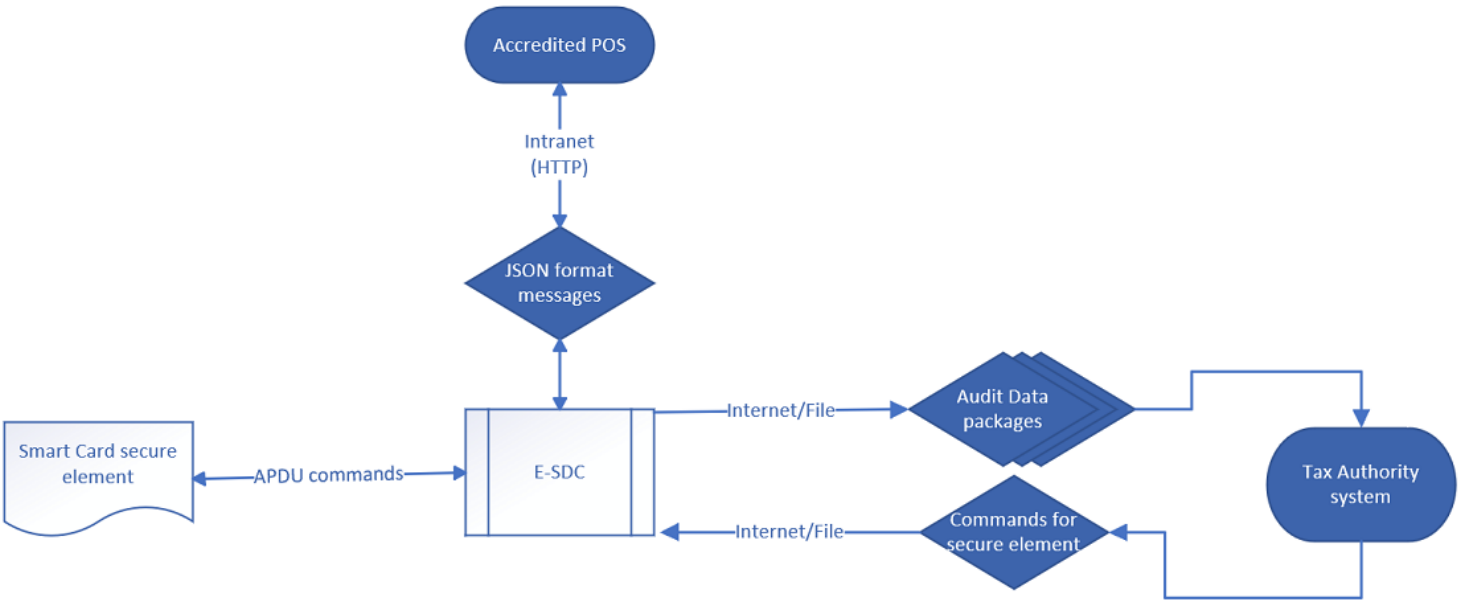
- a clear understanding of the multi-environment development concept - E-SDCs are developed in the development Sandbox environment and used by taxpayers in the official Production environment
- using APDU commands for communication with secure elements
- using Client Certificate (certificate store, pkcs12, pkcs11) authentication via HTTP protocol
- calling Web API
- using RSA and AES algorithms for encryption
- parsing X.509 certificates and using the obtained data

Integration with other EFD components

E-SDC is one of three components of any EFD setup.

NOTE:
For an overview of all EFD components and how they communicate with each other, see Electronic Fiscal Device.
For an overview of how EFD components interact to create a fiscal invoice, see Fiscal Invoice.

The graph below shows the communication pattern that an E-SDC establishes with other EFD components.



What is the typical process flow of E-SDC operations?

All E-SDC solutions must follow the basic steps in the process of configuration and creating fiscal invoices (although some might have additional, manufacturer-specific, steps).

For a detailed step-by-step explanation of all standard E-SDC operations, see Processes.

NOTE:
Be mindful that E-SDC must be initialized every time when a new smart card is used. For more details, see E-SDC Initialization.

Communication with an accredited invoicing system (POS)

E-SDC communicates with an accredited invoicing system (POS) to receive invoice requests and return invoice responses - as part of fiscalizing each invoice.

Communication between a POS solution and E-SDC is established through JSON formatted messages using the POS to SDC Protocol.

Communication with a secure element

E-SDC communicates with a secure element (issued to taxpayers a smart card) to:

- perform fiscalization of data submitted as an invoice request from POS
- enable the audit of invoices fiscalized by that secure element
- enable transmission of other messages exchanged between the secure element and the tax authority (e.g. transferring new tax rates to the secure element)

Communication between an E-SDC and a secure element is established using APDU Commands.

For authentication purposes, an E-SDC also must be able to communicate with the PKI Applet installed on each smart card secure element. For more information, see PKI Applet.

Communication with the TaxCore.API (tax authority system)

E-SDC communicates with TaxCore.API to:

- submit fiscalized audit packages to the TaxCore database
- receive commands from TaxCore (which it then forwards to the secure element)

Communication between E-SDC and tax authority is established via TaxCore.API.

File-based communication

In case of prologued internet failure, E-SDC must allow file-based communication with the tax authority's system for the purpose of configuration and performing local audits.

All sections of the technical instructions

Technical instructions specific for E-SDC vendors/developers consist of the following sections:

1. High Level Requirements
   This section describes high-level requirements to consider when

2. Architecture
   This figure shows the components of a fiscalization system and their mutual relationship.

3. Data Structures
   This section contains descriptions of the main data structures used during the fiscalization and audit processes.

4. Processes
   This section describes processes performed by an E-SDC.

5. Protocols
   This section describes Application Programming Interfaces (API) and protocols exposed by an E-SDC or used by an E-SDC to communicate with the other components (TaxCore.API, Secure element Applet, PKI Applet or SD Card/USB Flash Drive) required to fulfill its primary role – to safeguard a transaction and to transfer the audit packages to the tax authority's system.

6. Manuals
   E-SDC must have a user manual that explains the following topics in detail:

Related articles

- EFD Vendors General Information

High-Level Requirements

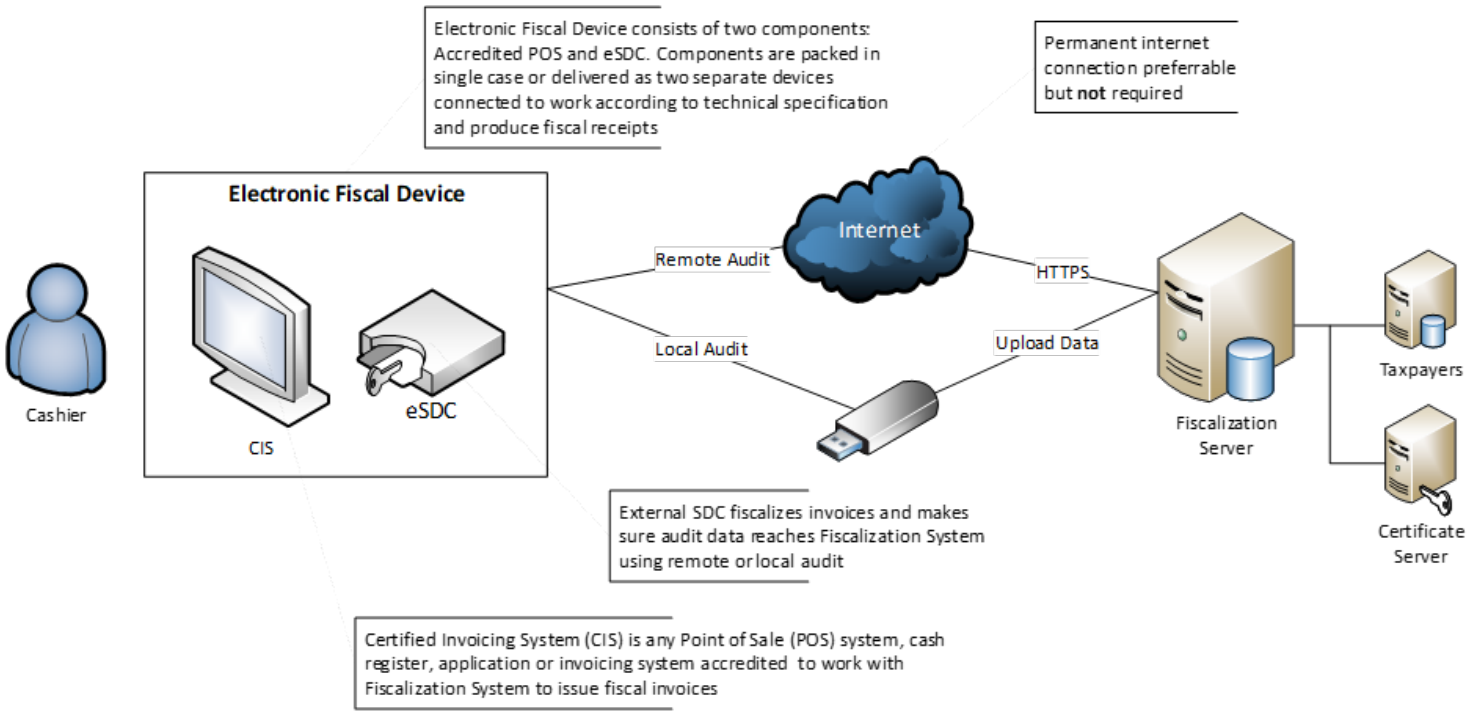
This section describes high-level requirements to consider when designing an E-SDC.

1. E-SDC is able to generate an invoice without an Internet connection.
2. E-SDC relies on a Secure Element (delivered as a smart card) to safeguard an invoice.
3. E-SDC must be designed so that it can work in multiple environments (development and production), depending on the certificate on the secure element it is using.
4. E-SDC calculates tax liabilities based on received Invoice items and defined tax rates.
5. Initial E-SDC configuration can be performed via a file (e.g. on USB disk) or the Internet connection.
6. E-SDC exposes a standardized protocol to a POS - JSON via HTTP.
7. E-SDC processes all commands received from the tax authority's system in consecutive order. These commands can include time synchronization, an update of tax rates, etc.
8. E-SDC encrypts audit data and stores it locally in an encrypted form.
9. E-SDC periodically sends stored audit data to a tax authority and this process is called an audit.
10. E-SDC does not have to keep audit data submitted and successfully stored on the tax authority's system.
11. E-SDC submits a proof of audit (PoA) generated by the Tax Service's system to the Secure Element as soon as the E-SDC receives it.
12.

E-SDC does not store the Secure Element's PIN code except in the working memory. Once the E-SDC is restarted, the cashier is required to enter the PIN code again.

13. E-SDC stores Audit Packages in its non-volatile memory for the length of the period prescribed by the governing tax authority.
14. E-SDC keeps a log of all required error events with the exact local date and time

This figure shows the high-level architecture of Fiscalization System, involving an Accredited POS, E-SDC and tax authority's system (TaxCore).



Connectivity And Modes of Operation

Introduction

External Sales Data Controller (E-SDC) device exposes HTTP protocol for communication with an Accredited POS via a UTP cable, Wi-Fi and similar.

E-SDC uses a Secure Element to digitally sign invoices received from the Accredited POS and to produce audit data. The audit data is stored on the E-SDC's internal non-volatile memory, which enables a local and remote audit.

Multiple POSs can be connected to a single E-SDC. However, this should be avoided as multiple devices could send the data simultaneously, and since every smart card has its own limitations (resources, processing speed), that could slow down the overall process at sale point.

Use an online mode whenever possible

Taxpayers are encouraged to use an online mode whenever it's possible - V-SDC service will be widely available and accessible for a variety of Accredited POS devices and software solutions. But, in order to rollout, a fiscalization system needs to have the ability to close any possible gap in the fiscal discipline due to poor network coverage or the Internet unavailability.

Semi-Connected mode

E-SDC must be able to work in the Semi-offline mode.

In the semi-offline mode, the E-SDC can work both online (if the Internet is available) and offline (if the Internet is not available).

In the semi-offline mode, the Secure Element signs an invoice and the E-SDC device immediately tries to contact the tax authority's system and perform a remote audit by invoking Submit Audit Package web method.

If the tax authority's system is not accessible (for example, the internet connection is interrupted), the E-SDC automatically switches to the offline operation. In the offline operation, the Secure Element signs an invoice and the E-SDC device stores an audit package locally in a secure manner.

When the connection with the tax authority system is re-established, the E-SDC will send the locally stored audit packages.

If the connection is interrupted for a prolonged period, the audit packages will be retrieved through a local audit process.

Architecture

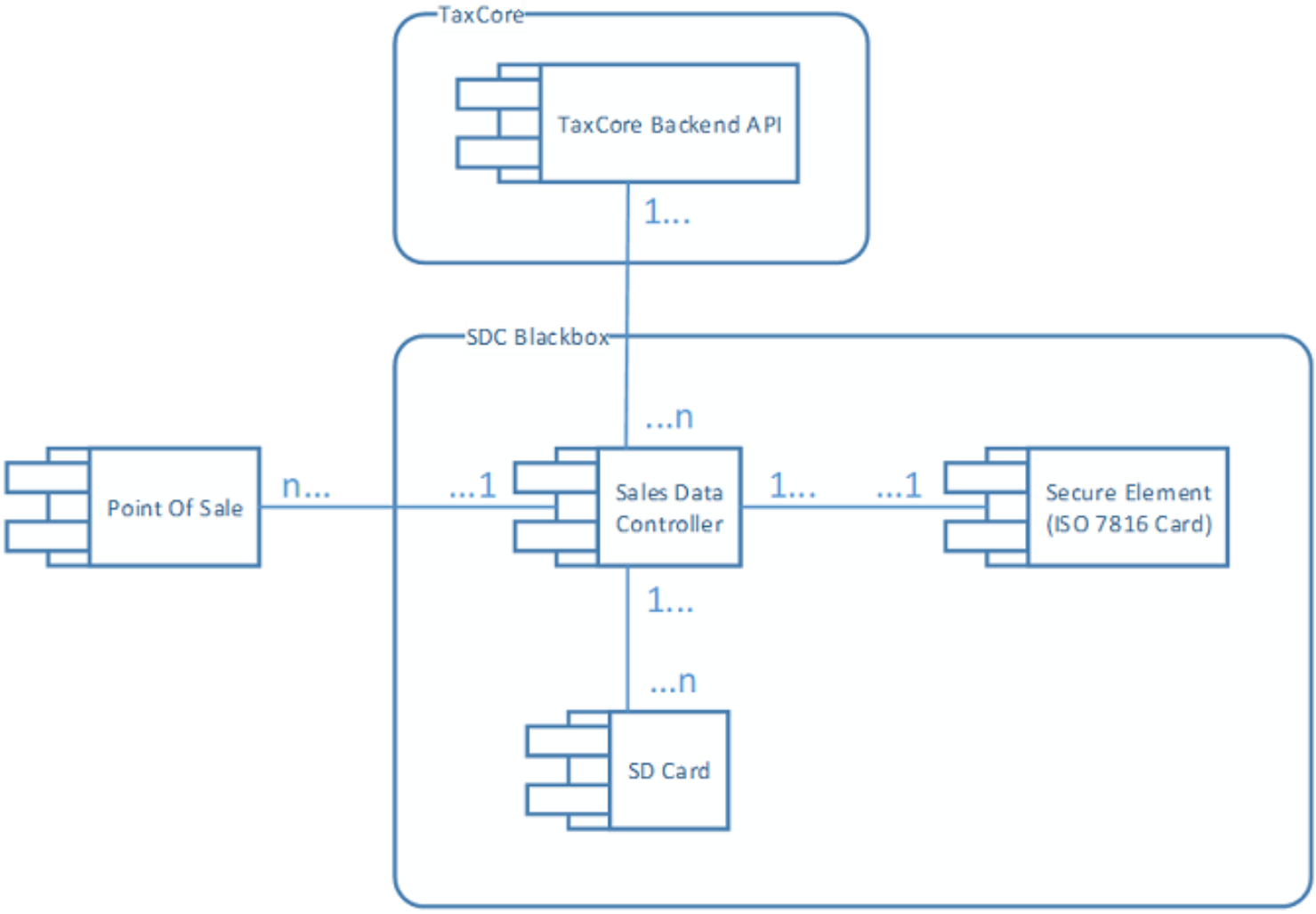
Content

1. E SDC Implementation
   E-SDC can be implemented as a hardware device or a software service depending on the E-SDC manufacturer’s decision and clients’ infrastructure. E-SDC can also be implemented as an integral part of a POS.

2. Smart Cards
   For standard operations, each E-SDC requires a Smart card issued by a tax authority, which consists of two applets:

3. E SDC States
   The diagram below shows the basic E-SDC states and transitions.

This figure shows the components of a fiscalization system and their mutual relationship.



ESDC Architecture – Image showing the components of a fiscalization system and their mutual relationship

E-SDC Implementation

E-SDC can be implemented as a hardware device or a software service depending on the E-SDC manufacturer's decision and clients' infrastructure. E-SDC can also be implemented as an integral part of a POS.

In any of those cases, the E-SDC component has to pass the same accreditation process and prove that E-SDC is implemented according to the instructions described in this document.

1. Ports
   Each E-SDC must provide ports or connectivity for the following purposes

2. Standards
   E-SDC device complies with all current local regulations regarding safety usage, electromagnetic compatibility, temperature range and power supply.

3.

5. Power Supply
   It is allowed to use both AC and DC voltage for power supply. In the case of the AC voltage, a device shall work with the frequency range 50-60 Hz. The power supply circuit used by the E-SDC shall be protected with an automatic circuit breaker, suitable for electronic devices (type I).

Ports

Each E-SDC must provide ports or connectivity for the following purposes

Secure Element

E-SDC has a smart card reader in compliance with ISO/IEC 7810 and ISO/IEC 7816 standard. Smart Card reader may be internal to the E-SDC device or external usually connected via USB port with computer running E-SDC software.

The supported Smart Card sizes are 1FF (credit card size) and 2FF (mini SIM card size).

SIM card	Standard reference	Length (mm)	Width (mm)	Thickness (mm)	Volume (mm^3)
Full-size (1FF)	ISO/IEC 7810:2003, ID-1	85.60	53.98	0.76	3511.72
Mini-SIM (2FF)	ISO/IEC 7810:2003, ID-000	25.00	15.00	0.76	285.00

Audit

If the E-SDC device uses a USB flash drive for a Local Audit, USB connectors "USB Type B female" must be used.

Applied USB communication protocol shall be “USB 2.0” or higher.

For situations when an SD Flash memory card is used for a Local audit, a device must have an easily accessible Micro SD card connector.

For a Remote Audit, these Instructions do not limit a manufacturer in choosing a communication port as long as the invoice signing is not interrupted.

POS Connectivity

POS must be able to connect to the E-SDC using at least one of the following ports:

Ethernet

Ethernet port in compliance with IEEE 802.3 standard, present on an E-SDC device. The minimum speed of the Ethernet port is at least 10 Mb/s.

Wireless

Wireless connection in compliance with IEEE 802.11 (Wi-Fi/Bluetooth) to a POS device and a local network.

Standards

E-SDC device complies with all current local regulations regarding safety usage, electromagnetic compatibility, temperature range and power supply.

Particularly, in terms of electromagnetic compatibility, a product shall comply with the following standards

- EN 55022:2010
- EN 55032:2012
- EN 55024:2010 A1:2015.

A device's operational temperature range shall be 0° - 70° C (Commercial range).

Power Supply

It is allowed to use both AC and DC voltage for power supply. In the case of the AC voltage, a device shall work with the frequency range 50-60 Hz. The power supply circuit used by the E-SDC shall be protected with an automatic circuit breaker, suitable for electronic devices (type I).

In order to protect sensitive electronic components, the smart card and data stored on non-volatile memory, besides the mandatory basic protection, it is recommended that device is equipped with the additional fast overcurrent protection in the form of a fast fuse, e-fuse or similar device with the short time of operation.

When the DC voltage is used, protection against the reverse polarity shall be applied.

If the power supply voltages are higher than 75 Vdc or 50 Vac, a manufacturer shall obtain the appropriate certificate from a local authority, or represent a certificate valid in the country of use.

Smart Cards

For standard operations, each E-SDC requires a Smart card issued by a tax authority, which consists of two applets:

- Secure element Applet - used to apply a digital signature and maintain a set of fiscal counters in the offline mode
- PKI Applet - used to authenticate and establish a secure connection with the TaxCore.API web service

Both applets share the same PIN code. PIN is chosen during the process of requesting a secure element - see Requesting Additional Certificates.

PIN can't be changed after it is selected.

Each smart card is uniquely identified by a UID - Unique Identifier. Each digital certificate issued for E-SDCs has UID embedded in the certificate's subject field.

1. Secure Element Applet
   Secure element (SE) is a fiscal component implemented as a special software or a device designed to receive invoice data, perform signing and data processing and generate a response which is sent back to the caller for further actions.

2. PKI Applet
   PKI Applet contains a digital certificate and a private key used to authenticate E-SDC to TaxCore.API web services.

Secure Element Applet

Introduction

Secure element (SE) is a fiscal component implemented as a special software or a device designed to receive invoice data, perform signing and data processing and generate a response which is sent back to the caller for further actions.

This response provides the authenticity of invoice data.

Secure Element is issued and controlled by the tax authority. The main purpose of the SE is to sign invoices using a taxpayer's digital certificate, control audits and maintain a set of fiscal counters.

Each taxpayer is uniquely identified using digital certificates based on the Public Key Infrastructure (PKI).

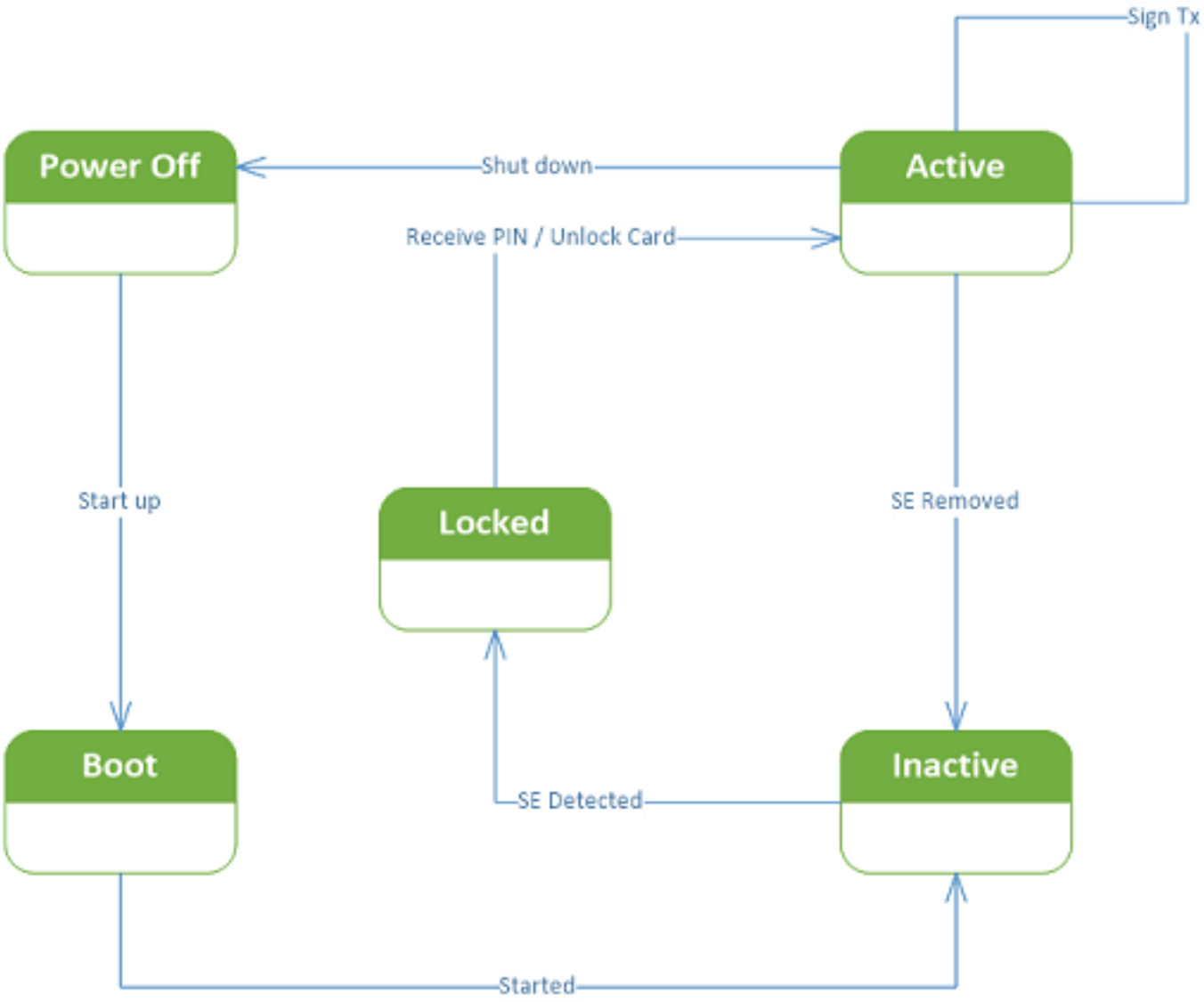
Secure element will stop issuing invoices if the maximum allowed amount for that particular fiscal device is exceeded – this facilitates the regular audit and forces taxpayers to report back to the tax authority system. Likewise, the SE will continue to produce fiscal invoices once it receives proof from TaxCore.API that audit has been received and stored on the tax authority's system.

PKI Applet

PKI Applet contains a digital certificate and a private key used to authenticate E-SDC to TaxCore.API web services.

E-SDC States

The diagram below shows the basic E-SDC states and transitions.



E-SDC States – Image showing the basic E-SDC states and transitions

Data Structures

This section contains descriptions of the main data structures used during the fiscalization and audit processes.

1. Date and Time
   E-SDC has access to the current time. Real-Time Clock or a similar component must be installed and used to maintain the correct time while the power is off. In case of a power outage, the Real-Time Clock must be able to operate without interruption for up to 6 months.

2. Audits
   Audit data represent a machine-readable formatted fiscal invoice signed by a taxpayer's private key followed by journal data. Journal data is a textual representation of a fiscal invoice generated by E-SDC.

3. Commands
   Commands are a means of communication between the tax authority's system and E-SDCs. Commands are stacked in the queue list on the server for a specific E-SDC and submitted to the E-SDC as a part of the response once it reports to the tax authority's system using a Remote or a Local audit.

Date and Time

E-SDC has access to the current time. Real-Time Clock or a similar component must be installed and used to maintain the correct time while the power is off. In case of a power outage, the Real-Time Clock must be able to operate without interruption for up to 6 months.

E-SDC synchronizes the time with the NTP Server, configured by the *Configure Time Server URL Command*.

UTC standard is used only in the `sdcDateTime` value when creating an audit package to be sent to TaxCore.API.

Local time is used by E-SDC for all other purposes, such as:

- Date and time sent by a POS to an E-SDC
- Date and time printed on a journal (textual representation of an invoice) generated by an E-SDC
- Date and time sent by an E-SDC to a POS

JSON-based protocols use date and time according to ISO 8601 where applicable. For example:

Date: **2019-12-11**

Date and time in UTC:

2019-12-11T10:06:33+00:00

2019-12-11T10:06:33Z

20191211T100633Z

Week:

2019-W50

Date with week number:

2019-W50-3

Date without year:

--12-11[1]

Ordinal date:

2019-345

Time offsets are represented in format "**LocalTime+/-Zone**" as in case of **2019-12-11T10:06:33+00:00**

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

All binary-based protocols E-SDC to Secure element - APDU commands use Unix Timestamp format formatted as a 64bit unsigned integer Big Endian (for example: 1495018011910 is 2017-05-17T10:46:51.910Z).

NOTE:
Be mindful of the minimum (1900-01-01T00:00:01Z) and maximum (9999-12-31T23:59:59Z) values.
Also, individual tax jurisdictions might have specific requirements regarding the maximum difference between the time of invoice creation (SDC time) and the moment when an audit package arrives to СУФ. Make sure you are familiar with those requirements.

Audits

Audit data represent a machine-readable formatted fiscal invoice signed by a taxpayer's private key followed by journal data. Journal data is a textual representation of a fiscal invoice generated by E-SDC.

Content of audit data is kept in encrypted form (audit package) ensuring no changes have been made and that no one has been able to access its content after creation, except the tax authority's system, after a successful audit process.

1. Format of Audit Data
   Invoice created as described in section Create Invoice represents audit data. Audit data is a JSON representation of an invoice that consists of two separate data structures:

2. Format of the Audit Proof Request
   The audit-proof request is a textual file in JSON format:

3. Format of the Audit Package
   A file containing encrypted audit data is called an Audit Package. Encryption of audit data prevents access to sales data by unauthorized persons.

Format of Audit Data

Invoice created as described in section Create Invoice represents audit data. Audit data is a JSON representation of an invoice that consists of two separate data structures:

`InvoiceRequest` – this object is created by POS and submitted to E-SDC

`InvoiceResponse` – this object is created by E-SDC and returned to POS

Model

This Model is designed and based on OpenAPI-Specification V2 (https://github.com/OAI/OpenAPI-Specification).

```
Invoice {
    request (InvoiceRequest),
    result (InvoiceResult)
}
```

Example

NOTE:
All field values in the example are for informative purposes only.

```
{
```

```
{
  "request": {
    "dateAndTimeOfIssue": "2021-09-02T14:45:59",
    "cashier": "Software QA Engineer",
    "buyerId": "DT12345",
    "buyerCostCenterId": "Data Tech",
    "invoiceType": 0,
    "transactionType": 1,
    "payment": [
      {
        "amount": 5.0,
        "paymentType": 2
      },
      {
        "amount": 20.0,
        "paymentType": 1
      }
    ],
    "invoiceNumber": "07-2021-3",
    "referentDocumentNumber": "BQVWAAR4-BQVWAAR4-23",
    "referentDocumentDT": "2021-09-02T13:45:29",
    "options": {
      "omitTextualRepresentation": "0",
      "omitQRCodeGen": "1"
    },
    "items": [
      {
        "gtin": "987654321",
        "name": "Mate 20 Lite",
        "quantity": 2.0,
        "discount": null,
        "labels": [
          "A",
          "B"
        ],
        "unitPrice": 10.0,
        "totalAmount": 20.0
      },
      {
        "gtin": "987654321",
        "name": "Credit",
        "quantity": 1.0,
        "discount": null,
        "labels": [
          "A"
        ],
        "unitPrice": 5.0,
        "totalAmount": 5.0
      }
    ]
  },
  "result": {
    "requestedBy": "BQVWAAR4",
    "sdcDateTime": "2021-09-02T12:46:00.8266132Z",
    "invoiceCounter": "4/27NR",
    "invoiceCounterExtension": "NR",
    "invoiceNumber": "BQVWAAR4-BQVWAAR4-27",
    "taxItems": [
      {
        "categoryType": 0,
        "label": "A",
        "amount": 2.0642,
        "rate": 9.0,
        "categoryName": "VAT"
      },
      {
        "categoryType": 0,
```

```
□□□□"label": "B",
□□□□"amount": 0.0,
□□□□"rate": 0.0,
□□□□"categoryName": "VAT"
□□□}
□□],
□□"verificationUrl": "https://ft8.test.taxcore.dti.rs/v/?vl=A0JRVldBQVI0QlFWV0FBUjQbAAAABAAA
□□"verificationQRCode": "R0lGODlhlAGUAfcAAAAAAAAAMwAAZgAAmQAAzAAA/wArAAArMwArZgArmQArzAAr/wB
□□"journal": "============ FISCAL INVOICE ============\r\nTIN:                                RS
□□"messages": "Success",
□□"signedBy": "BQVWAAR4",
□□"encryptedInternalData": "PtYAAizJaZutdu8gViJuVkh13X2xiYUoSM8APq7Il5/a/Ubtm8Y1aMt+MBG1b5Mpa
□□"signature": "AglDF5QXeQN0sbffN+TiDNVKq3ySAr9KGe2ohOzROfJOHgQ9T68dv+VjA6arNY40gKuLUE8BjqVO
□□"totalCounter": 27,
□□"transactionTypeCounter": 4,
□□"totalAmount": 25.0,
□□"taxGroupRevision": 5,
□□"businessName": "Premier League DTI",
□□"tin": "RS654321",
□□"locationName": "Premier League DTI",
□□"address": "Kruzni put 7",
□□"district": "Lestane",
□□"mrc": "0100-1002-BQVWAAR4"
□}
}
```


Format of the Audit-Proof Request

The audit-proof request is a textual file in JSON format:

```
ProofOfAuditRequest {
auditRequestPayload (string),
sum (integer) 64bit unsigned,
limit (integer) 64bit unsigned
}
```

Field	Type	Description
auditRequestPayload	Base64 Encoded String	Byte array obtained from the secure element operation Start Audit, encoded as Base64 string
sum	64bit unsigned integer	Sum of SALE and REFUND of Secure Element, as per Amount Status command in Fiscalization
limit	64bit unsigned integer	The Limit Amount of the Secure Element, as per Amount Status command in Fiscalization

Format of the Audit Package

A file containing encrypted audit data is called an Audit Package. Encryption of audit data prevents access to sales

data by unauthorized persons.

Only the tax authority's system software running on the premise, and used by authorized personnel, can decrypt audit packages. A one-time symmetric key is newly created for each audit package.

The Audit Package is a textual file in JSON format:

```
AuditPackage {
key (string),
iv (string),
payload (string)
}
```

Field	Type	Description
key	Base64 Encoded String	One-time symmetric key (256Bit long) encrypted using RSA with TaxCore public key, obtained from a secure element using APDU command Export TaxCore Public Key
iv	Base64 Encoded String	Initialization vector Key encrypted using RSA and TaxCore public key, obtained from a secure element using APDU command Export TaxCore Public Key
payload	Base64 Encoded String	Audit data encrypted with **key** and **iv** using AES256 algorithm (CBC CypherMode and PKCS7 PaddingScheme). It is a Base64Encoded JSON format of an invoice, as described in section Format of the Audit Package.

Commands

Introduction

Commands are a means of communication between the tax authority's system and E-SDCs. Commands are stacked in the queue list on the server for a specific E-SDC and submitted to the E-SDC as a part of the response once it reports to the tax authority's system using a Remote or a Local audit.

Commands are always delivered as an array structure. Commands are executed in consecutive order, starting from the first one in the array.

Below is the data structure of a single command:

```
Command {
      "commandId": (GUID),
      "type": (enum CommandsType),
      "payload": (Json string),
       "uid": (string)
    }
```

```
enum CommandsType
 {
     SetTaxRates = 0,
     SetTimeServerUrl = 1,
     SetVerificationUrl = 2,
     TaxCorePublicKey = 4 // deprecated,
     ForwardProofOfAudit = 5,
     SetTaxCoreConfiguration = 7
     ForwardSecureElementDirective = 8
 }
```

commandId is a unique identifier assigned by tax authority's system. Once a command is successfully executed, TaxCore.API shall be notified as described in the section Notify Command Processed.

type defines the type of command and a format of a Payload as described in the following sections. Valid values are defined by CommandsType enum.

Payload transfers the information form TaxCore.API to an E-SDC in JSON format. The format of each type is described in the following sections.

Commands Execution Results

Commands Execution Results are returned to TaxCore.API after each command is executed. It is a confirmation to TaxCore.API that a certain command is executed. This structure is used only for file-based communication, when E-SDC indirectly (e.g. via TAP) reports commands execution status for multiple commands.

Model

```
[
    {
        "commandId": (GUID),
        "success": (boolean),
        "dateAndTime": (string)
    }
]
```

Example

```
[
    {
        "commandId": "945bb863-5c7f-4826-9ae3-26debcac331a",
        "success": true,
        "dateAndTime": "2017-06-17T04:33:47+00:00"
    }
]
```

Command Types

0 - Set Tax Rates Command

1 - Set Time Server URL Command

2 - Set Verification URL Command

5 - Forward Proof of Audit Command

7 - Set TaxCore Configuration Command

8 - Forward Secure Element Directive

Set Time Server URL Command

E-SDC updates the URL of the time server used to keep a local clock in sync. Payload is the URL of the target NTP server.

NOTE:
This command is an *initialization* command during E-SDC's first communication with the Tax Service system. After that, it is an *update* command for every next communication.

Example

0.europe.pool.ntp.org

Set Tax Rates Command

Payload contains a cumulative array of all tax rates ever defined by the Tax Authority, along with their effective date and time.

NOTE:
This command is an *initialization* command during E-SDC's first communication with the Tax Service system. After that, the same command is used to *update* the existing tax rates.

The structure of a group is defined in section Tax Rates. The effective date can be in the past or in the future.

Tax rates with the future date should be stored in non-volatile memory and applied to start from their effective

date. If more than one group has the same date, the one with higher revision (`groupId`) should be applied. This means that the next revision with the same effective date replaces the previous.

The payload of the command is structured as follows:

```
[
  {
    "validFrom": "2022-01-15T06:27:58Z",
    "groupId": 20,
    "taxCategories": [
      {
        "name": "ECAL",
        "categoryType": 0,
        "taxRates": [
          {
            "rate": 10.0,
            "label": "F"
          }
        ],
        "orderId": 5
      },
      {
        "name": "PB",
        "categoryType": 2,
        "taxRates": [
          {
            "rate": 0.1,
            "label": "P"
          }
        ],
        "orderId": 6
      },
      {
        "name": "STT",
        "categoryType": 0,
        "taxRates": [
          {
            "rate": 1.0,
            "label": "E"
          }
        ],
        "orderId": 2
      },
      {
        "name": "VAT",
        "categoryType": 0,
        "taxRates": [
          {
            "rate": 9.0,
            "label": "A"
          }
        ],
        "orderId": 1
      }
    ]
  },
  {
    "validFrom": "2021-09-30T22:00:00Z",
    "groupId": 21,
    "taxCategories": [
      {
        "name": "ECAL",
        "categoryType": 0,
        "taxRates": [
          {
```

```
□□□□□□"rate": 10.0,
□□□□□□"label": "F"
□□□□□}
□□□□],
□□□□"orderId": 5
□□□},
□□□{
□□□□"name": "PB",
□□□□"categoryType": 2,
□□□□"taxRates": [
□□□□□{
□□□□□□"rate": 0.1,
□□□□□□"label": "P"
□□□□□}
□□□□],
□□□□"orderId": 6
□□□},
□□□{
□□□□"name": "STT",
□□□□"categoryType": 0,
□□□□"taxRates": [
□□□□□{
□□□□□□"rate": 1.0,
□□□□□□"label": "E"
□□□□□}
□□□□],
□□□□"orderId": 2
□□□},
□□□{
□□□□"name": "VAT",
□□□□"categoryType": 0,
□□□□"taxRates": [
□□□□□{
□□□□□□"rate": 9.0,
□□□□□□"label": "A"
□□□□□}
□□□□],
□□□□"orderId": 1
□□□}
□□]
□}
]
```


Set Verification URL Command

As a part of the invoice fiscalization, an E-SDC creates a unique URL for generating a QR code and validates an invoice. Verification URL is returned to the POS as a part of the Response. This command tells the E-SDC which URL will be used to Create Verification URL.

Payload is a URL of the server used to verify invoices. Detailed instructions for Verification URL creation are explained in Create Verification URL.

NOTE:
This command is an *initialization* command during E-SDC's first communication with the Tax Service system. After that, it is an *update* command for every next communication.

Example

https://sandbox.taxcore.online/v/?vl=

Forward Proof of Audit Command

Introduction

After the audit is executed successfully on the tax authority's system an E-SDC receives the proof-of-audit command. The E-SDC loads proof of audit and verifies if the format is valid. If the format is valid, proof of audit is sent to the Secure Element on the smart card (End Audit APDU command).

The payload is a byte array encoded as a base64 string.

If the format is invalid or the E-SDC and the Secure Element cannot process proof of audit for any reason, the E-SDC signals error message to the operator.

Example

```
0x0CEC878B2B7771B68D6D13EA3C1695A7EBCE395A7777940466EABB36ECA725300EE63E5BCF4CBDFC7C
```

Set TaxCore Configuration Command

This command's payload contains information about environment configuration. E-SDC stores this information locally and provides it to POS on demand. The same parameters can also be obtained from TaxCore.API. Payload is JSON structure as defined below.

NOTE:
This command is an *initialization* command during E-SDC's first communication with the Tax Service system. After that, it is an *update* command for every next communication.

Field	Description
organizationName	Name of the tax authority organization

serverTimeZone	Time zone of the server location
street	Tax authority street address
city	Tax authority city
country	Tax authority country
endpoints	List of available endpoints
environmentName	Name of the environment
logo	Link to the tax authority's official logo
ntpServer	NTP server used for time synchronization
supportedLanguages	List of language-culture strings supported by TaxCore

Example


```
{
  "organizationName": "Data Tech International",
  "serverTimeZone": "Fiji Standard Time",
  "street": "Kruzni put 7",
  "city": "Belgrade",
  "country": "RS",
  "endpoints": {
    "taxpayerAdminPortal": "https://tap.sandbox.taxcore.dti.rs:443/",
    "taxCoreApi": "https://api.sandbox.taxcore.dti.rs:443/",
    "vsdc": "https://vsdc.sandbox.taxcore.dti.rs:443/",
    "root": "https://frontendui.sandbox.taxcore.dti.rs:443/v/?vl="
  },
  "environmentName": "SANDBOX",
  "logo": "https://i.imgur.com:443/tg5ad6O.png?hash=59421698",
  "ntpServer": "http://0.europe.pool.ntp.org:80/",
  "supportedLanguages": [
    "en-US",
    "sr-Cyrl-RS"
  ]
}
```


Forward Secure Element Directive Command

Introduction

Secure Element Directive payload is transmitted to the Secure Element Applet on the smart card using (Secure Element APDU command).

The payload is a byte array encoded as a base64 string.

Example

```
Xb/JzQSvncdsUPo/9U0y0ZELDS4exa+X6uPnGnQjzOBm1uJkJVg4wdut5hicqmwv82Laxuu9OiunaPpEM7R9
```

Processes

This section describes processes performed by an E-SDC.

It contains the following sections:

1. E SDC Initialization
   Prior to the first use, the E-SDC has to be initialized. E-SDC must have access to the Secure Element during the initialization process in order to establish a secure connection with the TaxCore.API to obtain a set of initialization commands. The initialization commands are explained in the section Commands.

2. Standard Operation
   This section contains a description of standard E-SDC operations.

3. Malfunctions and Non serviceable Devices
   If the Secure Element is damaged and its data cannot be restored from the card, but the E-SDC is operational, the tax authority system shall be able to dump data from the E-SDC device and upload the audit packages using the same application used to upload audit packages submitted by a taxpayer.

E-SDC Initialization

Prior to the first use, the E-SDC has to be initialized. E-SDC must have access to the Secure Element during the initialization process in order to establish a secure connection with the TaxCore.API to obtain a set of initialization commands. The initialization commands are explained in the section Commands.

For instructions on how to download the initialization commands, see Get Initialization Commands.

After processing the received initialization commands, the E-SDC must upload configuration commands results to TaxCore.API.

In case of a poor or no internet connection, configuration commands results can be uploaded via file-based communication as explained in section E-SDC Stores a Command Execution Result to the SD Card or USB Drive.

NOTE:
Initialization of E-SDC has to be performed each time a new smart card is inserted into the reader.

Standard Operation

This section contains a description of standard E-SDC operations.

1. Extracting Expiration Date from Digital Certificate
   Digital certificate exported using the *Export Certificate* APDU command (in DER format) contains the certificate (secure element) expiration date.

2. Extracting Taxpayer Identification Number (TIN) from Digital Certificate-from-Digital-Certificate)
   Digital certificate exported using the *Export Certificate* APDU command (in DER format) contains taxpayer TIN and POS location (Shop or HQ Address that shall appear on the textual representation of the invoice).

3. Extracting UID from Digital Certificate
   Digital certificate exported using the *Export Certificate* APDU command (in DER format) contains the UID of the taxpayer's secure element.

4. Extracting Taxpayer Information from Digital Certificate
   Digital certificate exported using the *Export Certificate* APDU command (in DER format) contains taxpayer TIN, Business Name, Shop Name and POS location (Shop or HQ Address that will appear on the textual representation of the invoice).

5. E SDC Executes Commands
   E-SDC can receive Commands in two ways:

6. Sync Date and Time
   As an E-SDC is the source of date and time for the invoices, it is of the utmost importance to keep the device clock in sync.

7. Enter PIN to Unlock the Secure Element
   Before the Secure Element applet can be used, a valid PIN code must be supplied from the POS using the Ethernet connection. Once the E-SDC receives a PIN code, it will try to execute the *PIN Verify* APDU command.

8. Verify Secure Element Expiration Date and Time

Each Secure Element is valid form 3 to 5 years (varies by Environment and local policies).

9. Fiscalization of an Invoice
   Invoice fiscalization is the main function of an E-SDC. Fiscalization is the process of handling an invoice request from an accredited invoicing system to produce fiscal invoices.

10. Audit Process
    An audit is a process of sequential transferring of audit packages from an E-SDC to the tax authority's system and handling the response generated by the system for the specific device.

11. Notifications
    E-SDC device shall have an appropriate way to show the status of the device, information about the smart card and processes running on the E-SDC.

12. Switching Smart Cards During Operation
    During normal operation, taxpayers/cashiers might switch the smart card they are using for issuing fiscal invoices.

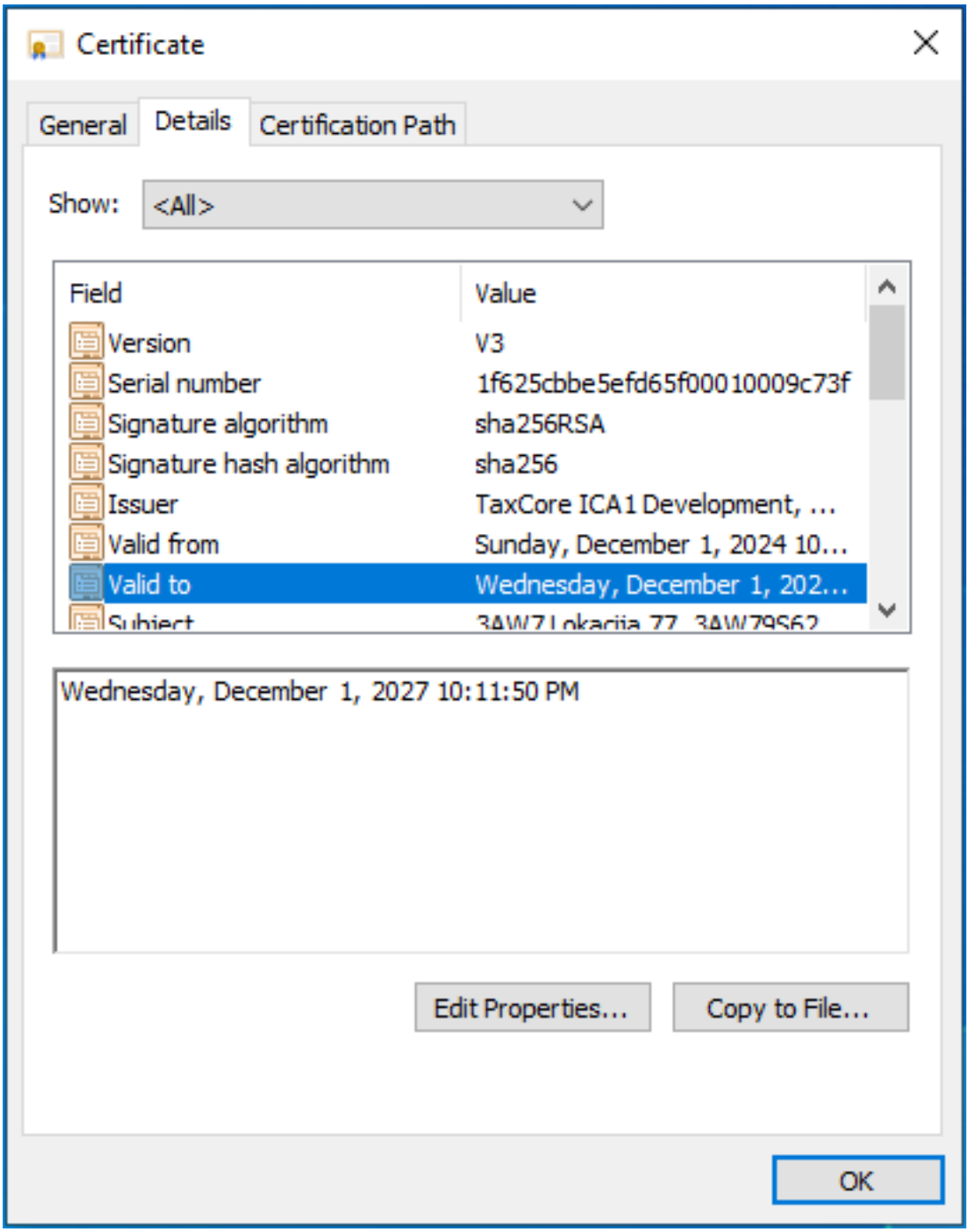
13. E SDC Logging
    E-SDC must keep a log about all required error events. It must log every error chronologically by local date and time (exact hour and minute).

Extracting Expiration Date from Digital Certificate

Digital certificate exported using the *Export Certificate* APDU command (in DER format) contains the certificate (secure element) expiration date.

Once the expiration date is extracted, it should be saved in **E-SDC's volatile memory**, until the smart card is removed/replaced or E-SDC is reset.

The date can be extracted from the certificate property **Valid to**



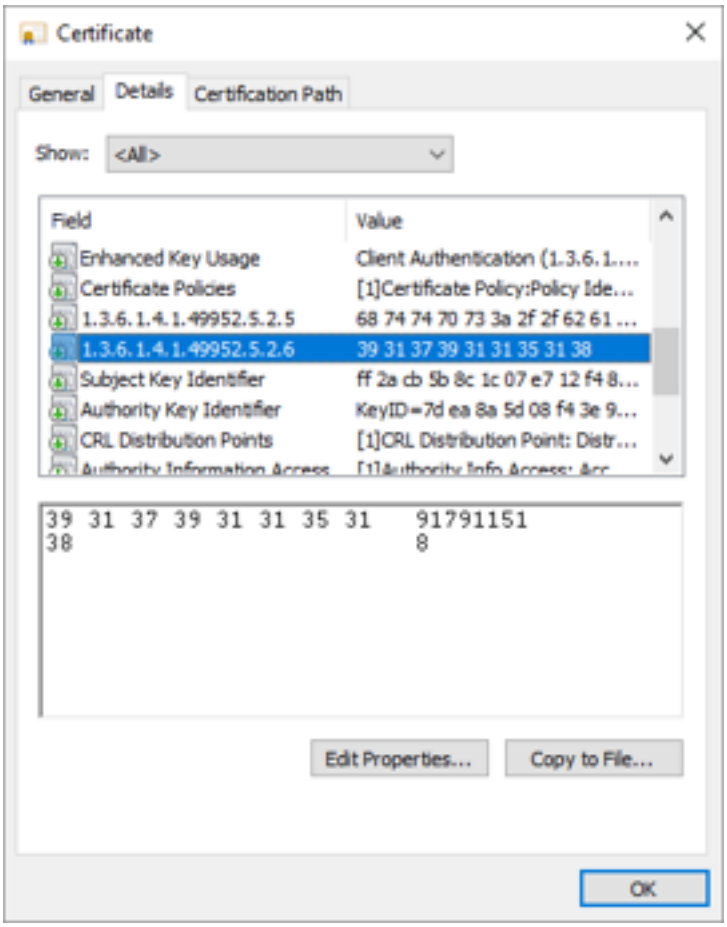
Extracting Taxpayer Identification Number (TIN) from Digital Certificate

Digital certificate exported using the *Export Certificate* APDU command (in DER format) contains taxpayer TIN and POS location (Shop or HQ Address that shall appear on the textual representation of the invoice).

TIN is stored in the digital certificate as an OID value. OID is dynamically created during a smart card personalization and depends on the target environment. The Test and Production environments will have different OIDs.

In order to use the same E-SDC with the Test and Production environments, the correct OID has to be constructed using the following procedure:

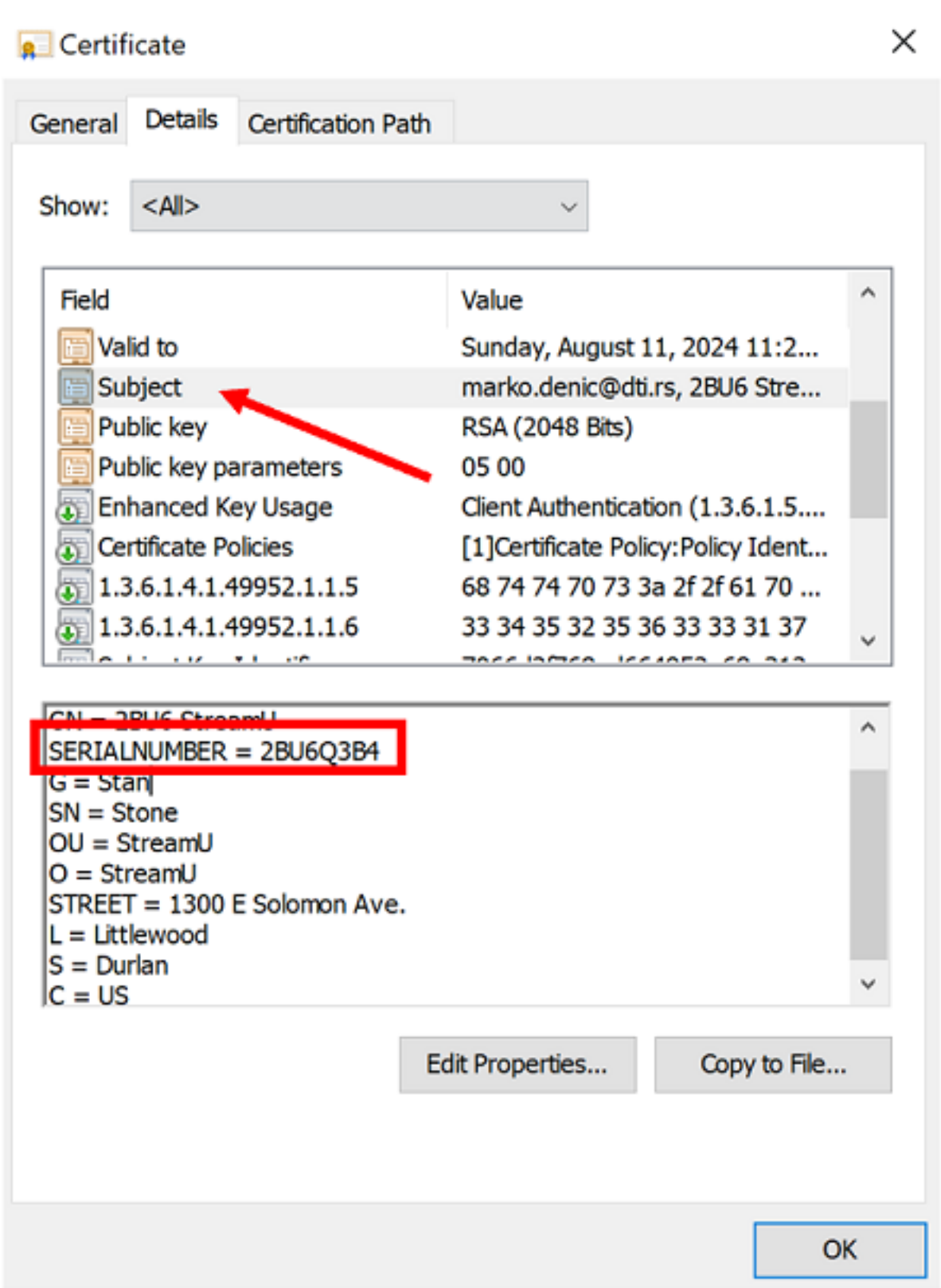
1. Get the certificate using the *Export Certificate* APDU command;
2. Read value of EnhancedKeyUsage (for example, 1.3.6.1.4.1.49952.5.2.3.3);
3. The fourth and the third integer to the right identify the environment;
4. Construct the OID that contains TIN, by replacing stars with the numbers using the following pattern - 1.3.6.1.4.1.49952...6;
5. For this example, resulting OID will be 1.3.6.1.4.1.49952.5.2.6;
6. Read the value of resulting OID containing Taxpayer TIN.



Extracting UID from Digital Certificate

Digital certificate exported using the *Export Certificate* APDU command (in DER format) contains the UID of the taxpayer's secure element.

UID is stored in the digital certificate as an **SERIALNUMBER** value in the certificate **Subject** field.



Extracting Taxpayer Information from Digital Certificate

Digital certificate exported using the *Export Certificate* APDU command (in DER format) contains taxpayer TIN, Business Name, Shop Name and POS location (Shop or HQ Address that will appear on the textual representation of the invoice).

Subject Field of the certificate contains the following information:

CN = Certificate Common Name - first 4 characters of the secure element UID and the Shop name

SERIALNUMBER = UID if the taxpayer's secure element

G = Authorized Person first name

SN = Authorized Person first name

OU = Shop name

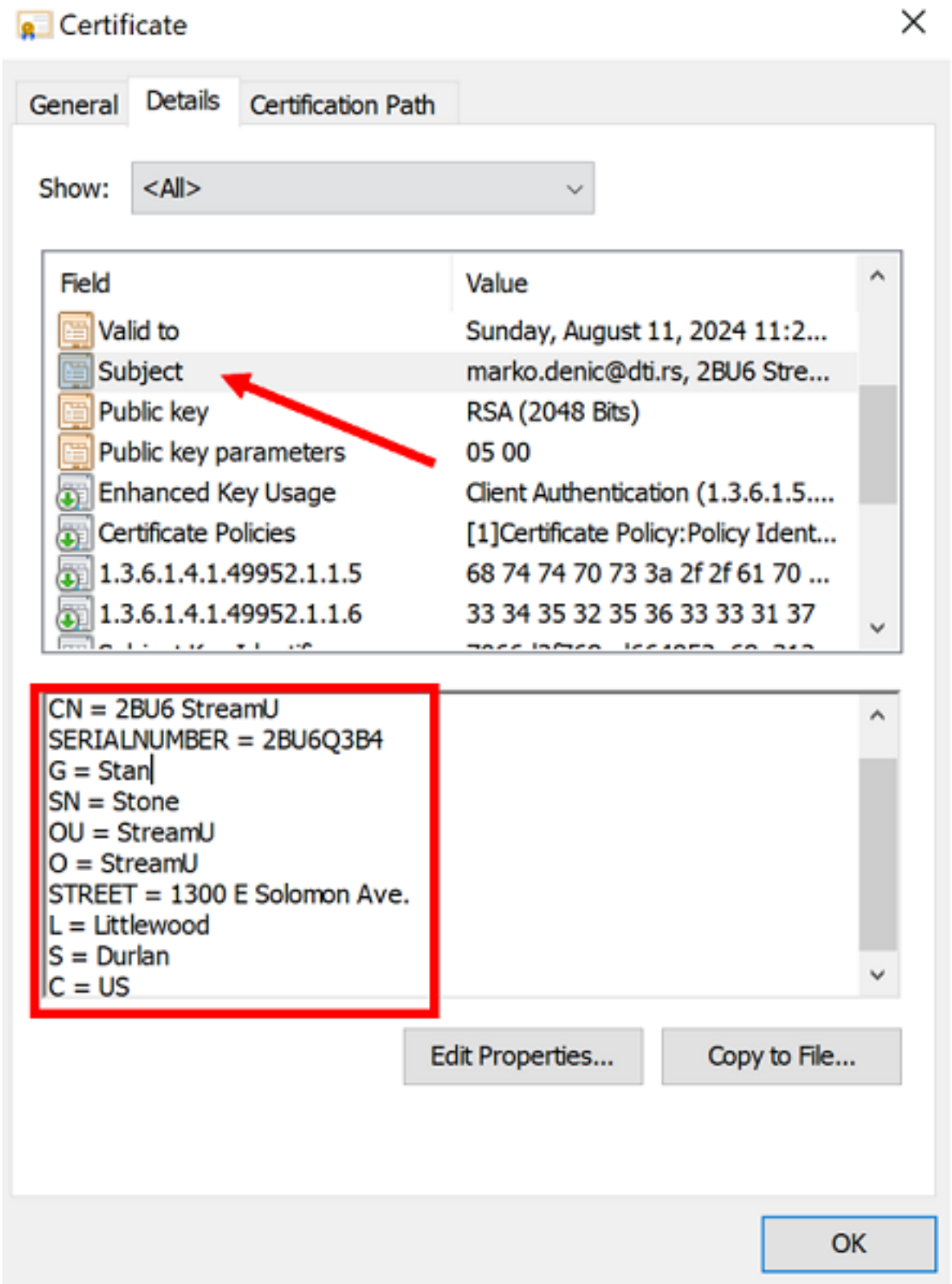
O = Business name

STREET = Physical street address

L = City/town

S = State, District or Region

C = Country



E-SDC Executes Commands

E-SDC can receive Commands in two ways:

- as a response to some TaxCore.API service calls (Notify Online Status and Submit Audit Package)
- via a external storage units

Processing commands depends on the command type. The process of execution for each command is explained

in the Commands section.

After the execution of the commands, in case the E-SDC is online, it notifies TaxCore.API about the execution success as explained in Notify Command Processed.

Sync Date and Time

As an E-SDC is the source of date and time for the invoices, it is of the utmost importance to keep the device clock in sync.

If the internet connection is available, the E-SDC shall sync time with the recommended NTP service at least once every 48h.

If the E-SDC does not support online or semi-connected operation modes, the manufacturer shall provide and document a simple way to check, set and keep date and time in sync on the E-SDC.

Enter PIN to Unlock the Secure Element

Before the Secure Element applet can be used, a valid PIN code must be supplied from the POS using the Ethernet connection. Once the E-SDC receives a PIN code, it will try to execute the *PIN Verify* APDU command.

NOTE:
Depending on the provided PIN, the Secure Element will remain either unlocked for further use or locked until a valid PIN is entered. This means that the E-SDC should not execute the PIN Verify APDU command again as long as the communication session between the E-SDC and the smart card is open.

E-SDC will send a response to the POS based on the result of the PIN Verify command execution.

It is important to note the Secure Element interprets data as byte containing digits, so the E-SDC must perform appropriate conversion before data is sent to the Secure Element. For example, if a PIN transmitted from a POS is “2017” (0x32 0x30 0x31 0x37 in ASCII hexadecimal representation), data sent to the SE shall be 0x02 0x00 0x01 0x07.

Verify Secure Element Expiration Date and Time

Introduction

Each Secure Element is valid form 3 to 5 years (varies by Environment and local policies).

Any invoice fiscalized using expired secure element will be automatically marked as invalid.

How to obtain digital certificate

E-SDC must obtain Secure Element expiration date from digital certificate returned from Secure Element using Export Certificate APDU command in DER format.

This operation must be executed whenever smartcard is inserted into reader or E-SDC is switched on.

What E-SDC must do if digital certificate has expired?

E-SDC must check current date and time against expiration date and time **BEFORE** invoice is signed by Secure Element. If current date and time is greater than expiration date and time of the digital certificate E-SDC must stop fiscalizing any invoices.

Fiscalization of an Invoice

Introduction

Invoice fiscalization is the main function of an E-SDC. Fiscalization is the process of handling an invoice request from an accredited invoicing system to produce fiscal invoices.

Process

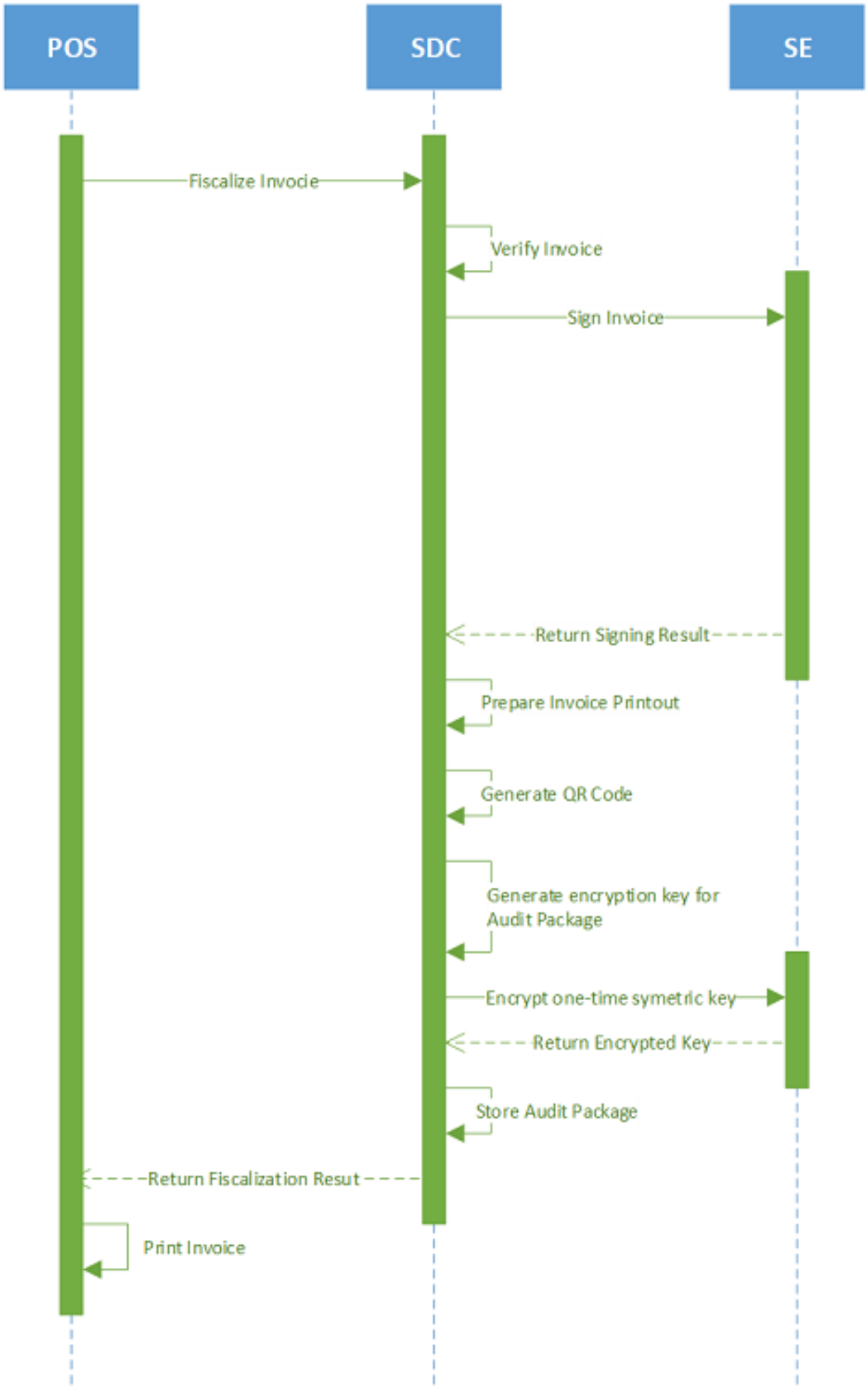
The following steps are executed by the E-SDC once an invoice request data is received from an Accredited POS:

1. POS generates a request data and sends it as an invoice request to the E-SDC;
2. E-SDC verifies the format and content of the invoice request;
3. E-SDC verifies the current E-SDC date and time value is smaller than the smart card certificate expiration date;
4. E-SDC determines which tax rate group to use based on the value of invoice type, ReferentDocumentNumber and ReferentDocumentDT fields;
5. E-SDC calculates taxes based on the selected tax rates group;
6.

E-SDC sends the invoice data to the Secure Element for fiscalization providing the current date and time and PIN code/password if required;

7. Secure element signs the invoice and returns the data to the E-SDC;
8. E-SDC produces a journal – a textual representation of an invoice;
9. E-SDC generates a verification URL;
10. [optionally] E-SDC creates a QR Code – a graphical representation of a verification URL;
11. E-SDC creates an invoice with all mandatory elements (receipt data, previously generated signature, verification URL and journal), generates a one-time key and encrypts the invoice using a symmetric algorithm. The E-SDC encrypts a one-time symmetric key using the tax authority's system public key and adds it to the package so the tax authority's system decrypts the symmetric key and access the package content once it arrives in the Service's system.
12. E-SDC returns a response to the POS and optionally generates journal data.

The process is illustrated in the figure below.



Fiscalization of An Invoice – Image of the fiscalization process

Content

Calculate Taxes

Introduction

Taxes are calculated by an E-SDC after a POS has sent a valid request. The tax amount for particular items on an invoice is 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.

How To Determine Which Tax Rate Group to Use

By default, E-SDC must use current `TaxRateGroup` based on current date and time of its clock to calculate taxes for each invoice.

In case of Copies and Refunds following rules are applied:

- If `referentDocumentNumber` is specified and `referentDocumentDT` is ommited or blank applicable `TaxRateGroup` must be determined based on current date and time of its clock and used to calculate taxes
- If both `referentDocumentDT` and `referentDocumentNumber` are specified, `TaxRateGroup` shall be determined based on value of `referentDocumentDT`

Algorithm

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

1. Determine which Tax Rate Group to use (see above).
2. Make an array of distinct tax labels associated with the items in the POS request (e.g. A, B, C, F, ...).
3. Calculate the tax amount for each individual label in the array:
    - Iterate through all items in the POS request
    - 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 an item's total price. These tax amounts are calculated as follows:
        - ♠ 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 item total amount.
        - ♠ 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%).

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

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

♠

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).

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

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

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

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

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

o

Summarize calculated tax amounts per label for items as the total amount sum for that label.

Example 3: the request contains two items from Example 1 and Example 2, the total sums for labels are: A=$0.8715 , B=$1.0457 , C=$0.2804 and F=$0.3738.

o

Summarize fixed tax amounts per label for items (each multiplied with quantity) as the respective total amount sum for that label.

Example 4: An item has quantity 2 with a total price of $10 and applied labels: A(5% tax-on-net) and E($0.10 fixed tax). The total sums for labels are: A=$0.4667 and 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). The total sums for labels are: A=$0.4531, C=$0.2854 and E=$0.2000.

Example 6: the request contains two items: one with a total price of $5 and quantity 1, and another with a price of $10 and quantity 2. Both have applied label E(0.10$ fixed tax). The total sum for label E=$0.3000.

- Each of these summarized amounts per label represents one tax item in the array `taxItems` in Invoice Response, along with other properties related to the category for this label.

4. After all of 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. These tax amounts for the category are displayed on the invoice journal.

   Example 7: 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 and ET=$0.3738.

5. Once the Tax calculation is completed, assign `GroupId` of the tax rate group to the field `TaxGroupRevision` of `InvoiceResult`.

Rounding

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

Examples:

3.44445555666 → 3.4445

3.4440012345 → 3.4440

3.44466012345 → 3.4447

3.444116012345 → 3.4441

Create Verification URL

Verification URL is created based on values submitted by a POS to an E-SDC and values returned to the E-SDC from APDU commands as follows:

1. Byte array is created:

Start	Bytes	Invoice Field	Is an invoice field	Description
0	1	version	Yes	Current version is 0x03
1	8	requestedBy	Yes	UID, ASCII encoding (e.g. JKGB3K14)
9	8	signedBy	Yes	UID, ASCII encoding (e.g. JKGB3K14)
17	4	totalCounter	Yes	Int32 Little Endian
21	4	transactionTypeCou	Yes	Int32 Little Endian
25	8	totalAmount	Yes	totalInvoiceAmount * 10000 as Uint64 bit Little Endian
33	8	dateAndTime	Yes	Unix Timestamp (number of milliseconds), 64bit unsigned integer Big Endian
41	1	invoiceType	Yes	0x00 (Normal), 0x01 (Pro Forma), 0x02 (Copy), 0x03(Training),0x04(A
42	1	transactionType	Yes	0x00 (Sale), 0x01 (Refund)
43	1	N/A	No	Buyer ID length in bytes
44	0-20	buyerId	Yes	Unicode Encoding
44-64	256 or 512	encryptedInternalD	Yes	Encrypted Internal Data received from SE after Invoice Sign APDU command, 256 or 512 bytes long
300-320 or **556-576**	256	signature	Yes	Signature received from SE after Invoice Sign APDU

				command, 256 bytes long
556-576 or **812-832**	16	N/A	No	MD5 hash of all previous bytes

2. Created byte array is encoded as base64 string, which is additionally encoded, to comply with the URL standards.
3. Encoded string is appended to the verification URL received from the Set Verification URL Command.

NOTE:
Values for `dateAndTime` and all other fields must match the values submitted to the Secure Element for digital signing (see *Sign Invoice* in Fiscalization), as well as the values in the audit package (see Create Invoice) .

Create a QR Code

QR code contains a Verification URL that is described created in the section Create Verification URL.

It is the most convenient way of exposing the Verification URL because it enables customers to easily scan their fiscal invoices using a QR code reader.

How to create a QR code

Base64 encoded string is created from GIF image bytes and attached to the Invoice Response

Important parameters for creating a QR code:

- Minimal size = 40x40mm
- ErrorCorrectionLevel = L
- FixedModuleSize = 4
- QuietZoneModules = Zero
- BlackAndWhite
- ImageFormat = Gif

For more information about using QR codes in the TaxCore system, see QR Code

Create a Textual Representation of an Invoice

A textual representation of a Receipt shall be created as described in the chapter Anatomy of a Fiscal Receipt. One row on a receipt is 40 characters long to fit 2.25 inch / 58 mm wide paper roll commonly used in thermal printers.

SDC Date and Time field printed on a journal (textual representation of an invoice) generated by E-SDC are **locally time-based**.

Any amount shall be rounded to 2 (two) decimal places using the half-round up method only on the textual representation of an invoice.

NOTE:
Although the textual representation of an invoice (journal) can optionally be omitted from the E-SDC's response to POS, it **must be submitted to the tax authority** as part of the audit package.

1. Localization of textual representation of the invoice
   E-SDCs are generally built to work in multiple environments. As part of implementation roadmap you may decide to prepare your product for multiple markets or to target one market only.

2. Mapping Digital Certificate Subject Parameters to Invoice Fields
   Digital certificate exported using the *Export Certificate* APDU command (in DER format) contains taxpayer TIN and POS location (Shop or HQ Address that shall appear on the textual representation of the invoice).

Localization of textual representation of the invoice

Introduction

E-SDCs are generally built to work in multiple environments. As part of implementation roadmap you may decide to prepare your product for multiple markets or to target one market only.

Localization is optional

If you decide to support only one market E-SDC may support only one language. For example, Implementation for Fiji may support English language only.

What to do if you want to support multiple

languages?

- Textual representation must adhere to general instructions outlined in Anatomy of a Fiscal Receipt.
- All Invoice and transaction types abbreviations must be localized. For example **NS** (Normal Sale) in English is localized as **ПП** (Промет Продаја) in Serbian Cyrilic.
- All labels on invoice must be translated
- Languages supported by your E-SDC implementation **AND** TaxCore.API must be contained in the Result of the Get Status service.
    - in case your E-SDC supports en-US, fr-FR and sr-Latn-RS and TaxCore.API supports only fr-FR E-SDC must report fr-FR as the only supoprted language
    - in case your E-SDC supports en-US only and TaxCore.API supports only fr-FR E-SDC must return configuration error and stop any initialization.
- Content generated by POS or Secure Element should not be modified or localized (i.e. item names)
- If your E-SDC and TaxCore.API supports multiple languages POS may decide language of generated textual representation of invoice using request HTTP headers when invoking Create Invoice endpoint.

Accreditation

Accreditation for specific jurisdiction may require E-SDC to support specific language and culture. For Example, E-SDCs for Serbia requires support for sr-Cyrl-RS.

Mapping Digital Certificate Subject Parameters to Invoice Fields

Digital certificate exported using the *Export Certificate* APDU command (in DER format) contains taxpayer TIN and POS location (Shop or HQ Address that shall appear on the textual representation of the invoice).

This example shows the mapping between a subject name/value pairs and invoice fields.

Subject Field parameters with examples:

CN = P22V International Trek Center

SERIALNUMBER = P22VC8VR

G = Albert

SN = Mungin

OU = International Trek Center

O = International Trek Center

STREET = 8844 Garcia

L = West Covina

S = California

C = US

Invoice Field	Subject Parameter Name	Note
TIN	N/A	obtained by OID as explained in Extracting Taxpayer Identification Number from Digital Certificate
Business Name	O	Legal name under which the business operates - see Extracting Taxpayer Information from Digital Certificate
Shop Name	OU	It may be the same as Business Name if the Company HQ and sales location are the same. - see Extracting Taxpayer Information from Digital Certificate
Address	STREET	Street name and number - see Extracting Taxpayer Information from Digital Certificate
Location	L	City or town - see Extracting Taxpayer Information from Digital Certificate
State	S	State, District or Region - see Extracting Taxpayer Information from Digital Certificate
Country	C	ISO 2-letter Country Code. Optional field on the textual representation of the invoice - see Extracting Taxpayer Information from Digital Certificate

Creating an Audit Package

Once an invoice is created (`InvoiceRequest` and `InvoiceResult`) the E-SDC is ready to create an audit package and store it in the non-volatile memory. In order to achieve that, follow these steps:

1. Convert `sdcDateTime` data to UTC;
2. Generate a random one-time symmetric key for AES256;
    - KeySize = 256
    - Padding = PaddingMode.PKCS7
    - Mode = CBC
    - BlockSize = 128
    - IV = 16bytes
    - Key =32bytes
3. Encrypt Audit Data as JSON string using the one-time key;
4. Convert the encrypted invoice to base64 string and store it in the Payload field of Audit package;
5. Get the TaxCore Public key using *Export TaxCore Public Key* APDU command).
6. Encrypt the one-time key, using RSA encryption (padding PKCS1 (fOAEP: false)), with TaxCore public key, convert it to base64 string and store it in the Key field;
7. Encrypt Initialization Vector (IV), using RSA encryption (padding PKCS1 (fOAEP: false)), with TaxCore public key, convert it to base64 string and store it in the IV field;
8. Save the Audits as an Audit Package file, named as `{UID}-{UID}-{Ordinal_Number}.json`;
9. (Optionally) Generate a QR code, and attach it to `InvoiceResult` (make sure that the QR code is not stored in the Audit Package);
10. Return `InvoiceResult` to the POS;
11. If the internet connection is available try to send the Audit Data to TaxCore.API as explained in the section Remote Audit;

NOTE:
After submitting an audit package to TaxCore.API, if status 4 is received back (see Submit Audit package) , the E-SDC should immediately delete that audit package from the its local storage. If the TaxCore.API returns status 1, the E-SDC should try to resubmit an audit package. If any other status is received (other that 1 or 4), the audit package must not be deleted, and the E-SDC should not try to resubmit that audit package to TaxCore.API.

Audit Process

Introduction

An audit is a process of sequential transferring of audit packages from an E-SDC to the tax authority's system and handling the response generated by the system for the specific device.

There are two specific scenarios: **Remote Audit** and **Local Audit**.

NOTE:
Basic rules and processes described in this section apply to both scenarios. Details are explained in separate sections - see Remote Audit and Local Audit.

Depending on the scenario, an audit may be triggered periodically, if one or more invoices are created, or after the insertion of an external memory device into an E-SDC.

An audit is always an asynchronous process. Depending on the amount of data and means of communication, it can take from less than a second to a couple of hours.

Once the E-SDC receives a response from the Secure Element (signed invoice), it must be encrypted and stored in E-SDC's non-volatile memory.

An E-SDC device must be fully functional during an audit. The POS must be able to sign new invoices as long as the Secure Element permits. There must be a mechanism in place that is responsible for the continuous operation of the Secure Element and E-SDC while audit packages are being transmitted to the tax authority's system or an external memory unit.

Remote Audit

Remote audit is the process of transferring data to the tax authority's system using an internet connection. It is the most common way to perform audits for any device with a stable internet connection.

An E-SDC checks if TaxCore.API is reachable. If TaxCore.API is reachable, the E-SDC authenticates the tax authority's system by using a server-side certificate installed on the TaxCore.API endpoint, enabling HTTPS protocol. The tax authority's system authenticates the E-SDC using a digital certificate issued on the Secure Element and issues a token for that session.

The E-SDC starts sending audit packages, performing a series of audits until all the data stored on its non-volatile memory is audited.

A Remote audit is not the only audit option for E-SDC. If the network connection is not available due to the interruption of the service or a missing GPRS modem or network card, E-SDC is able to perform a Local Audit.

Remote audit step-by-step:

1

1. E-SDC submits an audit package by invoking TaxCore.API service Submit Audit Package
2. Tax authority's system verifies the data and returns a response containing:
    - audit package status
    - a set of Commands (including the Proof of Audit command)

Local Audit

Local audit initiated by a taxpayer is a common scenario for devices that lack the ability to connect to the internet due to the technical limitations of the devices or limited infrastructure.

Unlike in Remote Audit process, during the Local Audit, the E-SDC doesn't submit the ARP file and audit packages to TaxCore.API. Instead, those files are copied to an SD Card or a USB Flash Drive.

Local audit step-by-step:

1. An audit is initiated by inserting an SD card or a USB Flash drive into an E-SDC device.
2. E-SDC signals the beginning of the audit to the Secure Element (invokes *Start Audit* APDU command);
3. E-SDC copies audit packages to external memory unit (e.g. SD card, USB flash drive), starting with the oldest unaudited package, in a piecemeal fashion, as described in E-SDC Stores Audit Files on SD Card or USB Drive.

NOTE:
Audit packages remain stored in E-SDC's local memory until it receives a Proof-of-Audit command from TaxCore.API

4. The Secure Element returns ARP (260 bytes) to the E-SDC;

NOTE:
When performing Local Audit, the ARP file should be generated and saved each time when at least one audit package is submitted to the tax authority’s system, by using an external memory unit (see File-based communication).

5. An operator uploads the audit packages and the ARP file to the Tax Authority's system (by using the Taxpayer Administration Portal or the tax authority's Back Office Portal).
6. After the successful upload, a set of Commands (including the Proof of Audit command) are generated by the tax authority's system.

7. An operator downloads the commands to an external memory unit (by using the Taxpayer Administration Portal or the tax authority's Back Office Portal).
8. An operator inserts the external memory unit with the commands file into an E-SDC which processes the commands as described in E-SDC Executes Commands, i.e. passes the payload to the *End Audit* APDU command;

NOTE:
The Proof of Audit command might be missing due to many reasons, such as: not all packages have been submitted, the tax authority system is experiencing a delay in data processing... Perhaps, the Proof of Audit command for a specific local audit might never be returned.

9. The Secure Element resets its current unaudited amount to 0.00 and returns the OK message to the E-SDC;
10. The E-SDC generates the `Commands Execution Results` structure (as described in Commands) and saves it to a file on the external media, as described in E-SDC stores a command execution result to the SD card or USB drive.
11. The operator uploads the `Commands Execution Results` file to the Tax Authority's system (by using the Taxpayer Administration Portal or the tax authority's Back Office Portal).

Proof of Audit

Introduction

Proof-of-Audit (POA) is generated by the tax authority's system once all expected audit packages have been received and securely stored on the tax authority's system.

Even if, due to the failure of the EFD component or some other reason, the taxpayer is unable to send one or more audit packages, the Tax Authority still has the option to issue a Proof-of-Audit if it can determine the tax liability for that secure element.

A POA cycle begins with E-SDC initiating the *Start Audit APDU command* (**Audit Start**). A POA cycle finishes with the Tax Authority's system receiving a confirmation that the secure element has successfully executed the issued Proof-of-Audit command (**Audit End**).

NOTE:
This article describes the default operation mode of the *Proof of Audit Service*. However, upon the Tax Authority's decision, the service can use different strategies for issuing a proof-of-audit. For the currently applicable non-

standard issuance strategies, see Currently Applicable Non-Standard Strategy for Issuing Proof of Audit.

There are two scenarios for obtaining a Proof-of-Audit:

- **As part of the Local Audit process**
- **Via direct communication with TaxCore.API as described below**

1. E-SDC signals the beginning of the audit to the Secure Element (invokes *Start Audit* APDU command);
2. The Secure Element returns ARP (260 bytes) to the E-SDC;
3. E-SDC starts the audit by sending audit data (over HTTPS). ARP is submitted to the tax authority's system using the same communication channel;
4. If the request is correct, the system returns the HTTP status code 200 (OK);
5. Tax authority's system generates a set of Commands (including the Proof of Audit command);
6. E-SDC submits a commands request to the tax authority's system via one of the following channels:
   - ♠ by invoking TaxCore.API service Notify Online Status
   - ♠ by invoking TaxCore.API service Submit Audit Package
7. Tax authority's system returns the commands to the E-SDC as a response;
8. E-SDC processes the commands as described in E-SDC Executes Commands, i.e. passes the payload to the *End Audit* APDU command;
9. The Secure Element resets its current unaudited amount to 0.00 and returns an OK message to the E-SDC;
10. E-SDC reports the results of the commands' execution to TaxCore.API as described in Notify Commands Processed

NOTE:
Regardless of which scenario was used to initiate the POA cycle, once the commands (containing the Proof-of-Audit command) have been generated by the tax authority's system, the cycle can be completed by using the other scenario.

POA cycle frequency

Audit Start should be initiated periodically, and the length of the period between two Audit Starts should be set dynamically. In regular conditions, there is no reason to initiate an Audit more often than every 30 minutes; although this can be extended to a period of a couple of hours as well (depending on the turnover). There is no need to initiate unnecessary Audits if the secure element(s) does not register sale amounts that will cause it to cross its assigned limit.

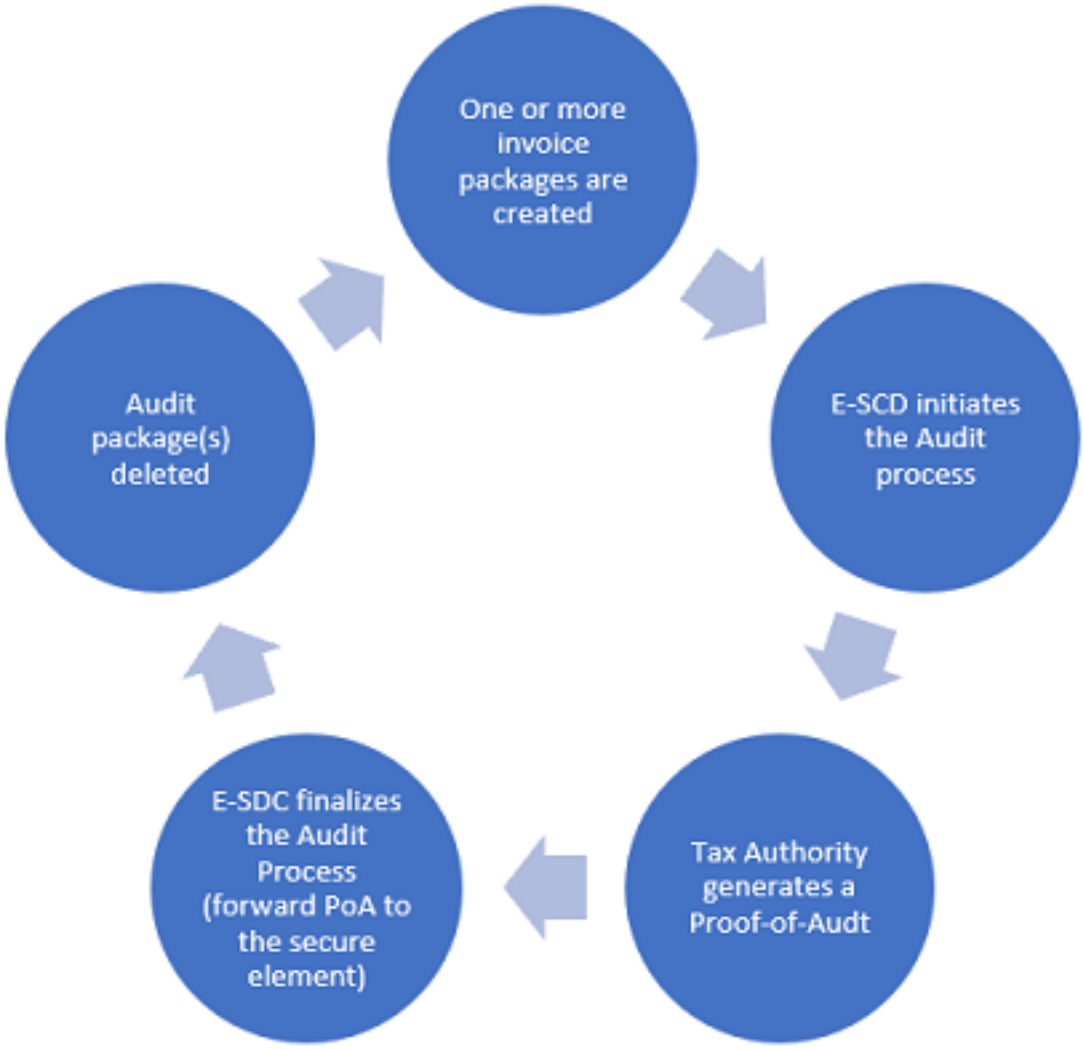
However, if the secure element is reaching its limit, the Audit should be initiated immediately and the period between two Audit Starts should be shortened to 10 minutes (unless there were no new invoices created in that period).

NOTE:
Although the Audit Start should be initiated periodically, audit packages should be submitted continuously, as soon as they are created (regardless of the Audit Start and Audit End).

The СУФ system needs about 10 minutes to process submitted data and generate a Proof-of-Audit command, provided that there are no missing audit packages for that Audit.

Once the Secure Element receives a valid Proof-of-Audit, the E-SDC can delete the audit packages included in that Audit.

NOTE:
E-SDC must store the SDC number of the last invoice created before an Audit Start (the last invoice included in that Audit) so it would know which invoices to delete upon receiving a valid Proof-of-Audit.



Be mindful of these cases

Sometimes, audit packages can arrive in СУФ database after the Start Audit command - in that case, the system

will again need at least 10 minutes after the arrival of the last audit package to generate the End Audit command (**see case 2a below**).

If two Audit Starts are initiated before an Audit End command is generated, the Proof-of-Audit for the first Audit Start will not be valid. Only the next Proof-of-Audit (covering both Audit Starts) will be valid (**see Case 3 below**). Moreover, the Proof-of-Audit for the first Audit Start will be replaced by the next Proof-of-Audit, so E-SDC will receive only the latest Proof-of-Audit. But if E-SDC happens to receive the first Proof-of-Audit, it will be invalid and rejected by the Secure Element.

If one or more audit packages issued by the same secure element (same UID) do not arrive in the database, the End Audit command (Proof-of-Audit) will never be generated.

Audit cycle cases

This means that there are three possible scenarios for completing the Audit cycle:

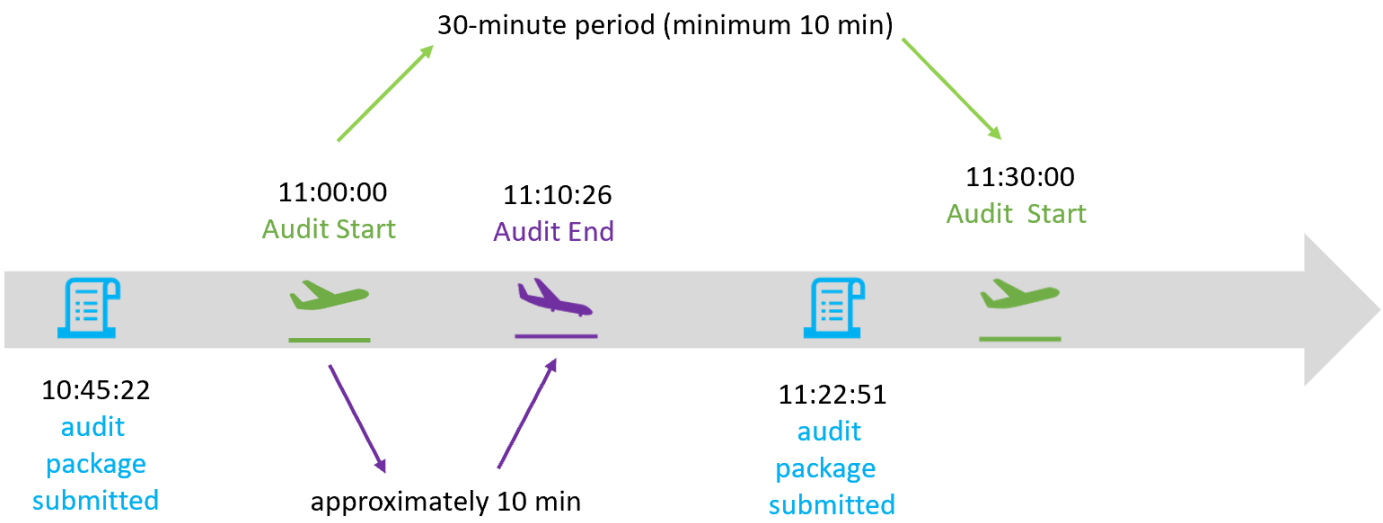
1. One audit package is created and one Audit Start is initiated between completing two Audit Ends (two Proof-of-Audits) - **see Case 1 below**
2. Multiple audit packages are created and one Audit Start is initiated between two Audit Ends (two Proof-of-Audits) - **see Case 2 below**
    - Sometimes, an E-SDC can initiate an Audit Start before submitting all the audit packages from that Audit. In that case, the system will wait for the last audit package from that Audit to arrive before it starts generating the Audit End command (Proof-of-Audit) - **see Case 2a below**
3. Multiple audit packages are created and multiple Audit Starts are initiated between two Audit Ends (two Proof-of-Audits) - **see Case 3 below**

Case 1 – Audit is performed after the creation of each audit package

This is the simplest case, where no additional audit packages are generated during the whole audit process, as follows:

1. Create an audit package
2. Initiate the Audit process by invoking the Start Audit APDU command
3. Receive a proof of audit and pass it to the End Audit APDU command
4. If EndAudit returns the value "true", you can safely delete the audit package(s)
5. If EndAudit returns the value "false", continue until a valid proof of audit is received
6. The period until the next Audit Start must be **at least 10 minutes (recommended from 30 minutes to a couple of hours)** after the previous Audit Start

The figure below illustrates the process:

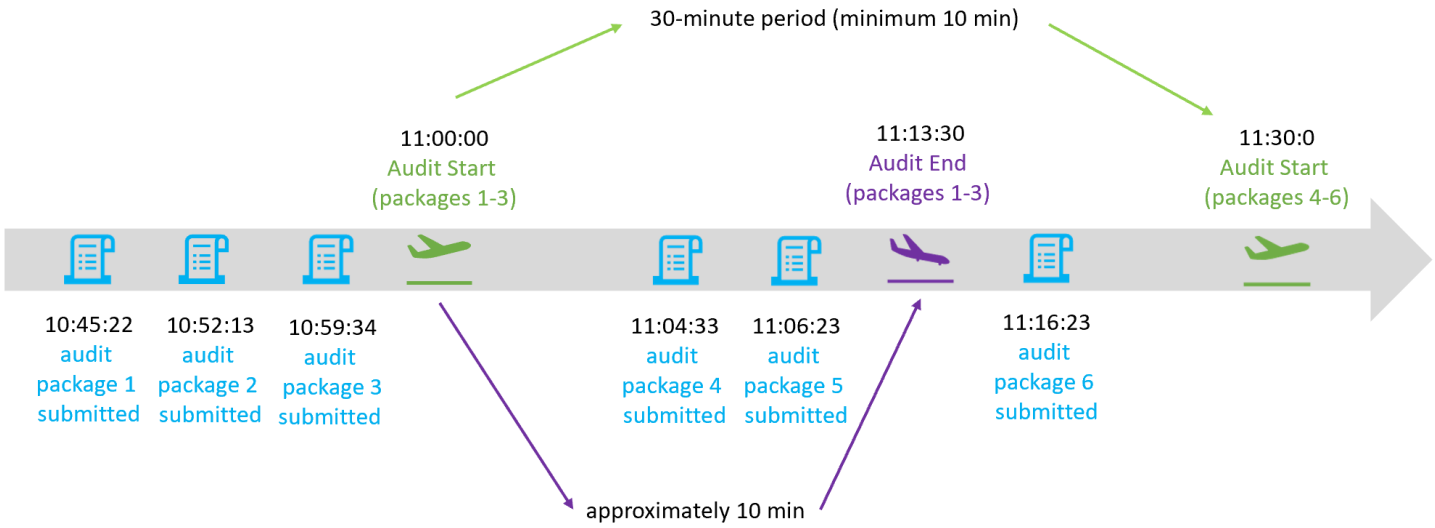


Case 2 – Audit is performed after multiple audit packages have been created

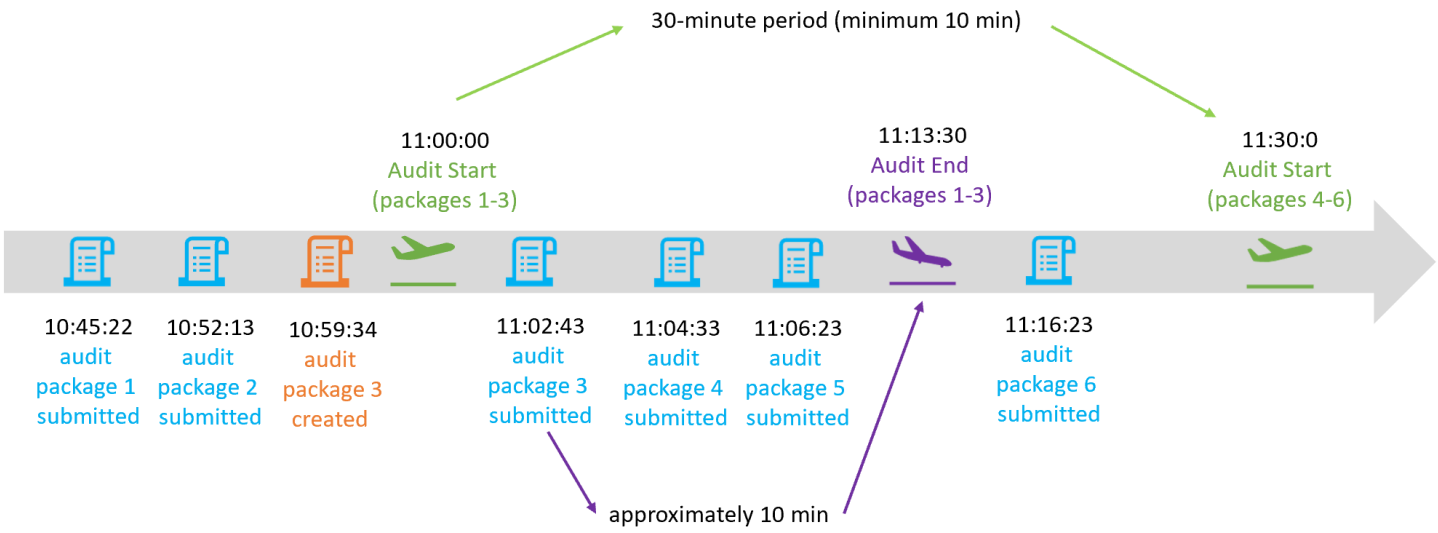
In this case, new packages can be created after an audit start:

1. Create audit packages 1-3 (as shown on the diagram)
2. Initiate the audit process by invoking the Start Audit APDU command
3. Continue to fiscalize invoices and create audit packages 4-6
4. Receive a proof of audit and pass it to the End Audit APDU command
5. If EndAudit APDU command returns value true you can delete remaining audit packages 1-3 because it is the last initial audit being invoked by E-SDC. Audit packages 4-6 are created after the call to BeginAudit APDU command so they are not audited in this cycle
6. If EndAudit APDU command returns value false, continue (return to point 1) until a valid proof of audit is received
7. The period until the next Audit Start must be **at least 10 minutes (recommended from 30 minutes to a couple of hours)** after the previous Audit Start

The figure below illustrates the process:



Case 2a - Audit Start is initiated before all audit packages are submitted

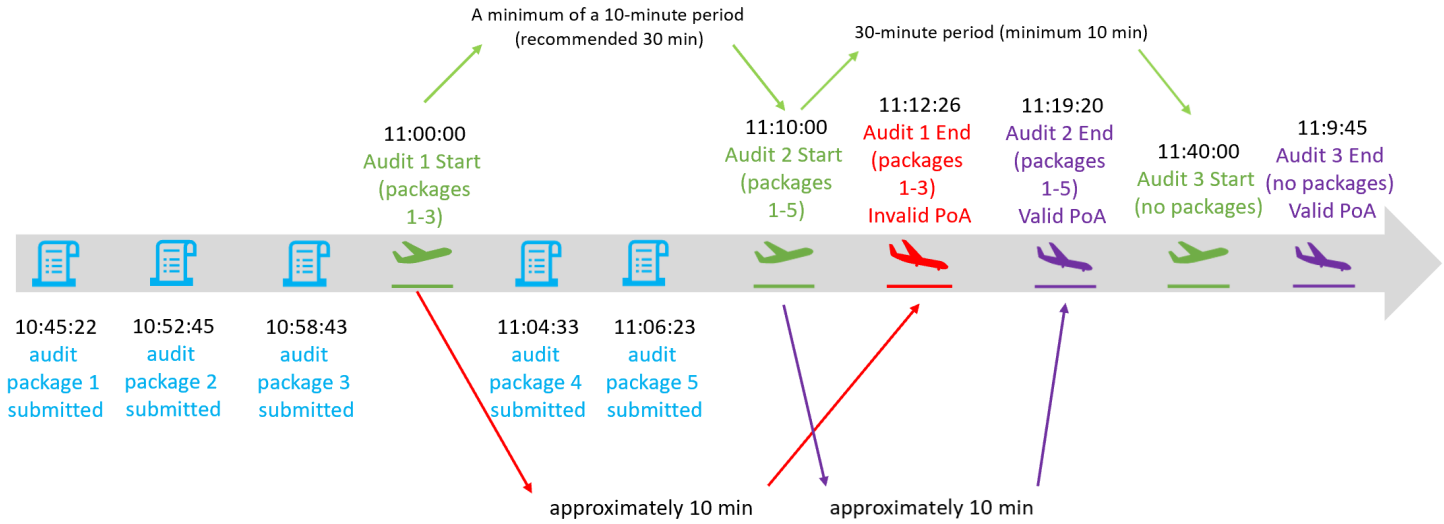


Case 3 – Audit is started multiple times before the Proof-of-Audit is generated

This case involves multiple audit starts:

1. Create Audit Packages 1-3
2. Initiate the Audit process by invoking the Start Audit APDU command
3. Continue to fiscalize invoices and create Audit Packages 4 and 5
4. Initiate another audit process by invoking the Start Audit APDU command – the previous audit is canceled
5. Receive the Proof-of-Audit and pass it to End Audit APDU command
6. If EndAudit returns value true you can delete remaining audit packages 1-5 because it is the last BeginAudit being invoked by E-SDC.
7. If EndAudit APDU command returns value false, continue until a valid Proof-of-Audit is received
8. A Proof-of-Audit generated for the first Audit Start (Audit 1 below) is not considered valid. Only the Proof-of-Audit which is generated for the last Audit Start (Audit 2 below) is considered valid and will be forwarded to the secure element.
9. The period until the next Audit Start must be **at least 10 minutes (recommended from 30 minutes to a couple of hours)** after the previous Audit Start

The figure below illustrates the process:



Notifications

Introduction

E-SDC device shall have an appropriate way to show the status of the device, information about the smart card and processes running on the E-SDC.

A cashier could get the device notifications by receiving an onscreen message, by observing the colors from the light-emitting diodes (LED) or any other similar component set for displaying visual notifications.

Required Notification

The following visual notifications shall be available to a cashier:

1. Smartcard is inserted but the E-SDC is not yet configured with the tax rates, verification URL or NTP service address. This is a common situation before initialization commands are executed by E-SDC;
2. Enter PIN Code for the Secure element – Smart card is inserted but E-SDC has not received the PIN Code from POS;
3. E-SDC is ready to sign an invoice;
4. Smart card is missing or unavailable;
5. Audit package transfer is in progress (Local audit on an SD card or USB flash drive, or an online audit);
6. Firmware update is in progress (if applicable);
7. Audit data storage is almost full;
8. Audit data storage is full;
9. Time for audit;
10. Commands in progress (currently running)

Switching Smart Cards During Operation

During normal operation, taxpayers/cashiers might switch the smart card they are using for issuing fiscal invoices.

In that case, the E-SDC must perform the following activities:

- **if the new smart card is for the same environment** - the E-SDC will first submit the unsubmitted invoices that were created with the previously used smart card
- **if the new smart card is for a different environment** - the E-SDC will keep the unsubmitted invoices (created with the previously used smart card) in its internal memory until they can be submitted (old smart card is returned)

For more information about different environments, see Identification of Environments and Important Endpoints.

E-SDC Logging

Introduction

E-SDC must keep a log about all required error events. It must log every error chronologically by local date and time (exact hour and minute).

E-SDC log must be available for easy export (download, USB flash drive...) and presented in a human-readable format.

Required Logging

The following error events must be logged:

- Any Invoice Request sent by POS that E-SDC failed to process
- Any APDU error returned by SE
- Any error returned by TaxCore
- Any error caused by internal E-SDC operations
- Any error during the E-SDC Initialization process (Enter PIN and Command processing).
- Any error during the Invoice Fiscalization process
- Any error during the Audit process (Local and Remote).
- Any error during the process of Date and Time Synchronization

The above errors are the minimum requirements, but E-SDC can also keep a log of other events.

Malfunctions and Non-serviceable Devices

Dump Audit Packages Kept on E-SDC when Secure element is damaged

If the Secure Element is damaged and its data cannot be restored from the card, but the E-SDC is operational, the tax authority system shall be able to dump data from the E-SDC device and upload the audit packages using the same application used to upload audit packages submitted by a taxpayer.

Protocols

This section describes Application Programming Interfaces (API) and protocols exposed by an E-SDC or used by an E-SDC to communicate with the other components (TaxCore.API, Secure element Applet, PKI Applet or SD Card/USB Flash Drive) required to fulfill its primary role – to safeguard a transaction and to transfer the audit packages to the tax authority's system.

Accredited POS systems can communicate with the E-SDC using the POS to SDC Protocol.

1. TaxCore.API
   TaxCore.API is a REST API exposed by a tax authority's system to E-SDC devices. It provides services used by the E-SDCs to submit Audit Packages, to notify TaxCore if the online status has been changed and to receive configuration commands.

2. Secure Element Applet API
   Communication with a Secure element Applet API is performed through standard APDU commands.

3. File Based Communication
   This section contains the description of the **File-based** communication with E-SDC.

TaxCore.API

TaxCore.API is a REST API exposed by a tax authority's system to E-SDC devices. It provides services used by the E-SDCs to submit Audit Packages, to notify TaxCore if the online status has been changed and to receive configuration commands.

An E-SDC is authenticated by TaxCore.API using a client digital certificate and an authentication Token.

Authentication

Introduction

Communication between a Client and TaxCore.API is carried out via the HTTPS protocol.

The Client is authenticated by TaxCore.API using either a client certificate or an authentication token obtained from TaxCore.API. To obtain an authentication token, a client certificate authentication has to be successfully conducted as the first step. For more information see Request Authentication Token.

Once token has been obtained HTTP request must contain *TaxCoreAuthenticationToken* key in the request HTTP headers with a valid authentication token as a value.

Digital Certificates and PIN Codes

The tax authority's system issues a Secure Element to a taxpayer as follows:

1. Taxpayer's digital certificate is stored in the Secure Element.
2. The Secure Element is stored on the smart card.
3. The PIN or password is generated and printed on the PIN mailer.
4. The Secure Element and PIN code are securely delivered to the taxpayer.

Digital Certificates for Testing Purpose

The tax authority will issue the requested number of test digital certificates to each accredited supplier and each accredited taxpayer.

Authentication Token

E-SDC uses an authentication token when calling the TaxCore API web services. Authentication token is obtained from TaxCore API by calling the service Request Authentication Token and providing a Taxpayer's digital certificate.

Role of the PKI Applet

PKI (public key infrastructure) Applet is installed along with the Secure Element Applet on the same Smart Card.

The role of the PKI applet is to support the secure communication and client certificate authentication with TaxCore.API using HTTPS protocol. The certificate used to establish a secure connection is stored on a smart card and it can be accessed from the PKI Applet using PKSC#11 API.

The certificate is loaded in the slot / token structure on the PKI Applet.

After the certificate is extracted from the smart card (in DER format) it can be used as a standard X.509 certificate for TLS/SSL and HTTPS protocols.

Valid PIN is required to read the certificate from PKI Applet using PKCS#11 API. Pin for PKI Applet is the same as the PIN for the Secure Element Applet.

Content

Required Drivers

Introduction

Smart Cards are programmed with PKI firmware according to GIDS (Generic Identity Device Specification) standard. Appropriate drivers shall be installed/programmed on an E-SDC in order to enable PKI Applet usage.

Windows OS Drivers

GIDS driver is an integral part of Windows OS since Windows 7 SP1, enabling the instant use of a smart card. No additional driver installation is required.

Linux OS Drivers

In order to use PKI Applet on Linux based OS, a pkcs11 driver from the OpenSC library is required. OpenSC libraries and tools are freely available on https://github.com/OpenSC.

In the following example, the installation of required drivers, libraries and tools on Debian / Ubuntu flavor of Linux OS with USB based card reader is shown. It is assumed that OpenSSL is used for TLS/SSL communication.

1. Install card reader driver

```
apt-get install libudev-dev
wget https://alioth.debian.org/frs/download.php/file/4126/pcsc-lite-x.y.z.tar.bz2
tar -xf pcsc-lite-x.y.z.tar.bz2
cd pcsc-lite-x.y.z
./configure
make
make install
```

```
aptitude install libusb-1.0-0-dev
wget https://alioth.debian.org/frs/download.php/file/4111/ccid-x.y.z.tar.bz2
tar -xf ccid-x.y.z.tar.bz2
cd ccid-x.y.z
./configure
make
make install
copy 92_pcscd_ccid.rules file from src directory to /etc/udev/rules.d/
# aptitude install libltdl-dev
wget http://ftp.de.debian.org/debian/pool/main/o/openct/openct_x.y.z.orig.tar.gz
tar -xf openct_x.y.z.orig.tar.gz
cd openct_x.y.z
./configure
# make
make all
```

2. Install OpenSSL development library

```
apt-get install libssl-dev
```

3. Install OpenSC package

```
wget http://cznic.dl.sourceforge.net/project/opensc/OpenSC/opensc-x.y.z/opensc-x.y.z.tar.gz
tar -xf opensc-x.y.z.tar.gz
cd opensc-x.y.z
./configure
make
make install
```

Run opensc-tool command from terminal

If message that libopensc.so.3 cannot be loaded find it with

```
find / -name "libopensc.so"
```

Copy found library to /usr/lib

4. Install libp11 library

```
apt-get install libp11-2
```

5. Install engine_pkcs11 library

- Download source code from https://github.com/OpenSC/engine_pkcs11/releases/
- Build and install library according to instructions found project page

After the above steps are executed, the certificate shall be accessible from the appropriate slot/token using a PKCS11 family of functions from the lipb11 library. ENGINE family of functions can be used to load the pkcs11 engine in the OpenSSL.

Other Platforms and Operating Systems

Please contact OpenSC community (https://github.com/OpenSC) for further information.

SDC

Introduction

SDC Section of TaxCore.API is used by any SDCs to establish connection with backend.

Activities

- Get initialization and configuration information required for normal operation
- Submit prepared audit packages
- Notify backend of online/offline status
- Pull pending commands from backend
- Notify backend of command execution results

Content

1. Request Authentication Token
   When requesting the authentication token, a Client authenticates itself with a valid Digital Certificate (stored in the PKI applet). If the token is successfully created it is returned to the Client as a string. In order to receive an authentication token, each client must establish a secure connection to "/api/v3/sdc/token" endpoint on TaxCore.API and authenticate using client digital certificate.

2. Get Initialization Commands
   For each new smart card issued by a tax authority, a set of commands is generated, which contain information necessary for invoice signing (Tax Rates, Verification URL, NTP server etc.).

3. Notify Online Status
   If an E-SDC is online, it shall periodically (once every 1 – 5 minutes) invoke the "Notify Online Status" operation on TaxCore.API.

4. Notify Command Processed
   After an E-SDC processes commands received from TaxCore.API, it will report the results of execution to TaxCore.API.

5. Submit Audit Package
   After the invoice audit package is created (explained in the section Creating an Audit Package), it shall be transferred to TaxCore.API the next time an Internet connection is available.

6. Submit Audit Request Payload ARP
   E-SDC invokes the Start Audit APDU command and receives 260 bytes of data that represent the Audit Request Payload (ARP). ARP has to be converted to the string using Base64 encoding.

Request Authentication Token

Introduction

When requesting the authentication token, a Client authenticates itself with a valid Digital Certificate (stored in the PKI applet). If the token is successfully created it is returned to the Client as a string. In order to receive an authentication token, each client must establish a secure connection to "/api/v3/sdc/token" endpoint on TaxCore.API and authenticate using client digital certificate.

A request is composed as follows:

1. Create HTTPS GET request object
2. Add HTTP headers "Accept: application/json" and "Content-Type: application/json"
3. Read certificate from the PKI Applet
4. Use the certificate from the PKI Applet to establish SSL/TLS connection
5. Send a request to "/api/v3/sdc/token" operation on TaxCore.API web service.
6. Read the response as JSON structure defined below

Endpoints

Endpoint	Example
<TaxCore_API_URL_obtained_from_certificate_as_e: here>/api/v3/sdc/token	`https://api.sandbox.suf.purs.gov.rs/api/v`

NOTE:
Development and production environments, as well the environments in different countries, have different URLs. For this reason, URLs and names in your documentation, code and UI should not be hardcoded but configurable or extracted from a digital certificate.

Method

`GET`

Header

Add the following HTTP headers to each request:

- `Accept: application/json`

Authentication

The certificate-based authentication is used only to request a token. Request for the authentication token is periodically invoked to obtain a new token and to verify the date and time.

For authentication details please refer to Authentication

Request

N/A

Response

Field	Type	Description
token	string	The Token is valid for 8 hours by default. A Client uses the current token when calling all other services exposed by TaxCore.API.
expiresAt	string	Date and time of token expiration - in UTC time. When a token expires a Client must request a new token. If the Client requests a new token while the current token is still valid, TaxCore will return the current token.

Example

```
{
  "token": "245ebd69-1438-4dc3-a65b-18f1a527f093",
  "expiresAt": "2020-12-23 15:18:33Z"
}
```

Get Initialization Commands

Introduction

For each new smart card issued by a tax authority, a set of commands is generated, which contain information necessary for invoice signing (Tax Rates, Verification URL, NTP server etc.).

Commands can be downloaded using one of the following channels:

- By invoking TaxCore.API service Notify Online Status (typically by E-SDC)
- By invoking TaxCore.API service Submit Audit Package (typically by E-SDC)
- By invoking TaxCore.API service Get Initialization Commands (typically by E-SDC)
- By using the Taxpayer Administration Portal

Once the commands are processed, E-SDC reports the execution status to TaxCore.API as explained in section Notify Command Processed.

E-SDC can explicitly require initialization commands, by invoking TaxCore.API service *Get Initialization Commands*.

Initialization commands include:

- *Configure Time Server URL Command*
- *Set Tax Rates Command*
- *Update Verification URL Command*
- *Update TaxCore Configuration*

To get initialization commands compose HTTPS GET request as follows:

1. Add headers "Accept: application/json", "Content-Type: application/json" and header that contains an authentication token
2. Submit GET request to https://<taxcore_api_url>/api/v3/sdc/commands

Endpoints

Endpoint	Example
<TaxCore_API_URL_obtained_from_certificate_as_e: here>/api/v3/sdc/commands	`https://api.sandbox.suf.purs.gov.rs/api/v`

NOTE:
Development and production environments, as well the environments in different countries, have different URLs. For this reason, URLs and names in your documentation, code and UI should not be hardcoded but configurable or extracted from a digital certificate.

Method

`GET`

Header

Add the following HTTP headers to each request

- `TaxCoreAuthenticationToken: <token-value-returned-from-Request-Authentication-Token-method>`
- `Accept: application/json`

Authentication

For authentication details please refer to Authentication

Request

N/A

Response

The response contains a list of commands that must be executed by E-SDC, as described in section Commands.

Notify Online Status

Introduction

If an E-SDC is online, it shall periodically (once every 1 – 5 minutes) invoke the “Notify Online Status” operation on TaxCore.API.

Compose HTTPS PUT request as follows:

1. Add headers "Accept: application/json", "Content-Type: application/json" and header that contains authentication token
2. Add a string “true” or "false" to the body of the request (depending on whether the E-SDC is online or going offline)
3. Submit PUT request to https://<taxcore_api_url>/api/v3/sdc/status

After the request is sent, TaxCore.API shall return a response with a JSON formatted string containing a list of commands, as described in section Commands, that an E-SDC shall execute (new tax rates, verification URL, NTP URL or public key used for encryption). The Command list can be empty.

Endpoints

Endpoint	Example
`https://api.sandbox.suf.purs.gov.rs/api`	`https://api.sandbox.taxcore.online/api/v3`

Method

`PUT`

Header

Add the following HTTP headers to each request

- `TaxCoreAuthenticationToken: <token-value-returned-from-Request-Authentication-Token-method>`
- `Accept: application/json`
- `Content-Type: application/json`

Authentication

For authentication details please refer to Authentication

Request

true

The list of commands will be obtained only if the string "true" is submitted in the request

Response

The response contains a list of commands that must be executed by E-SDC, as described in section Commands.

Notify Command Processed

Introduction

After an E-SDC processes commands received from TaxCore.API, it will report the results of execution to TaxCore.API.

1. Add headers "Accept: application/json", "Content-Type: application/json" and header that contains authentication token
2. Add a string “true” or "false" to the body of the request (depending on whether the E-SDC successfully processed commands)
3. Submit PUT request to https://<taxcore_api_url>/api/v3/sdc/commands/{commandId}

Endpoints

Endpoint	Example
<TaxCore_API_URL_obtained_from_certificate_as_e: here>/api/v3/sdc/commands/{commandId}	`https://api.sandbox.suf.purs.gov.rs/api/v 205A-4CBF-AD0C-6617D42AE466`

NOTE:
Development and production environments, as well the environments in different countries, have different URLs. For this reason, URLs and names in your documentation, code and UI should not be hardcoded but configurable or extracted from a digital certificate.

Method

`PUT`

Header

Add the following HTTP headers to each request

- `TaxCoreAuthenticationToken: <token-value-returned-from-Request-Authentication-Token-method>`
- `Accept: application/json`
- `Content-Type: application/json`

Authentication

For authentication details please refer to Authentication

Request

true

Example

https://api.sandbox.taxcore.online/api/v3/sdc/commands/CC63C53D-205A-4CBF-AD0C-6617D42AE466

Response

HTTP 200 OK

Submit Audit Package

After the invoice audit package is created (explained in the section Creating an Audit Package), it shall be transferred to TaxCore.API the next time an Internet connection is available.

Compose HTTPS POST request as follows:

1. Add headers "Accept: application/json", "Content-Type: application/json" and header that contains an authentication token
2. Add an Audit Package as a JSON message to the body of the HTTP POST request
3. Submit POST request to https://<taxcore_api_url>/api/v3/sdc/audit

After the request is sent, TaxCore.API responds with a JSON formatted text containing a status of operation and a list of commands that an E-SDC shall execute.

NOTE:
Values for `sdcDateTime` and all other fields (see Create Invoice) must match the values submitted to the Secure Element for digital signing (see *Sign Invoice* in Fiscalization), as well as the values in the verification URL (see Create Verification URL).

NOTE:
In case the field `Request.Item.Name` exceeds the defined maximum length (see Create Invoice), its value is truncated.

NOTE:
In case the field `Request.DateAndTimeOfIssue` is out of range (see Create Invoice), its value is replaced with a null value.

Endpoints

Endpoint	Example
<TaxCore_API_URL_obtained_from_certificate_as_e: here>/api/v3/sdc/audit	`https://api.sandbox.suf.purs.gov.rs/api/v`

NOTE:
Development and production environments, as well the environments in different countries, have different URLs. For this reason, URLs and names in your documentation, code and UI should not be hard-coded but configurable or extracted from a digital certificate.

Method

`POST`

Header

Add the following HTTP headers to each request

- `TaxCoreAuthenticationToken: <token-value-returned-from-Request-Authentication-Token-method>`
- `Accept: application/json`
- `Content-Type: application/json`

Authentication

For authentication details, please refer to Authentication.

Request

Request that must be generated and sent by SDC is described in this section Format of the Audit Package.

Example

```
{
  "key": "VGhpcyBJcyBLZXkK...",
  "iv": "VGhpcyBJcyBJVgo...",
  "payload": "VGhpcyBJcyBQYXlsb2FkCg..."
}
```

Response

The response contains a list of commands that must be executed by E-SDC, as described in section Commands.

```
AuditDataStatus {
status (integer, optional) = ['0', '1', '2', '3', '4', '5', '6']integerEnum:0, 1, 2, 3, 4, 5,
commands (Array[Command], optional)
}]
```

Data Fields

status - returned after TaxCore.API unpacks and verifies audit packages. If all verifications are successful, the status should have the value **4**. Other values can help E-SDC developers rectify problems with audit packages. Their meanings are the following:

- **0** - Invalid audit package - TaxCore API failed to decrypt the received audit package. The possible reasons are: the received file is corrupt, has no content, the file is encrypted using the wrong TaxCore public key, etc.
- **1** - Invoice cannot be stored - the received invoice is probably valid, but due to the internal server error TaxCore.API is unable to store it
- **72** - E-SDC sent the wrong TIN or SignedBy or RequestedBy fields when signing this invoice
- **3** - Invoice internal data was encrypted in a wrong way
- **4** - Invoice is verified
- **5** - Taxes on this invoice were calculated using a tax rates group that does not exist or is obsolete
- **6** - At the moment of invoice signing, the certificate on the secure element was already revoked
- **8** - At the moment of invoice signing, the certificate on the secure element was not officially issued
- **23** - Information about the E-SDC's Manufacturer Registration Code is missing
- **24** - Information about the E-SDC's Manufacturer Registration Code is in a wrong format
- **67** - SDC time is out of the allowed range

68 - Reference time is out of the allowed range

- **69** - Invoice contains TaxItem for a label that does not exist in the tax rate group named in `Result.TaxGroupRevision`
- **70** - Invoice contains TaxCounters (in internal data) for CategoryOrderId which does not exist in the tax rate group named in `Result.TaxGroupRevision`, i.e. E-SDC sent a non-existing CategoryOrderId to the secure element
- **71** - The invoice was submitted without information about the payment
- **73** - Invoice contains SDC time (`result.sdcDateTime`) that is in the future compared to the time of invoice arrival to TaxCore.API. The gap is greater than the allowed tolerance period.
- **74** - Invoice contains Reference time (`request.referentDocumentDT`) that is in the future compared to the invoice's SDC time (`result.sdcDateTime`). The gap is greater than the allowed tolerance period.
- **75** - Invoice contains SDC time (`result.sdcDateTime`) that is in the past compared to the time of invoice arrival to TaxCore.API. The gap is greater than the allowed tolerance period.
- **76** - Invoice contains Reference time (`request.referentDocumentDT`) that is in the past compared to the invoice's SDC time (`result.sdcDateTime`). The gap is greater than the allowed tolerance period.
- **78** - Invoice does not contain a Verification URL
- **commands** - contains a list of commands that E-SDC should execute, as described in section Commands.

NOTE:
If status 4 (*Invoice is verified*) is received from TaxCore.API, that audit package should immediately be deleted from the E-SDC's local storage (see Creating an Audit Package). If any other status is received, the audit package must not be deleted.

NOTE:
The E-SDC should **try to resubmit** an audit package to TaxCore.API. **only** if it receives status 1 (*Invoice cannot be stored*) from the TaxCore.API. If any other status is received, the audit package should not be resubmitted.

Example


```
{
  "status": 0,
  "commands": [
    {
      "commandId": "3930CEEF-F637-444D-8295-F629D6E482D3",
      "type": 1,
      "payload": "0.europe.pool.ntp.org",
      "uid": "ABCD1234"
    }
  ]
```

}

Submit Audit Request Payload - ARP

E-SDC invokes the Start Audit APDU command and receives 260 bytes of data that represent the Audit Request Payload (ARP). ARP has to be converted to the string using Base64 encoding.

E-SDC invokes the Amount Status APDU command and receives the current sum and limit for the secure element.

These 3 values are submitted to endpoint https://<taxcore_api_url>/api/v3/sdc/audit-proof as an Audit-Proof Request structure in the body of the HTTP request.

Compose HTTPS POST request as follows:

1. Add headers "Accept: application/json", "Content-Type: application/json" and header that contains authentication token
2. Create a request structure as per the below model and add it to the body of the HTTP POST request
3. Submit POST Request to https://<taxcore_api_url>/api/v3/sdc/audit-proof

Endpoints

Endpoint	Example
<TaxCore_API_URL_obtained_from_certificate_as_e: here>/api/v3/sdc/audit-proof	`https://api.sandbox.suf.purs.gov.rs/api/v proof`

NOTE:
Development and production environments, as well the environments in different countries, have different URLs. For this reason, URLs and names in your documentation, code and UI should not be hardcoded but configurable or extracted from a digital certificate.

Method

`POST`

Header

Add the following HTTP headers to each request

-

```
TaxCoreAuthenticationToken: <token-value-returned-from-Request-Authentication-Token-method>
```

- `Accept: application/json`
- `Content-Type: application/json`

Authentication

For authentication details please refer to Authentication

Request

JSON structure as defined in section Format of the Audit-Proof Request

Model

```
ProofOfAuditRequest {
auditRequestPayload (string),

sum (integer) 64bit unsigned,

limit (integer) 64bit unsigned
}
```

Example

```
{
  "auditRequestPayload": "d4A/iLtwmDYeZyacm/nDlCF...",
  "sum": 11034,
  "limit": 100000
}
```

Response

```
HTTP 200 OK
```

Secure Element Applet API

Communication with a Secure element Applet API is performed through standard APDU commands.

For a detailed description of APDU communication, APDU commands data structure and particular bytes meaning, please refer to ISO/IEC 7816-4 standard.

Commands are grouped into three categories based on the type of usage:

1.

..
   General
2. Fiscalization
3. Audit

Important Notes

1. All APDU commands are sent to the Smart Card using T1 communication protocol
2. All amounts or counter values are submitted to/received from the Secure element using Big-endian. Big-endian is an order in which the "big end" (most significant value in the sequence) is stored first (at the lowest storage address)
3. P1 and P2 values are considered in the request processing when,
    1. Select Applet Command
    2. Force using CRC for Data in APDU transmission
4. PIN is sent in ASCII hex format from **SE applet version 3.2.2**
5. CRC is available from **SE applet version 3.2.5**, and it is optional to use
6. From **SE applet version 3.2.8**, the SDC Time on an invoice must be within the secure element's certificate validity period, i.e., between NotBefore and NotAfter
7. PIN can be sent in both ASCII and decimal hex format from **SE applet version 3.2.9** as a backward compatibility. ASCII hex format is considered the default behavior
8. From **SE applet version 3.2.10**, if the SDC Time on an invoice is outside of the secure element's certificate validity period (i.e., before the NotBefore limit or after the NotAfter limit), the secure element returns error code 0x6308

Content

1. General Commands
   Secure Element Applet is installed as a non-default applet on a smart card. Before any APDU command is invoked, the applet is selected using the standard Select command.

2. Fiscalization
   PIN verification is a method that “unlocks” a card for invoice signing and other operations protected by PIN code. Depending on the SE applet version, PIN is sent in decimal or hex format with ASCII encoding, and it is sent as an array of byte digits.

3. Audit
   Returns 259 bytes data structure represents public card key (256 bytes modulus and 3 bytes exponent). This key is used to encrypt Audit packages.

4. Secure Element Specific APDU Error Codes
   This table contains the expected error codes and descriptions that a caller may encounter while working with the Secure Element Applet.

General Commands

Secure Element Applet is installed as a non-default applet on a smart card. Before any APDU command is invoked, the applet is selected using the standard Select command.

NOTE:
The availability of specific commands, as well as their content, depends on the secure element (SE) version. You can use the *Get Secure Element Version* command (see below) to check the version of the SE you are using.

Select Applet

As previously mentioned, the Smart Card has two applets installed. This command selects the Secure Element Applet and routes subsequent APDU commands to it.

APDU Request

SE Version	IsoCase	Class	Instruction	P1-P2	Command Length (Lc)	Command Data	Expecte Lengt (Le)
>= 2.0.0	Case3Sho	0x00	0xA4	0x040(	0x10	0xA000000748	0x00

APDU Response

SE Version	Response Data	SW1SW2
>= 2.0.0	none	0x9000

Example:

Request: `00A4040010A000000748464A492D546178436F726500`

Response: `9000`

Get Secure Element Version

This command returns the version information about the current Api version. The response contains 12 bytes, where each 4 bytes represent unsigned integer of one version segment, making total of 3 version segments:

major, minor and patch.

APDU Request

SE CAP Version	IsoCase	Class	Instruction	P1-P2	Command Length (Lc)	Command Data	Expected Length (Le)
>= 2.0.0	`Case2Sho`	`0x88`	`0x08`	`0x000(`	none	none	`0x00`

APDU Response

SE CAP Version	Response Data	SW1SW2
>= 2.0.0	*12 bytes*	`0x9000`

Example 1:

Request: `8808040000`

Response: `000000020000000000000000 9000`

Example 2:

Request: `8808000000`

Response: `000000030000000100000001 9000`

Example 3:

Request: `8808000000`

Response: `000000030000000200000005 9000`

Forward Secure Element Directive

This command is used by E-SDC to forward instructions received from TaxCore.Api to Secure Element Applet via Secure Element APDU Command.

If APDU Command status (SW1SW2) is OK (`0x9000`), consider forward instructions operation is completed.

NOTE:
From the SE version 3.2.5, optionally, CRC can be calculated and used for data verification. If CRC is not used, the command is the same as in the previous applet version.

APDU Request

SE Version	IsoCase	Class	Instruction	P1-P2	Command Length (Lc)	Command Data	Expected Length (Le)
>= 2.0.0 (no CRC)	Case3Ext	0x88	0x40	0x040(	0x000200	*512 bytes received from TaxCore*	none
>= 3.2.5 (with CRC)	Case3Ext	0x88	0x40	0x0102	0x000204	*512 bytes received from TaxCore + 4 bytes for CRC*	none

APDU Response

SE Version	Response Data	SW1SW2
>= 2.0.0 (no CRC)	none	0x9000
>= 3.2.5 (with CRC)	none	0x9000

Example 1 (without CRC):

Command Data:
5DBFC9CD04AF9DC76C50FA3FF54D32D1910B0D2E1EC5AF97EAE3E71A7423CCE066D6E264255838C1DBAD

Request:
884004000002005DBFC9CD04AF9DC76C50FA3FF54D32D1910B0D2E1EC5AF97EAE3E71A7423CCE066D6E2

Response: 9000

Example 2 (with CRC):

Command Data witout CRC:
5DBFC9CD04AF9DC76C50FA3FF54D32D1910B0D2E1EC5AF97EAE3E71A7423CCE066D6E264255838C1DBAD

Command Data CRC: F50CFF4B

Command Data:
5DBFC9CD04AF9DC76C50FA3FF54D32D1910B0D2E1EC5AF97EAE3E71A7423CCE066D6E264255838C1DBAD

Request:
`884004000002045DBFC9CD04AF9DC76C50FA3FF54D32D1910B0D2E1EC5AF97EAE3E71A7423CCE066D6E2`

Response: `9000`

Export Certificate

This command exports the taxpayer certificate in a DER format. This certificate contains location data that is present on the textual representation of an invoice.

APDU Request

SE Version	IsoCase	Class	Instruction	P1-P2	Command Length (Lc)	Command Data	Expected Length (Le)
>= 2.0.0	`Case2Ext`	`0x88`	`0x04`	`0x040(`	none	none	`0x000000`

APDU Response

SE Version	Response Data	SW1SW2
>= 2.0.0	*raw bytes random length*	`0x9000`

Example:

Request: `88040400000000`

Response: *raw bytes of x509 certificate public key* + `9000`

Get Last Signed Invoice

This command returns information about the last singed invoice. The structure of the data recived is the same as the response is in the Sign Invoice command.

NOTE:
From the SE version 3.2.5, optionally, CRC can be calculated and used for data verification. If CRC is not used, the command is the same as in the previous applet version.

APDU Request

SE	IsoCase	Class	Instruction	P1-P2	Command	Command	Expected

Version					Length (Lc)	Data	Length (Le)
>= 3.1.1 (no CRC)	Case2Ext	0x88	0x15	0x040(	none	none	0x000000
>= 3.2.5 (with CRC)	Case2Ext	0x88	0x15	0x010:	none	none	0x000000

APDU Response

SE Version	Response Data	SW1SW2
>= 3.1.1 (no CRC)	*577 or 833 bytes*	0x9000
>= 3.2.5 (with CRC)	*581 or 837 bytes*	0x9000

Example 1 (without CRC):

Request: 88150400000000

Response: *577 or 833 bytes* + 9000

Response Data

Start (byte)	Length (bytes)	Field	Description
0	8	Date/time	Same as data sent from E-SDC to SE
8	20	Taxpayer ID	Same as data sent from E-SDC to SE
28	20	Buyer ID	Same as data sent from E-SDC to SE
48	1	Invoice type	Same as data sent from E-SDC to SE
49	1	Transaction type	Same as data sent from E-SDC to SE
50	7	Invoice amount	Same as data sent from E-SDC to SE
57	4	Sale or refund counter value	Depends on request's Tax type field
61	4	Total counter value (sale+refund)	Unsigned int 32bit big endian,

65	256 or 512	Encrypted Internal Data	Encrypted Internal Data length depends on the number of available tax rates programmed during personalization. It may be 256 or 512 bytes long.
321 or 577	256	Digital signature	

Example 2 (with CRC):

Request: `8815010200`

Response: *581 or 837* + `9000`

Response Data

Start (byte)	Length (bytes)	Field	Description
0	8	Date/time	Same as data sent from E-SDC to SE
8	20	Taxpayer ID	Same as data sent from E-SDC to SE
28	20	Buyer ID	Same as data sent from E-SDC to SE
48	1	Invoice type	Same as data sent from E-SDC to SE
49	1	Transaction type	Same as data sent from E-SDC to SE
50	7	Invoice amount	Same as data sent from E-SDC to SE
57	4	Sale or refund counter value	Depends on request's Tax type field
61	4	Total counter value (sale+refund)	Unsigned int 32bit big endian,
65	256 or 512	Encrypted Internal Data	Encrypted Internal Data length depends on the number of available tax rates programmed during personalization. It may be 256 or 512 bytes long.
321 or 577	256	Digital signature	
577 or 833	4	CRC	CRC is calculated from 0 to 577 or 833 bytes.

Get PIN tries left from SE Applet

This command returns how many PIN tries are left before the card is locked

APDU Request

SE Version	IsoCase	Class	Instruction	P1-P2	Command Length (Lc)	Command Data	Expected Length (Le)
>= 3.1.1	Case2Sho	0x00	0x16	0x040(	none	none	0x00

APDU Response

SE Version	Response Data	SW1SW2
>= 3.1.1	05 if 5 tries are left, 00 if the card is blocked	0x9000

Example:

Request: 8816040000

Response: 05 9000

Get CertParams

This command returns UID, SE Certificate NotBefore and SE Certificate NotAfter. NotBefore and NotAfter are in UTC time in Unix Timestamp format.

APDU Request

SE Version	IsoCase	Class	Instruction	P1-P2	Command Length (Lc)	Command Data	Expected Length (Le)
>= 3.2.8	Case2Sho	0x00	0x33	0x000(	none	none	0x00

APDU Response

SE Version	Response Data	SW1SW2

>= 3.2.8	*24 bytes*	`0x9000`

Example:

Request: `8833000000`

Response: `445337584C53524500001968743CA28000001AC9386D1E8 9000`

CertParam	Hex	Transform Value
UID	445337584C535245	DS7XLSRE
NotBefore	000001968743CA28	04/30/2025 15:14:49
NotAfter	000001AC9386D1E8	04/30/2028 15:24:49

Fiscalization

PIN Verify

PIN verification is a method that “unlocks” a card for invoice signing and other operations protected by PIN code. Depending on the SE applet version, PIN is sent in decimal or hex format with ASCII encoding, and it is sent as an array of byte digits.

For example, PIN 1234 can be represented in the following formats:

- decimal format - PIN is represented as 0x01, 0x02, 0x03, 0x04.
- ASCII hex format - PIN is represented as 0x31, 0x32, 0x33, 0x34.

APDU Request

SE Version	IsoCase	Class	Instruction	P1-P2	Command Length (Lc)	Command Data	Expected Length (Le)
2.0.0 □ SE version < 3.2.2 3.2.9 □ SE	`Case3Sho`	`0x88`	`0x11`	`0x000(`	`0x04`	*4 bytes where each represents*	none

version						*one PIN digit in decimal format*	
3.2.2 □ SE version	Case3Sho	0x88	0x11	0x000(	0x04	*4 bytes where each represents one PIN digit in ASCII hex format*	none

Example:

This is an example for PIN 1234.

SE Version	Command Data	Request	Response (correct PIN)	Error response (wrong PIN)
2.0.0 □ SE version < 3.2.2	01020304	88110000040102030	9000	6302
>= 3.2.2	31323334	88110000043132333	9000	6302

Sign Invoice

Signs invoice and returns fiscalization data for a submitted invoice.

NOTE:
From the SE version 3.2.5: Optional - CRC can be calculated and used for data verification. If CRC is not used, the command is the same as in the previous applet version.

From applet version 3.2.8: Mandatory - Invoice Date/time must be greater then Certificate NotBefore and lower then Certificate NotAfter.

APDU Request

SE Version	IsoCase	Class	Instruction	P1-P2	Command Length (Lc)	Command Data	Expected Length (Le)
>= 2.0.0 (no CRC)	Case4Ext	0x88	0x13	0x040(	*3 byte Command*	*Command Data byte*	0x0000

					Data byte array length	*array*	
>= 3.2.5 (with CRC)	`Case4Ext`	`0x88`	`0x13`	`0x0102`	*3 byte Command Data byte array length*	*Command Data byte array + 4 bytes for CRC*	`0x0000`

APDU Response

SE Version	Response Data	SW1SW2
>= 2.0.0 (no CRC)	*byte array*	`0x9000`
>= 3.2.5 (with CRC)	*byte array + 4 byte CRC*	`0x9000`

Data structure without CRC:

Command data:

Start (byte)	Length (byte)	Field	Description
0	8	Date/time	E-SDC timestamp UTC time in Unix Timestamp. Example: 1495018011910 is 2017-05-17T10:46:51.910Z
8	20	Taxpayer ID	Hex encoded byte array, leading bytes filled with 0x00. Taxpayer ID value can consist only of ascii printable characters. **Zeros can be added only on the left side**. MSB are sent first Example: Taxpayer ID = 928615467, Byte array = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x39, 0x32, 0x38, 0x36, 0x31, 0x35, 0x34, 0x36, 0x37} (byte 0x37 is sent last to SE)
28	20	Buyer ID	If unknown, leave zeroes. Formatting is the same as for Taxpayer ID
48	1	Invoice type	Values 0, 1, 2, 3, 4 as explained in section Create Invoice.
49	1	Transaction Type	Sale=0, Refund=1

50	7	Invoice amount	Sale or refund total amount (including taxes) - depends on applied tax types
57	1	Number of tax categories	Defines the number of tax categories which appear on the invoice (value between 0 and 26). The following data structure **Tax Categories** must be repeated exactly this number of times.
58	8	Tax Category (1)	The first Tax Category (mandatory if **Number of tax categories > 0**)
66	8	Tax Category (2)	The second Tax Category (mandatory if **Number of tax categories > 1**)
74	...	Tax Category (n)	

Tax Categories:

Start (byte)	Length (byte)	Field	Description
58	[1]	[Tax category ID]	The first tax category's OrderID, as explained in Tax Rates section (mandatory if **Number of tax categories** > 0)
59	[7]	[Tax category amount]	The first total tax amount for the category specified in preceding field **Tax category ID** (mandatory if **Number of tax categories** > 0)
66	[1]	[Tax category ID]	The next tax category's OrderID (mandatory if **Number of tax categories** > 1)
67	[7]	[Tax category amount]	The next total tax amount for the category specified in preceding field **Tax category ID** (mandatory if **Number of tax categories** > 1)

Response data:

Start (byte)	Length (bytes)	Field	Description
0	8	Date/time	Same as data sent from E-SDC to SE
8	20	Taxpayer ID	Same as data sent from E-SDC to SE
28	20	Buyer ID	Same as data sent from E-SDC to SE

48	1	Invoice type	Same as data sent from E-SDC to SE
49	1	Transaction type	Same as data sent from E-SDC to SE
50	7	Invoice amount	Same as data sent from E-SDC to SE
57	4	Sale or refund counter value	Depends on request's Tax type field
61	4	Total counter value (sale+refund)	unsigned int 32bit big endian,
65	256 or 512	Encrypted Internal Data	Encrypted Internal Data length depends on the number of available tax rates programmed during personalization. It may be 256 or 512 bytes long.
321 or 577	256	Digital signature	

Example without CRC:

Command Data:
`0000017B2D198AC40000000000000000050432D313030303030303031000000000000000000000000000000`

Request:
`8813040000009A0000017BE9B01AB40000000000000000050432D313030303030303031000000000000000`

Response: *byte array* + `9000`

Data structure with CRC:

Command data:

Start (byte)	Length (byte)	Field	Description
0	8	Date/time	E-SDC timestamp UTC time in Unix Timestamp. Example: 1495018011910 is 2017-05-17T10:46:51.910Z
8	20	Taxpayer ID	Hex encoded byte array, leading bytes filled with 0x00. Taxpayer ID value can consist only of ascii printable characters. **Zeros can be added only on the left side**. MSB are sent first Example: Taxpayer ID = 928615467, Byte array = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x39, 0x32, 0x38, 0x36, 0x31, 0x35, 0x34, 0x36, 0x37}

			(byte 0x37 is sent last to SE)
28	20	Buyer ID	If unknown, leave zeroes. Formatting is the same as for Taxpayer ID
48	1	Invoice type	Values 0, 1, 2, 3, 4 as explained in section Create Invoice.
49	1	Transaction Type	Sale=0, Refund=1
50	7	Invoice amount	Sale or refund total amount (including taxes) - depends on applied tax types
57	1	Number of tax categories	Defines the number of tax categories which appear on the invoice (value between 0 and 26). The following data structure **Tax Categories** must be repeated exactly this number of times.
58	8	Tax Category (1)	The first Tax Category (mandatory if **Number of tax categories > 0**)
66	8	Tax Category (2)	The second Tax Category (mandatory if **Number of tax categories > 1**)
74	...	Tax Category (n)	
...	4	CRC	CRC is calculated from 0 to 74 bytes (or to last byte if data).

Tax Categories:

Start (byte)	Length (byte)	Field	Description
58	[1]	[Tax category ID]	The first tax category's OrderID, as explained in Tax Rates section (mandatory if **Number of tax categories** > 0)
59	[7]	[Tax category amount]	The first total tax amount for the category specified in preceding field **Tax category ID** (mandatory if **Number of tax categories** > 0)
66	[1]	[Tax category ID]	The next tax category's OrderID (mandatory if **Number of tax categories** > 1)
67	[7]	[Tax category amount]	The next total tax amount for the category specified in preceding field **Tax category ID** (mandatory if **Number of tax categories** > 1)

Response data:

Start (byte)	Length (bytes)	Field	Description
0	8	Date/time	Same as data sent from E-SDC to SE
8	20	Taxpayer ID	Same as data sent from E-SDC to SE
28	20	Buyer ID	Same as data sent from E-SDC to SE
48	1	Invoice type	Same as data sent from E-SDC to SE
49	1	Transaction type	Same as data sent from E-SDC to SE
50	7	Invoice amount	Same as data sent from E-SDC to SE
57	4	Sale or refund counter value	Depends on request's Tax type field
61	4	Total counter value (sale+refund)	unsigned int 32bit big endian,
65	256 or 512	Encrypted Internal Data	Encrypted Internal Data length depends on the number of available tax rates programmed during personalization. It may be 256 or 512 bytes long.
321 or 577	256	Digital signature	
577 or 833	4	CRC	CRC is calculated from 0 to 577 or 833 bytes.

Example with CRC:

Command Data:
`0000017B2D198AC40000000000000000050432D313030303030303031000000000000000000000000000000`

Command Data CRC: `90F2BC39`

Request:
`88130102E0000017BE9B01AB40000000000000000050432D313030303030303031000000000000000000`

Response: *byte array invoice + 4 byte CRC +* `9000`

Amount Status

Returns 14-bytes-long data structure (7 bytes for sum SALE and REFUND, and 7 bytes for Limit Amount)

APDU Request

SE Version	IsoCase	Class	Instruction	P1-P2	Command Length (Lc)	Command Data	Expected Length (Le)
>= 2.0.0	`Case2Sho`	`0x88`	`0x14`	`0x040(`	none	none	`0x00`

APDU Response

SE Version	Response Data	SW1SW2
>= 2.0.0	*14 byte array*	`0x9000`

Example:

Request: `8814040000`

Response: `0000724AA18328038D7EA4C68000 9000` (SALE+REFUND=490878370600 , Limit Amount=1000000000000000)

Audit

Export TaxCore Public Key

Returns 259 bytes data structure represents public card key (256 bytes modulus and 3 bytes exponent). This key is used to encrypt Audit packages.

APDU Request

SE Version	IsoCase	Class	Instruction	P1-P2	Command Length (Lc)	Command Data	Expected Length (Le)
>= 2.0.0	`Case2Ext`	`0x88`	`0x07`	`0x040(`	none	none	`0x000000`

APDU Response

SE Version	Response Data	SW1SW2
>= 2.0.0	*259 bytes data*	0x0900

Example:

Request: 80070400000000

Response: *256 bytes modulus + 3 bytes exponent* + 9000

Export Audit Data

Exports encrypted audit data.

NOTE:
From the SE version 3.2.5, optionally, CRC can be calculated and used for data verification. If CRC is not used, the command is the same as in the previous applet version.

APDU Request

SE Version	IsoCase	Class	Instruction	P1-P2	Command Length (Lc)	Command Data	Expected Length (Le)
>= 2.0.0 (no CRC)	Case2Ext	0x88	0x12	0x040(	none	none	0x000000
>= 3.2.5 (with CRC)	Case2Ext	0x88	0x12	0x0102	none	none	0x000000

APDU Response

SE Version	Response Data	SW1SW2
>= 2.0.0 (no CRC)	*565 or 821 bytes data*	0x9000
>= 3.2.5 (with CRC)	*569 or 825 bytes data*	0x9000

NOTE:
Depending on the Internal Data, the total length of the structure is 565 or 821 bytes. For versions **3.2.5 or

later** if CRC is used, the total lenght can be 569 or 825 if CRC is added.

Exported audit data has the following structure, without CRC:

Offset	Length	Data	Note
0	4	TaxCore Key Version	
4	256	Crypted Internal Data	The length of Crypted Internal Data can be 256 or 512 bytes
260 or 516	20	Taxpayer Identification Number (TIN)	
280 or 536	20	Buyer ID	
300 or 556	1	Invoice type	
301 or 557	1	Transaction type	
302 or 558	7	Invoice amount	
309 or 565	256	Digital signature of the above structure	

Exported audit data has the following structure, with CRC:

Offset	Length	Data	Note
0	4	TaxCore Key Version	
4	256	Crypted Internal Data	The length of Crypted Internal Data can be 256 or 512 bytes
260 or 516	20	Taxpayer Identification Number (TIN)	
280 or 536	20	Buyer ID	
300 or 556	1	Invoice type	
301 or 557	1	Transaction type	
302 or 558	7	Invoice amount	
309 or 565	256	Digital signature of the above structure	

565 or 821	4	CRC	CRC is calculated from 0 to 565 or 821 bytes.

Example 1 (without CRC):

Request: `88120400000000`

Response: *565 or 821 bytes* + `9000`

Example 2 (with CRC):

Request: `88120102000000`

Response: *569 or 825 bytes* + `9000`

Start Audit

Notifies the Secure element that the audit process has been initialized by E-SDC.

Secure element returns an encrypted message that shall be submitted to TaxCore as the content of the field `auditRequestPayload` of audit-proof request.

NOTE:
From the SE version 3.2.5, optionally, CRC can be calculated and used for data verification. If CRC is not used, the command is the same as in the previous applet version.

APDU Request

SE Version	IsoCase	Class	Instruction	P1-P2	Command Length (Lc)	Command Data	Expected Length (Le)
>= 2.0.0 (no CRC)	`Case2Ext`	`0x88`	`0x21`	`0x040`	none	none	`0x000000`
>= 3.2.5 (with CRC)	`Case2Ext`	`0x88`	`0x21`	`0x0102`	none	none	`0x000000`

APDU Response

SE Version	Response Data	SW1SW2
>= 2.0.0 (no CRC)	*260 bytes data*	`0x9000`

>= 3.2.5 (with CRC)	*264 bytes data*	`0x9000`

Example 1 (without CRC):

Request: `88210400000000`

Response: *260 bytes data* + `9000`

Example 2 (with CRC):

Request: `88210102000000`

Response: *260 bytes data + 4 bytes CRC data* + `9000`

End Audit

Notifies the Secure element that the audit process has been finalized by TaxCore. If APDU Command status is OK (0x90 0x00) consider the audit operation is completed.

NOTE:
From the SE version 3.2.5, optionally, CRC can be calculated and used for data verification. If CRC is not used, the command is the same as in the previous applet version.

APDU Request

SE Version	IsoCase	Class	Instruction	P1-P2	Command Length (Lc)	Command Data	Expected Length (Le)
>= 2.0.0 (no CRC)	`Case3Ext`	`0x88`	`0x20`	`0x040`	`0x000100`	*256 bytes received from TaxCore*	none
>= 3.2.5 (with CRC)	`Case3Ext`	`0x88`	`0x20`	`0x0102`	`0x000104`	*256 bytes received from TaxCore + 4 bytes for CRC*	none

APDU Response

SE Version	Response Data	SW1SW2

>= 2.0.0 (no CRC)	none	`0x9000`
>= 3.2.5 (with CRC)	none	`0x9000`

Example 1 (without CRC):

Command Data:
`253AB91A21859A06813E8A880E10BA0C67A09DDBED0B7E001F638CA015D2E414744E0C5C2E0F5F827DFC`

Request:
`882004000001002 53AB91A21859A06813E8A880E10BA0C67A09DDBED0B7E001F638CA015D2E414744E0C 9000`

Example 2 (with CRC):

Command Data:
`253AB91A21859A06813E8A880E10BA0C67A09DDBED0B7E001F638CA015D2E414744E0C5C2E0F5F827DFC`

Command Data CRC: `CEE700A0`

Request:
`882001020001042 53AB91A21859A06813E8A880E10BA0C67A09DDBED0B7E001F638CA015D2E414744E0C 9000`

Secure Element Specific APDU Error Codes

This table contains the expected error codes and descriptions that a caller may encounter while working with the Secure Element Applet.

Error Code	APDU Command	Description	Error Code to POS
0x6301	Sign Invoice	PIN verification required before executing a command	1500
0x6302	Verify PIN	PIN verification failed – wrong PIN code	2100
0x6303	Verify PIN	Wrong PIN size	2100
0x6304	Sign Invoice	Maximum number of tax categories exceeded	SDC related
0x6305	Sign Invoice	Secure Element amount has reached the defined limit. The Secure Element is locked and no additional invoices can be signed before the audit is completed.	2210

0x6306	End Audit	End Audit is sent but there is no active Audit	SDC related
0x6307	Sign Invoice	Invoice fiscalization is disabled by system	2210
0x6308	Sign Invoice	Invoice DateTime must be within Certificate validity NotBefore and NotAfter	SDC related
0x6310	Verify PIN	The number of allowed PIN entries exceeded	2110
0x63FF	Sign Invoice	A Secure Element counter has reached its limit. The Secure Element must be replaced.	SDC related
0x6700	End Audit	Data must be 256 bytes long	SDC related
0x6A80	End Audit	Proof of Audit command payload provided as APDU Command Data does not match the latest Start Audit one which Secure Element expects. Probably a new Start Audit was initiated after this one was ended.	SDC related
0x6A80	Sign Invoice	The tax category order id exceeds the maximum allowed for the Secure Element.	2310
0x6F00	End Audit	APDU Command Data cannot be recognized as a valid Proof of Audit	SDC related

File-Based Communication

Introduction

This section contains the description of the **File-based** communication with E-SDC.

File-based communication between TaxCore.API and E-SDC is foundation of Local Audit process.

General structure for storing files on removable memory units:

Removable memory root

- UID (folder)
  - audit packages (file)
  - UID.arp (file)
- UID.commands (file)
-

UID.results (file)

NOTE:
The above structure displays files for all individual file transfers (copy audit files from E-SDC, upload commands to E-SDC, and transfer commands result from E-SDC). It is used here for explanatory purposes. Each individual file transfer includes only the files specific for that transfer.

Content

SD Cards or Flash Memory Drives Format

Each E-SDC shall work with the following file system formats of SD Cards and USB Flash drives:

- FAT
- FAT32
-

NTFS

E-SDC is Configured Using Initialization Commands from an SD Card

Commands File

JSON file containing all pending commands must be stored in the root of the external disk volume and named **{UID}.commands** (e.g `D:\\YJ37C9Z9.commands`)

Command file may be generated and downloaded from either Taxpayer Admin Portal (by taxpayers) or the internal back office portal (by tax authority officers).

Commands File Format

Command file content must be formatted as valid JSON file.

```
[
    {
      "commandId": "GUID",
      "type": 0,
      "payload": "Command Specific Json as string",
      "uid": "string"
    }
]
```

End Of Line Characters

E-SDC must be able to process commands file using any standard EOL characters.

E-SDC Stores Audit Files on SD Card or USB Drive

An E-SDC shall perform an audit automatically once an SD Card or USB drive is inserted. If any commands are received on the same medium, they shall be executed **before** the proceeding with the Local audit.

NOTE:
Audit files must remain stored in E-SDC's local memory until it receives a Proof-of-Audit command from TaxCore.API

All files shall be stored in the folder(s) named after the UID(s) of the secure element(s) which created them.

Example: `G:\BJ3PN1S9\`, where G is the root of the SD card/USB drive and contains audit packages created by the secure element with the UID BJ3PN1S9.

If the folder(s) do not exist, an E-SDC shall create new one(s) - one for each UID.

All audit package files stored in E-SDC's local memory, no matter which secure element was used to create them shall be copied to the appropriate folder(s). Audit package files must be named using the following convention: `{UID}-{UID}-{Ordinal_Number}.json`.

Depending on whether the smart card secure element is connected to the E-SDC, the folder named after its UID may contain one `{UID}.arp` file. The content of the `{UID}.arp` file is described in Submit Audit Request Payload - ARP.

File structure for this transfer should look like this:

Removable memory root

- UID folder
  - audit package files
  - UID.arp

NOTE:
Dumped audit package files must be created by secure element(s) from the appropriate target environment, e.g. if the Local Audit is performed for the production environment, only audit package files created by the production environment secure elements can be dumped on an SD Card or USB Drive. This is done in order to avoid uploading audit packages to environments which do not recognize the UIDs created for different environments.

E-SDC Executes Commands Received via SD Card or USB Drive

An E-SDC shall process commands automatically upon insertion of SD Card or USB Flash drive. Command execution takes precedence over a Local audit.

JSON file containing all pending commands must be stored in the root of the external disk volume and named **{UID}.commands** (e.g `D:\\YJ37C9Z9.commands`).

File structure for this transfer should look like this:

Removable memory root

- UID.commands

E-SDC shall execute only those commands with the same UID as UID assigned to the digital certificate of the Secure Element (stored in the *SerialNumber* field of the certificate subject).

Command types and the structure are explained in section Commands.

E-SDC Stores a Command Execution Result to the SD Card or USB Drive

After commands have been executed, E-SDC must generate a JSON file named **{UID}.results**. The result must be in format described in paragraph *Commands Execution Results* in Commands. It is stored in the root folder on the SD Card/USB drive.

Example: **G:\BJ3PN1S9.results**

where **G is the root of the SD card/USB drive** and **BJ3PN1S9 is an example UID** of the smart card in use

If file with the same name exists on the SD Card/USB drive, it must be overwritten.

File structure for this transfer should look like this:

Removable memory root

- UID.results

Manuals

E-SDC must have a user manual that explains the following topics in detail:

1. System requirements
2. Installation instructions for the technicians performing the installation and integration of an E-SDC device

or software at a sales point

3. Properly connecting the E-SDC to a POS
4. User instructions for the operator (cashier or shopkeeper) explaining normal operations in detail
    - Information about the supported card readers or integrated card reader
    - How commands are received and executed via USB drive or SD card
    - How to obtain help/support regarding the product
    - Steps for troubleshooting
    - How the product license is activated
    - Local and/or remote audit instructions
    - How to properly shut down or uninstall the product
5. A detailed explanation (with visuals) for each product notification
6. Explanation of each custom error (manufacturer-specific)

For POS Developers

Introduction

This article offers useful initial information about technical requirements for accrediting POS solutions.

NOTE:
Before reading the rest of this article, make sure you have read Getting Started With Accreditation
Also, before you start developing your solution, please read General Information for all vendors.

This article offers POS vendors who are interested in accrediting a POS solution the following insight:

- how a POS solution fits in with other EFD components?
- how to navigate the rest of the technical instructions in order to initialize development?

Integration with other EFD components

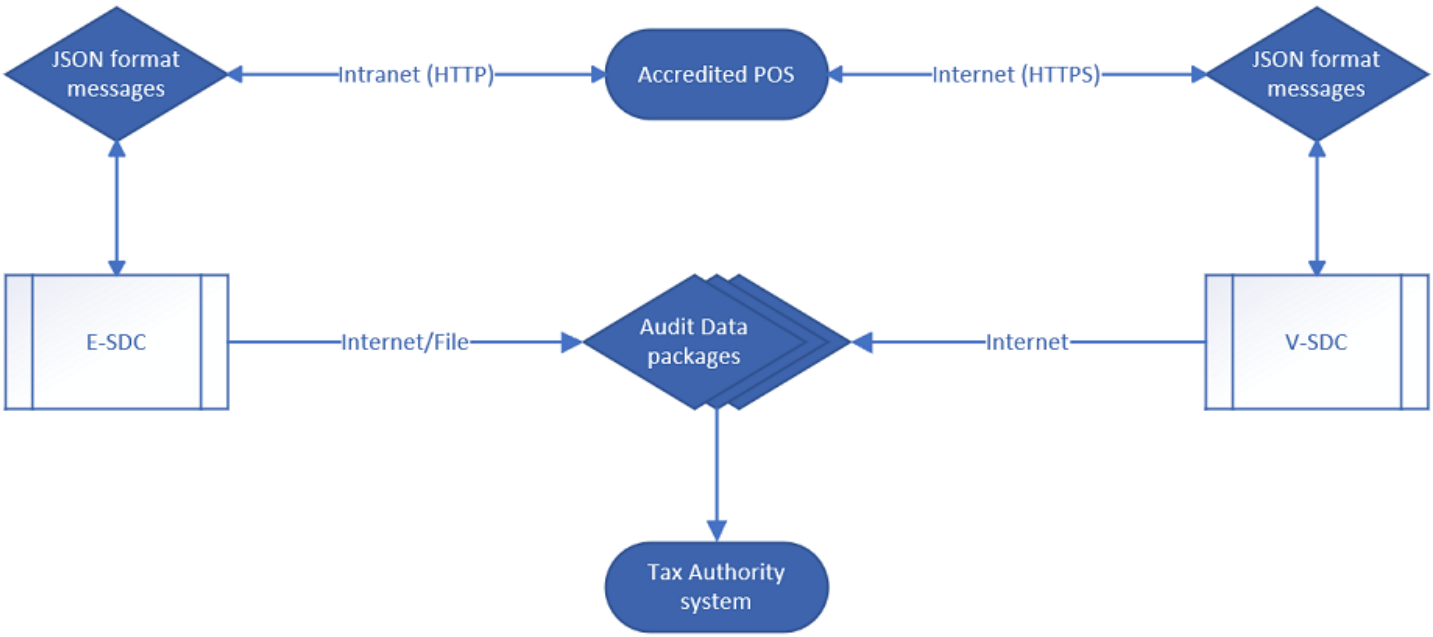
POS is one of three components of any EFD setup.

NOTE:
For an overview of all EFD components and how they communicate with each other, see Electronic Fiscal Device.

POS can work with just one SDC service or with both V-SDC and E-SDC (in which case they are used alternately, depending on the internet availability).

For more details about the differences between the two SDC options, as well as fiscalization recommendations for different business-type scenarios, see:

- Choosing an Appropriate Model
- Recommendation Examples



What can be accredited as a POS solution?

There are many options when it comes to the type of product that can be accredited as a POS solution - the main rule is that it fulfills all technical requirements contained in these instructions. The options include, but are not limited to:

- Standard cash registers
- ERP systems
- Middlewares that serve as a link between an invoicing system and an SDC service
- Mobile applications
- Web applications, etc.

What is the typical process flow of POS operations?

All accredited EFD solutions must follow the basic steps in the process of creating fiscal invoices (although some might have additional, manufacturer-specific, steps).

For a list of these basic steps, see Typical Process Flow.

Connecting with V-SDC service

Issuing fiscal invoices via a V-SDC service requires an internet connection. For more details about the process, see Connected Scenario.

Advantages of V-SDC service

- No specialized hardware
- POS can be implemented as a mobile app
- Compliance of the existing ERP system can be done quickly
- Cost of taxpayer fiscalization is reduced

Automatic audit

Disadvantages of V-SDC service

- Requires internet connection in order to create fiscal invoices

You also use the V-SDC service to issue fiscal invoices with an online POS solution. For more information, see Online POS and V-SDC Integration.

Communication between a POS solution and V-SDC is established using this POS to SDC protocol.

Connecting with E-SDC service

Issuing fiscal invoices via an E-SDC service can be performed both with or without an internet connection. For more details about the process, see Semi-Connected Scenario.

Advantages of E-SDC service

- Enables issuing fiscal invoices without internet connection

Disadvantages of E-SDC service (if implemented as a hardware/black box solution)

- Increases the cost of taxpayer fiscalization
- If implemented as a hardware/black box solution:
    - Requires a specialized/dedicated hardware
    - Prone to physical damage
    - May require a specialized/dedicated hardware maintenance

Communication between a POS solution and E-SDC is established using the following POS to SDC protocol.

What are the data formats that POS sends and receives?

The formats of all data exchanged between a POS and an SDC service (V-SDC or E-SDC) is described in section Data Formats.

Useful test cases

Please refer to section Test Cases for both standard and special test cases.

All sections of Technical Instructions for POS vendors

Technical instructions specific for POS vendors/developers consist of the following sections:

1. Quick Start

    1. **Register as a vendor** for the Sandbox environment
2. Choosing an Appropriate Model
   The following explanation should help you decide which fiscalization model is the most appropriate for your clients.

3. Typical Process Flow
   This section describes a typical process flow for successful fiscalization scenarios via V-SDC and E-SDC.

4. Connected Scenarios
   In this scenario POS connects to V-SDC and performs instant fiscalization of invoice using web service.

5. Semi Connected Scenarios
   Some jurisdictions may require taxpayers to connect and submit data from their E-SDC on predefined periods of time using any type of audit.

6. Data Formats
   This section describes the main data formats used during fiscalization.

7. Protocols
   Each POS or Invoicing System must communicate with either V-SDC or E-SDC to fiscalize invoices.

8. 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 creates **Invoice Requests**, and submits them via the HTTPS protocol directly to V-SDC API, using the **digital certificate** issued to Taxpayer. This process completes invoice fiscalization with a signed invoice returned to online POS.

9. How Tos
   This section contains articles, how-tos and cheat sheats that address particular aspects of POS development and operation in context of TaxCore solution

10. Test Cases
    Regardless of the type of invoicing system you are building, the same test cases apply:

Related articles

- EFD Vendors General Information

Quick Start

Quick step-by-step guide for POS accreditation

1. **Register as a vendor** for the Sandbox environment
2. **Receive a Developer Certificate** and use it to access the Developer Portal
3. **Use the Developer Portal** to request additional certificates for development and testing purposes if required
4. **Consult** all the sections in these **technical instructions** to see understand all the requirements and how they should be implemented
5. Use the applications and services on the Developer Portal to **develop, test and accredit your POS** application
    - Dev-ESDC for testing POS operation with E-SDC service
    - VSDC Request Submitter for testing POS operation with V-SDC service
6. **Compile user documentation** for your POS

7. Use the My Accreditations section on Developer Portal to **apply for accreditation** of your POS.

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	SignInvoice, GetTaxAuthorityParams	VerifyPIN, SignInvoice, AttentionGetStatus, GetSignedInvoice, GetTaxAuthorityParams
Authentication method	*PAC/PIN	**PIN code
SignInvoice method (Invoice Request)	Same as E-SDC + PAC	Same as V-SDC
Find out what is the latest signed invoice in case of communication failure	YES	YES
Internet connection required to work	YES	NO

*PAC code is sent to the 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 a PIN code to be sent through the VerifyPIN method in order for ESDC to be activated

Differences Between V-SDC and E-SDC

V-SDC	E-SDC

V-SDC uses the HTTPS protocol.	E-SDC uses the HTTP protocol.
An authentication certificate is required to establish communication between a POS and V-SDC.	No Authentication certificates are required to establish communication between a POS and E-SDC.
API methods used for V-SDC are: *SignInvoice* and *GetTaxAuthorityParams*.	API methods used for E-SDC are: *VerifyPIN*, *SignInvoice*, *Attention*, *GetStatus*, *GetSignedInvoice* and *GetTaxAuthorityParams*.
V-SDC requires sending a PAC code as a property in the *InvoiceFiscalizationRequest*, as a two-factor authentication method (PAC Code and POS PFX Certificate).	E-SDC does not require sending a PAC code as a property in the *InvoiceFiscalizationRequest*. Instead, E-SDC requires sending a PIN code through the VerifyPIN method in order for ESDC to be activated.
SignInvoice methods for both V-SDC and E-SDC are identical, except when a PAC code is required if the authentication between POS and V-SDC is established through the POS PFX certificate.	HASH property of *InvoiceFiscalizationRequest* or *GetSignedInvoice* is designed only for E-SDC, and not for V-SDC.
V-SDC is a cloud-based service, therefore a POS requires an available internet connection in order to sign invoices.	E-SDC is a semi-online-based solution; therefore, it does not require an available internet connection to sign invoices. E-SDC can be connected to the local network (LAN cable or Wi-Fi) or directly to the POS via a LAN cable.

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

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 the device generally depends on the 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 (E-SDC) 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 (E-SDC) 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.

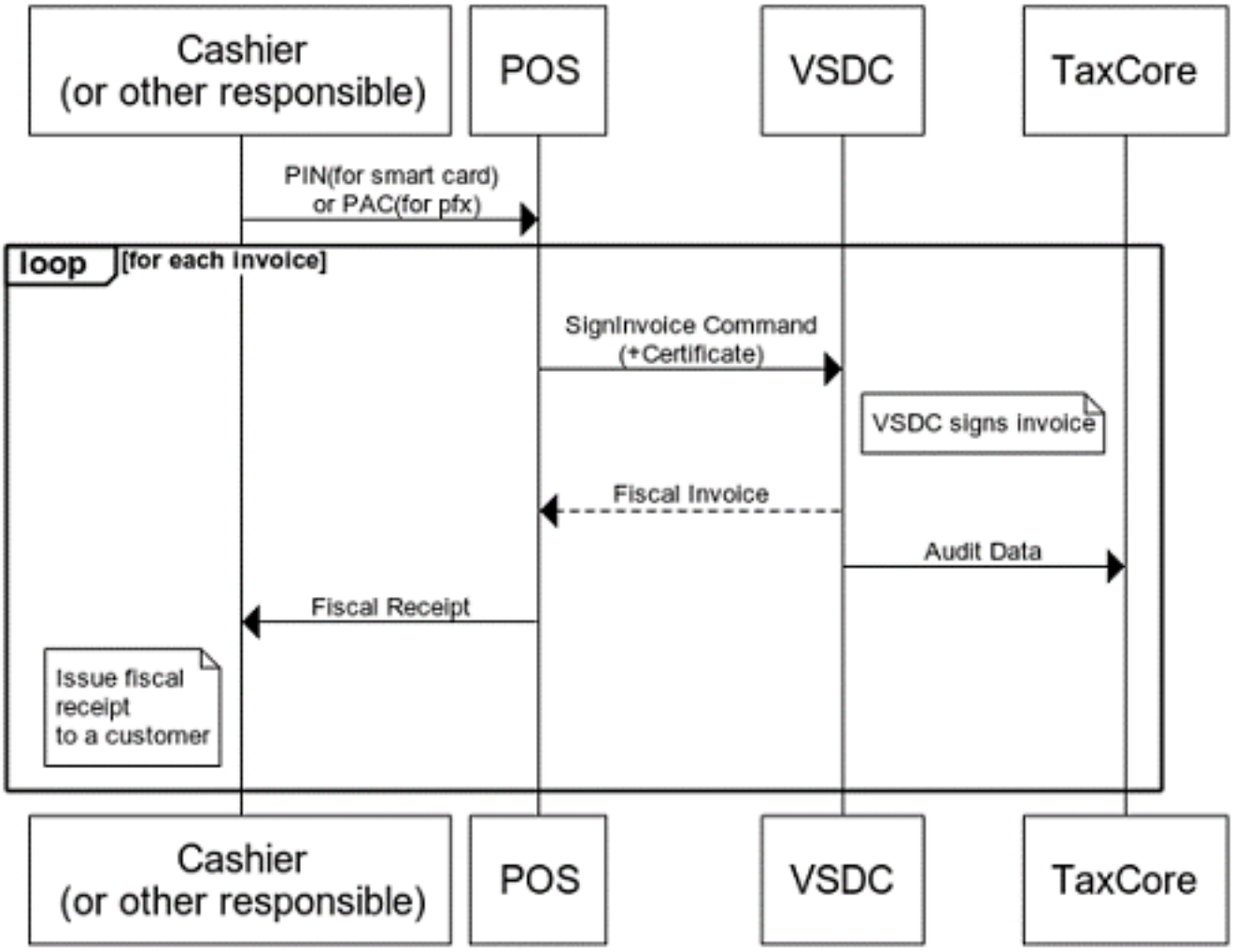
Web Shops

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.

Typical Process Flow

This section describes a typical process flow for successful fiscalization scenarios via V-SDC and E-SDC.

V-SDC process flow



Provide a valid PIN/PAC

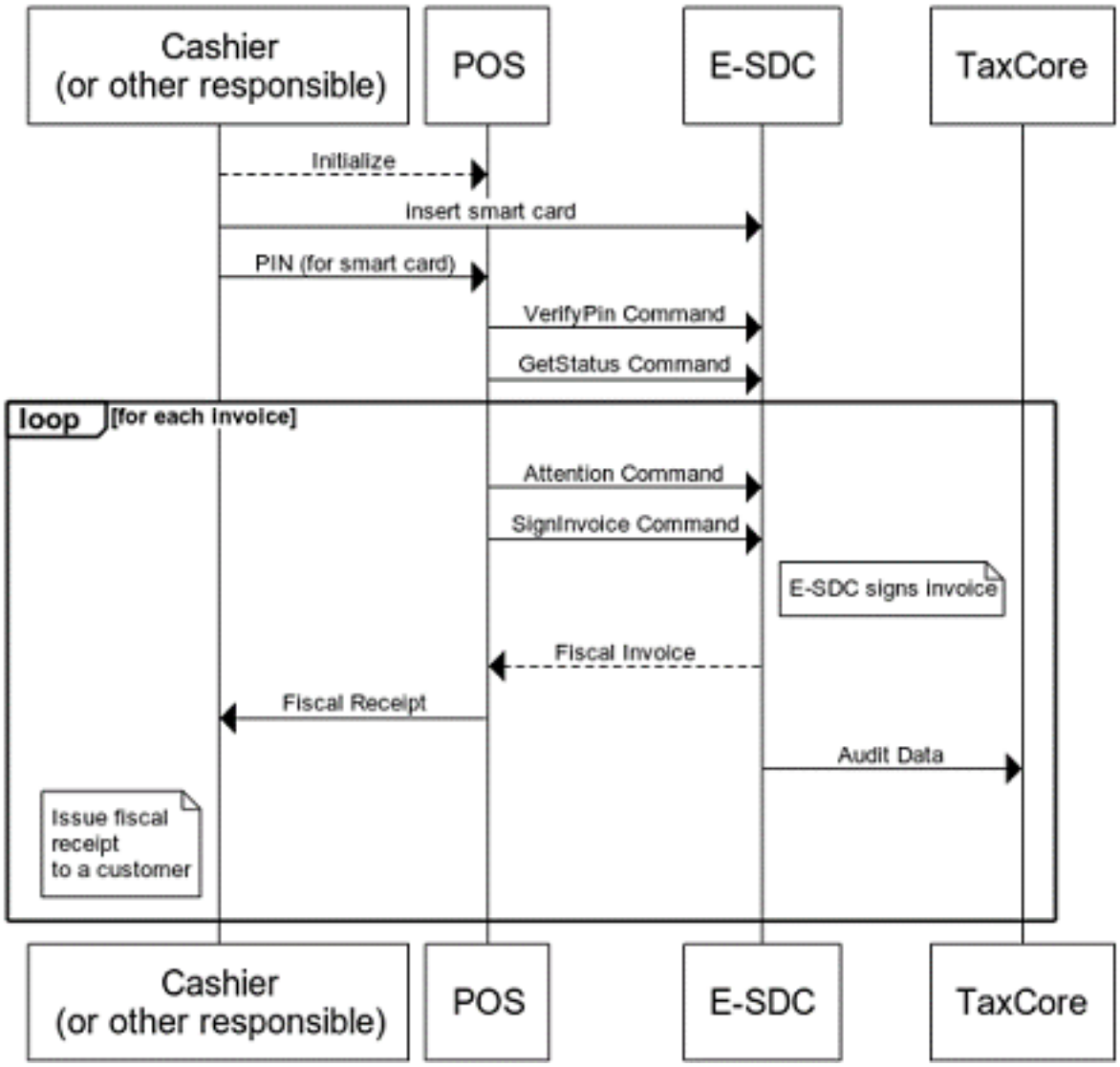
The process begins with a cashier creating an invoice request on an accredited POS. An invoice request transmits the transaction data (POS date and time, list of items, transaction type, payment type, etc.) to an available V-SDC.

When sending the request to a V-SDC, the cashier needs to provide a valid PIN (form smart cards) or PAC (for .pfx certificates) to confirm the authenticity of the certificate.

Fiscalization via V-SDC

In short, V-SDC receives the transaction data from POS, fiscalizes it, returns the fiscalized invoice back to POS and submits the audit data to TaxCore.

E-SDC process flow



Provide a valid PIN

The process begins with a cashier creating an invoice request on an accredited POS. An invoice request transmits the transaction data (POS date and time, list of items, transaction type, payment type, etc.) to the connected E-SDC.

When sending the request to an E-SDC, the cashier needs to provide a valid PIN (form smart cards) to confirm the authenticity of the certificate.

Fisacalization via E-SDC

Just like V-SDC, an E-SDC service also receives the transaction data from POS, fiscalizes it, returns the fiscalized invoice back to POS and submits the audit data to TaxCore - however, it offers an additional option to do all this with an internet connection.

The process involves the following steps:

1. Cashier enters the required information via an accredited POS
2. POS generates a request data and sends it as a request to the E-SDC using JSON via HTTP protocol;
3. E-SDC verifies the format of the invoice;
4. E-SDC calculates taxes based on the current tax rates;
5. E-SDC sends the invoice data to the Secure Element for fiscalization providing the current date and time and PIN code/password if required;

6. Secure element signs the invoice and returns the data to the E-SDC;
7. E-SDC produces a journal file – a textual representation of an invoice;
8. E-SDC generates a verification URL;
9. [optionally] E-SDC creates a QR Code – a graphical representation of a verification URL;
10. E-SDC creates an invoice with all mandatory elements (receipt data, previously generated signature, verification URL and journal), generates a one-time key and encrypts the invoice using a symmetric algorithm. The E-SDC encrypts a one-time symmetric key using the tax authority's system public key and adds it to the package so the TaxCore system shall decrypt the symmetric key and access the package content once it arrives to the TaxCore system.
11. If the internet connection is available, E-SDC sends the Audit Data to TaxCore
12. E-SDC returns the Invoice Fiscalization Result (Journal + QRCode/Verification URL) back to the POS
13. The POS prints the fiscal invoice and the Cashier gives it to the customer (or sends an electronic invoice to the customer)

Transaction Report on Invoicing System

Depending on the tax jurisdiction requirements, an Invoicing System (POS) must be able to generate and print a transaction report of all Normal and Advance invoices. The invoicing system must allow the operator to select the period for displaying the report.

NOTE:
Always check whether the tax jurisdiction where you wish to accredit your product has a requirement for generating transaction reports.

The report should contain the following information:

- Date and time of the report period start and end
- The number of issued invoices per invoice type
    - Normal Sale
    - Normal Refund
    - Advance Sale (if issued by the invoicing system)
    - Advance Refund (if issued by the invoicing system)
- The list of sold items - quantity, unit price, and total price
- Total amount on the Sale (Normal and Advance) invoices
- Total amount on the Refund (Normal and Advance) invoices
- Total amounts per payment type
    - cash
    - card
    - check
    - wire transfer
    - mobile money
    - voucher
    - other

Client Authentication

Introduction

Communication between POS and SDC must be secure and protected from any unauthorized use. Depending on type of Secure Element, specifics of the POS and SDC in use there are different approaches to secure communication between those two parties.

Each POS should support all three scenarios:

1. POS to V-SDC using digital certificate distributed in file format
2. POS to V-SDC using using digital certificate distributed on smart card
3. POS to E-SDC on the local network

POS Connects to V-SDC

POS and V-SDC API communication requires mutual authentication of client (POS) and server (V-SDC). Mutual authentication between parties is conducted using client and server digital certificates.

POS is required to use Client Certificate Authentication with each request targeting V-SDC API.

Secure Elements (and associated digital certificates) may be distributed in two formats

1. Files (PKCS12 *.pfx or *.p12)
2. Smart Cards

Authentication against V-SDC using digital certificate distributed in file format

In this scenario POS is using digital certificate installed in the local certificate store (key chain) for client authentication.

To improve security caller is required to submit PAC associated with used secure element as part of the HTTP request. More info is available in SDC protocol section.

Authentication against V-SDC using digital certificate distributed on smart card

If POS is using secure element from the locally attached smart card for client side authentication PAC is **not** required but user will be automatically prompted to enter PIN code for the selected smart card by Operating system to finalize client-side authentication against V-SDC server.

POS Connects to E-SDC

Communication between POS and E-SDC available on local machine or on the LAN must be safeguarded using network-level security.

E-SDC will require PIN code of the Secure element to enable fiscalization-related operations on the Secure Element Applet used by E-SDC.

Connected Scenarios

Connected Scenario is preferred way of communication between POS and SDC.

In this scenario POS connects to V-SDC and performs instant fiscalization of invoice using web service.

1. Accessing V SDC API
   Once valid Test certificate(s) are obtained, you can access the V-SDC.Api description on the following URL: [[*TaxCore.PublicConfiguration.VSDCApiUrl*]]

2. 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.

Accessing V-SDC API

Once valid Test certificate(s) are obtained, you can access the V-SDC.Api description on the following URL: [[*TaxCore.PublicConfiguration.VSDCApiUrl*]]

NOTE:
The following process applies only to the .pfx digital certificates, and not to smart card certificates.

To extract VSDC.Api URL from a certificate, follow these steps:

1. Open the .pfx certificate installed on your computer
2. In the **General** tab, your certificate should have one of these OIDs **1.3.6.1.4.1.49952.X.Y.3.7**
3. X and Y parameters identify the environment and their values change according to each environment.

4.

Click on the **Details** tab and find the line with an OID in this format - **1.3.6.1.4.1.49952.X.Y.7**

5. Again, X and Y parameters identify the environment, and the values will vary according to different environments, but they will be the same as on the **General** tab
6. Number **7** at the end identifies the V-SDC.Api URL
7. Read the value of this OID containing the URL of V-SDC.Api

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 authority, you will need to provide PAC value as an HTTP header to each request.

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

static void Main(string[] args)
{

    string invoiceRequest = @"{
  "dateAndTimeOfIssue": "2020-12-08T08:55:23.286Z",
  "cashier": "123456789",
  "buyerId": "RS34564565",
  "buyerCostCenterId": "567546",
  "invoiceType": "Normal",
  "transactionType": "Sale",
  "payment": [
    {
      "amount": 70.00,
      "paymentType": "Cash"
    }
  ],
  "invoiceNumber": "POS2017/998",
  "options": {
    "omitQRCodeGen" : "1" ,
    "omitTextualRepresentation" : "0"
  },
  "items": [
    {
      "name": "Sport-100 Helmet, Blue",
      "quantity": 2,
      "unitPrice": 34.23,
      "labels": [
        "A"
      ],
```

```
      "totalAmount": 68.46
    }
  ]
}";

    var httpContent = new StringContent(invoiceRequest,Encoding.UTF8,"application/json");
    HttpClient client;
    WebRequestHandler handler;

    GetClientAndHandler(out handler, out client);

    var response = client.PostAsync($"/api/v3/invoices", 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();
    client.DefaultRequestHeaders.Add("PAC", "123456");
}

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

Semi-Connected Scenario enables sale point to stay operational and issue fiscal invoices without permanent or reliable interent connection for prolonged periods of timee.

Some jurisdictions may require taxpayers to connect and submit data from their E-SDC on predefined periods of time using any type of audit.

If Internet connection and TaxCore.API are available to E-SDC at lease for couple of minutes per day it is usually enough time to submit all pending audit packages and comply with maximum audit period requirements.

For more information please refer to Semi-Connected Scenario.

Data Formats

This section describes the main data formats used during fiscalization.

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

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

3. Buyer TIN on Export Invoices
   A Tax Identification Number (TIN) is an identifier used in many countries to uniquely identify each taxpayer.

Date and Time

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

The date and time generated by E-SDC or V-SDC and printed on a receipt is local 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).

NOTE:
Be mindful of the minimum (1900-01-01T00:00:01Z) and maximum (9999-12-31T23:59:59Z) values.
Also, individual tax jurisdictions might have specific requirements regarding the maximum difference between

the time of invoice creation (SDC time) and the moment when an audit package arrives to СУФ. Make sure you are familiar with those requirements.

How SDC Calculates Taxes

Introduction

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

How does E-SDC Calculate Taxes?

The 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 the 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.

For more information, see article Calculate Taxes in the section of documentation for E-SDC vendors.

Rounding

An SDC (both E-SDC and V-SDC) rounds all amounts to 4 decimal places, using the half-round up method.

The rules for this are simple:

- Decide which is the last digit to keep (in this case, the fourth decimal place)
- Leave it the same if the next digit is less than 5
- But increase it by 1 if the next digit is 5 or more

Examples:

3.44445555666 → 3.4445

3.4440012345 → 3.4440

3.44466012345 → 3.4447

3.444116012345 → 3.4441

Any amount shall be rounded to 2 (two) decimal places, using the half-round up method, **only** on the textual representation of an invoice.

NOTE:
In SDC's internal memory, the amount will always be stored with 4 decimal places (if the amount has 4 or more decimals), regardless of invoice's textual representation with 2 decimal points.

Buyer TIN on Export Invoices

Introduction

A Tax Identification Number (TIN) is an identifier used in many countries to uniquely identify each taxpayer.

Every time when a taxpayer issues a **B2B fiscal invoice**, it **must contain a valid Buyer TIN** which uniquely identifies the buyer of sold goods/services.

However, when issuing a B2B fiscal invoice for exported goods/services (i.e. when goods/services are sold to a buyer from a foreign country), the cashier **must enter a Buyer TIN which starts with an official country code prefix**.

NOTE:
This rule might not apply in every tax jurisdictions. Make sure you become familiarized with each jurisdiction's legal requirements.

How to format TIN

Country code prefix is an **ISO 3166-1 alpha-2 (2 letters) country code**. It identifies the buyer's country of origin.

The complete and up to date list is available on https://www.iso.org/iso-3166-country-codes.html

The rest of the TIN is composed of numeric digits in most countries, but in some countries, it may contain letters.

When issuing a B2B invoice that is not for exporting goods/services, the country code prefix can be included in the Buyer TIN, but it is not mandatory.

Examples - Valid formatting

- RS123456789
- RS-123456789
- RS-12345-6789
- RS 123456789

Example of the printed B2B invoice

```
============ FISCAL INVOICE ============
TIN:                            980361508
Company:          Compete Enterprises Inc
Store:            Compete Enterprises Inc
Address:                    50 Via Del Sol
District:                               Ba
Cashier TIN:
Buyer TIN:                   RS1234567890
Buyers Cost Center:                1256KSR
POS Time:             14/07/2021 08:53:56
--------------NORMAL SALE---------------
Items
========================================
Name     Price         Qty.           Total
Komp 12e (A)
       1200.00           1          1200.00
----------------------------------------
Total Purchase:                    1200.00
Payment Method:                       Cash
========================================
Label         Name      Rate            Tax
A              VAT     9.00%          99.08
----------------------------------------
Total Tax:                           99.08
========================================
SDC Time:             14/07/2021 08:53:58
SDC Invoice No:       HQBK79BT-F6MYL8UM-2
Invoice Counter:                    2/2NS
========================================

[QR Code]

======== END OF FISCAL INVOICE =========
```

Buyer TIN on Export Invoices

Protocols

Introduction

Each POS or Invoicing System must communicate with either V-SDC or E-SDC to fiscalize invoices.

The communication protocol is standardized and public. API Documentation is available as part of this documentation.

Each E-SDC is required to implement and expose documented API to all POS and Invoicing systems. Correctness

of API implementation is, among other criteria, one of the requirements for E-SDC accreditation.

Each POS must connect to SDC to issue a fiscal invoice. Depending on the business and technical requirements POS can connect using one of the following approaches:

1. **POS connects to Taxcore.Api** to get environment configuration and tax rates [Optional, but highly recommended]
2. **POS connects to Development E-SDC**. This is used for development, testing and accreditation purposes only. Development E-SDCs are provided as web service to all POS vendors to enable development without the need to possess real E-SDC applications or devices.
3. **POS connects to V-SDC** using a digital certificate delivered in **PKCS12** format and PAC to authenticate to V-SDC.Api
4. **POS Connects to V-SDC** using digital certificate delivered on **smart card** and PIN Are used to authenticate to V-SDC.Api
5. **POS Connects to E-SDC**. Communication between POS and E-SDC is secured on a network level. Fiscalization of the invoice is performed by E-SDC and Secure Element inserted into E-SDC

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 creates **Invoice Requests**, and submits them via the HTTPS protocol directly to V-SDC API, using the **digital certificate** issued to Taxpayer. This process completes invoice fiscalization with a signed invoice returned to online POS.

Online POS **submits requests to V-SDC API directly**, in order to create an HTTPS 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 HTTPS requests initiated from within the scripts. TaxCore offers a solution for online POS, which will 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.

1. 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:

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

3. [Supported Browsers]
   Online POS and V-SDC are tested with the following browsers:

4. [Example of Integration Using a Simple HTML Page]
   The following code is an example of 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.

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>
```

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

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

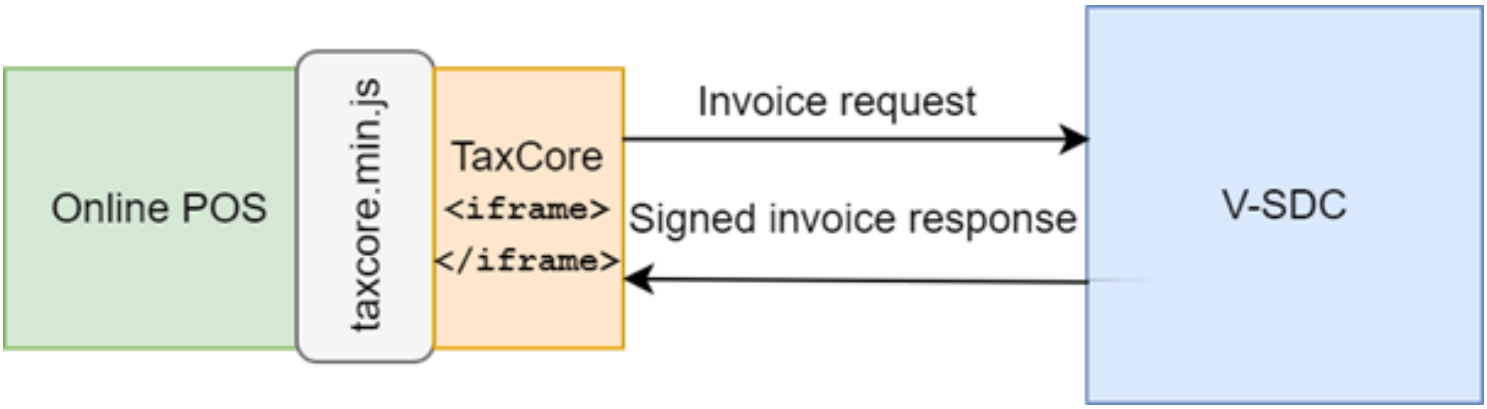
NOTE:
Both components must exist at the integration point for successful integration. JavaScript will first try to find taxcore sign element with id="taxcore_sign_element". If it can't be found, the exception will be thrown with an appropriate message (e.g. "*Could not find taxCoreElement. Check if you 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 (https://www.w3.org/TR/cors/#cross-origin-request-with-preflight-0).

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.



Detailed Specs - Image of 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.

TaxCore Sign Element

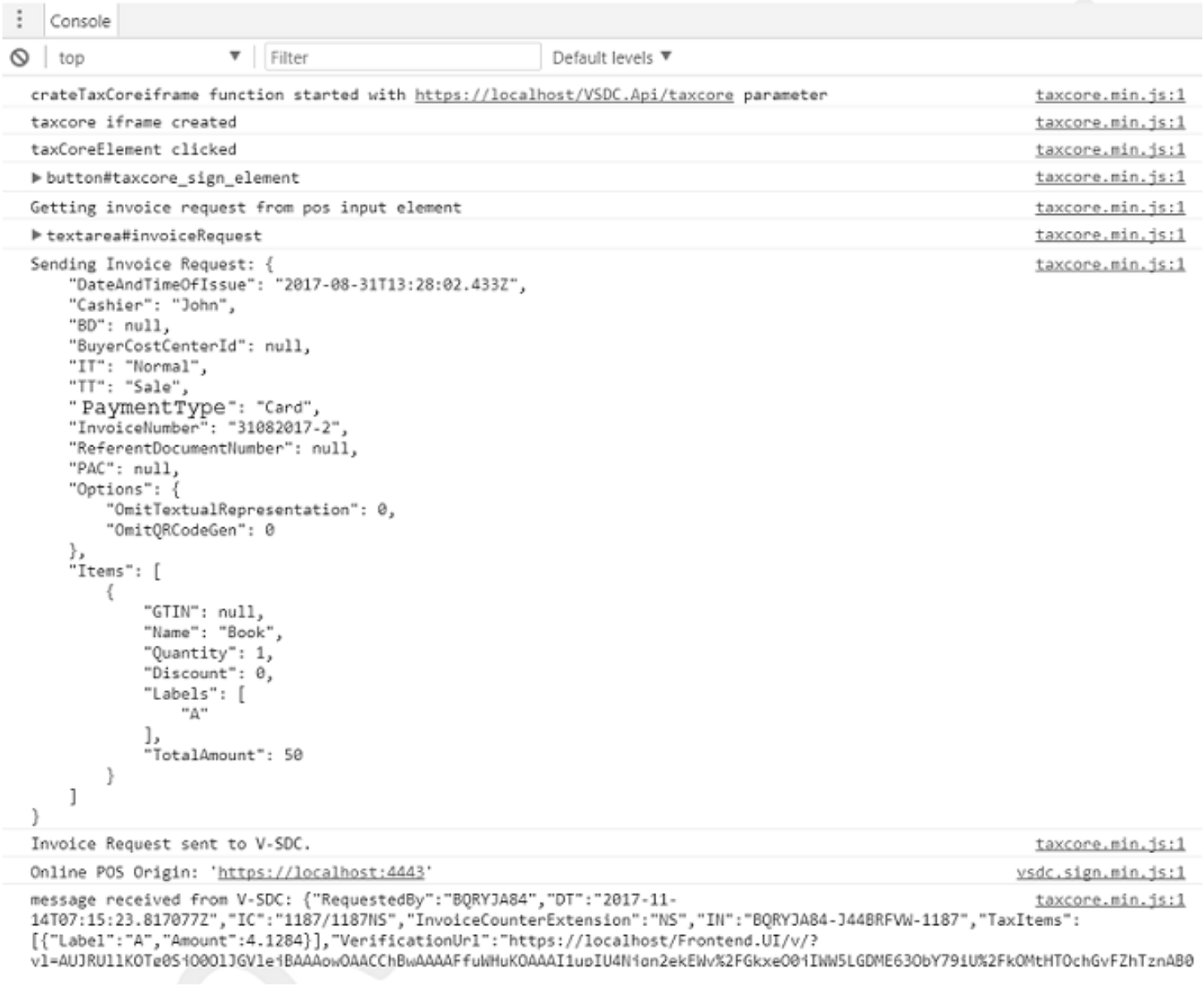
TaxCore Sign element is nothing more than an HTML tag with predefined id, that you'll place at your online POS page. Its task is to collect invoice data and forward them to V-SDC.

> **Note:** The id of TaxCore Sign Element **must** be equal to the 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-response 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.



Detailed Specs - Image of the sign-in element

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+

Example of Integration Using a Simple HTML Page

The following code is an example of 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": "2020-12-08T08:55:23.286Z",
  "cashier": "123456789",
  "buyerId": "RS34564565",
  "buyerCostCenterId": "567546",
  "invoiceType": "Normal",
  "transactionType": "Sale",
  "payment": [
    {
      "amount": 70.00,
      "paymentType": "Cash"
    }
  ],
  "invoiceNumber": "POS2017/998",
  "options": {
    "omitQRCodeGen" : "1" ,
    "omitTextualRepresentation" : "0"
  },
  "items": [
    {
      "name": "Sport-100 Helmet, Blue",
      "quantity": 2,
      "unitPrice": 34.23,
      "labels": [
        "A"
      ],
      "totalAmount": 68.46
    }
  ]
};

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

How Tos

This section contains articles, how-tos and cheat sheats that address particular aspects of POS development and operation in context of TaxCore solution

1. Issuing Copy Invoice In Case of Power Outage or Printer Spooler Failure
   Issuing an invoice copy to a customer is obligatory, this guarantees that the consumer always knows what they've purchased, and it is in line with the rules of the local tax authority.

2. How to Determine the Tax Amount Before Issuing a Sale Invoice?
   In an Invoice Request, a POS sends the **'TotalAmount'** which represents the gross amount of the products/services sold. This reflects the amount that a customer needs to pay before sending the API call to an SDC.

3. Handling Payments on Automatic Payment Stations
   Very often, businesses employ automatic payment stations to make it easier and faster for the customers to pay whatever they might be buying. This is most frequently the case with gas stations where customers don't necessarily need to physically enter in order to pay (unless they wish to buy food or other goods).

4. How to Print a QR Code Using a Dot Matrix Printer?
   Digitalizing the tax collection system comes with a lot of benefits for both consumers and businesses. One of the best solutions TaxCore came up with is allowing a customer to easily and quickly verify their receipt.

5. Printing Receipts Is Not the Only Option
   TaxCore solution doesn't require taxpayers to necessarily print receipts. It offers one more options better suited for the digital age.

Issuing Copy Invoice In Case of Power Outage or Printer Spooler Failure

Issuing an invoice copy to a customer is obligatory, this guarantees that the consumer always knows what they've purchased, and it is in line with the rules of the local tax authority.

Unfortunately, a business can suffer a sudden power outage, communication between POS and SDC may fail for variety of reasons and printer paper may jam in the middle of operation. In this case, the situation doesn't permit issuing an invoice copy to a customer. The solution is simple – it is enough to have the reprint option, which can be manual or automatic.

Most importantly, the copy needs to be processed by either E-SDC or V-SDC if one is available. Please note that

TaxCore as a system allows that invoices do not require a physical copy in order to be verified.

In case a business has a loyalty program, it can send invoices to customers personal e-mail addresses. The digital copies of invoices can also be sent via different messaging apps, such as WhatsApp, Viber, and WeChat.

In case the problem is not the power outage but a printer spooler failure, there is a solution for this too. A customer can be offered a QR code on the customer-facing display which they can scan using their smart-phones. This will provide them with the URL with an exact digital copy of their receipt.

TaxCore allows many features and solutions to businesses and taxpayers, and these should be taken advantage of whenever different solutions are necessary.

How to Determine the Tax Amount Before Issuing a Sale Invoice?

In an Invoice Request, a POS sends the **'TotalAmount'** which represents the gross amount of the products/services sold. This reflects the amount that a customer needs to pay before sending the API call to an SDC.

In Invoice Response, the POS will receive a breakdown of tax amounts as per labels used in the Invoice Request.

POS can learn in advance what is the valid tax rate, exposed by the Tax Authority by using the Get Environment Parameters method.

If anyone wishes to know in advance what will be the tax portion after fiscalization, the instructions on how an SDC (External or Virtual) calculates taxes can be found here - Calculate Taxes.

For any other reason, a POS can always print a Proforma Sale (PS) invoice to obtain a so-called 'quotation'. This will tell the cashier exactly what the tax portion will be prior to concluding the sale. Just remember that every PS should be referenced in the Normal Sale (NS) invoice which makes the transaction conclusive. For more information on document referencing see Anatomy of a Fiscal Receipt.

Handling Payments on Automatic Payment Stations

Very often, businesses employ automatic payment stations to make it easier and faster for the customers to pay whatever they might be buying. This is most frequently the case with gas stations where customers don't necessarily need to physically enter in order to pay (unless they wish to buy food or other goods).

If a customer doesn't physically enter the shop to pay for the gas, how will they receive a fiscal receipt? In this case, an OPT (Outdoor Payment Terminal) is installed where customers can pay using both cash and debit/credit cards.

This still means that all the transactions done at the station must be sent to СУФ, regardless of whether POS and

cashier were directly involved or not (in case of an OPT). It is mandatory by law that each customer receives a verifiable invoice (receipt) of the goods or services they've purchased.

OTPs are usually equipped with a printer, which means that customers can easily receive a printed version of an invoice in case they paid for fuel, for example. In case a business can't equip an OTP with a printer, it should find a method to deliver the invoice to a customer's e-mail.

The platform offers many commodities to businesses, and the goal is to help everything run smoother and easier. Issuing a physical copy of an invoice is no longer mandatory and it can be done by simply sending it digitally, which results in happy customers.

How to Print a QR Code Using a Dot-Matrix Printer?

Digitalizing the tax collection system comes with a lot of benefits for both consumers and businesses. One of the best solutions TaxCore came up with is allowing a customer to easily and quickly verify their receipt.

The best way for a customer to see their receipt is to provide them with a URL that would show them a digital version of everything they purchased, may it be goods or services. TaxCore considered a couple of options that could help customers verify their purchase; however, none proved to be as simple and efficient as a QR code.

There Shouldn't Be a Mismatch

POS (Point of Sale) must assure that every fiscal invoice is in line with accreditation requirements before separating the invoice body and the QR code that would be printed on the dot-matrix copy. However, it is extremely important that the QR code applied to the dot-matrix copy is the correct one. Any possible mismatch between the QR code and the content will be subject to penalty as this means the law has been broken.

If there are any minor discrepancies, the consumer has the right to report the invoice as this is an option TaxCore provides them with so to protect their rights.

Why a QR Code and Not a URL?

In one way or another, a URL must be shown to a consumer - however, it is the form in which they see it that matters. Using a dot-matrix printer to print an entire URL on paper would be a huge waste of material and ink. Mostly because printing an entire URL would take up a whole A4 size page. Providing a consumer with this would be unacceptable and unsustainable. Moreover, no one would know how to verify a URL in its usual, long format.

Of course, there is the option to print a short URL version. And although this is somewhat more convenient, it is still not a good option because Tax Authority couldn't guarantee such an invoice. Moreover, this type of URL would be only a temporary one, which is not a long-term solution. This would automatically create a set of problems, and POS not being able to pass accreditation is probably the biggest one of them.

For reasons like these, QR code is the best possible option as it doesn't take up a lot of space, and it can be easily scanned. Probably the best solution to include it is to print the QR code on a separate slip printer and attach it to a dot-matrix invoice. As long as the customer preserves their right to verify a receipt, a dot-matrix printer is allowed to remain in service. Even though it may seem expensive, it is probably the best and easiest solution.

Printing Receipts Is Not the Only Option

TaxCore solution doesn't require taxpayers to necessarily print receipts. It offers one more options better suited for the digital age.

Vendors can easily send a verification hyperlink to a consumer. In case a receipt is issued electronically, it doesn't need to contain a QR code. It comes with a hyperlink "click here to verify invoice."

By clicking this message, the customer opens a hyperlink which verifies the receipt, and that receipt's internal data are automatically sent to the tax authority. This means that for a receipt to be verified and legitimate, it doesn't need to be printed – a digital copy is more than enough.

Different applications of electronic receipts

In retail

A digital invoice can be sent to someone who is buying through a webshop. The webshop can show the digital receipt to a customer. This receipt doesn't need to contain a QR code, only the "click here to verify invoice" message. Clicking this message opens a hyperlink that serves as a means to verify the receipt.

This option can prove to be very useful when it comes to loyal customers as well. For example, your store offers a loyalty program that customers can benefit from using a card you issued. Assuming you have this customer's e-mail, by simply scanning their loyalty card, the receipt will be automatically forwarded to them. It will contain all the products the customer bought, and the option to verify the receipt as well.

Using different messaging apps

In case you are using mobile POS to issue receipts, you can easily implement the share option. This option lets you share receipts via chat applications such as WhatsApp, Viber or WeChat. It is another option that you can use with loyal customers that helps you save a lot of time and speed up the process.

Showing the QR code to customers on the display

Another way to stop wasting both money and paper on printing receipts is to show the QR code on the customer-facing display. The consumer can scan this QR code using any QR code scanner and automatically have the digital version of the receipt on their phone.

Business to Business

Digital invoice has been used in business to business purchases for a while now. However, TaxCore makes the entire process easier and faster; and this is something businesses should take advantage of. Digitial invoices of all kind can be easily forwarded back and forth digitally, which is a great step-up from the way things were done before. Sending invoices digitally consumes much less time and allows for more efficiency.

TaxCore as a system allows a lot of functions to taxpayers, and it is up to them to decide which one of these suit their needs the most. Additionally, POS vendors are always welcome to initialize changes and come up with many new functions that make doing business easier than ever.

Fiscal Invoice - Printing and sharing option image

Test Cases

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

1. Every Normal Sale invoice (NS) is assigned with a unique SDC Invoice No consisted of RequestedBy UID – SignedBy UID – OrdinalNumber
2. Refund is made as a result of the previous Sale, so you'll use the SDC Invoice No of the Sale invoice in the Ref No field
3. Copy (CS or CR) is made based on Normal receipt, so you'll use NS or NR SDC invoice No in the Ref No 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

1. Issue Invoice
   A receipt must contain visible markings Receipt Type “NORMAL”, “COPY”, “TRAINING” or “PROFORMA”.

2. Issue Normal Sale or Refund Invoice With Buyer Identification
   Invoices with buyer identification can be issued in all invoice types and transaction types. At a beginning of the invoice creation cashier must ask the buyer for the Buyer's TIN, and optionally a Buyer Cost Center.

3. Special Cases
   This section covers special cases when it comes to issuing fiscal invoices. Its goal is to help you provide clear guidelines for your customers on how to handle these situations.

Issue Invoice

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

Steps

1.

Cashier on Accredited POS selects an invoice type and then registers the transaction by:

- typing items,
- selecting items from the previously made list,
- scanning bar code with a bar code reader.

2. The Cashier chooses the payment method and finishes the invoice.
3. Next, an accredited POS sends a message to V-SDC/E-SDC. After a successful invoice data verification, the invoice is signed, counters and totals are updated and internal data is completed.
4. The V-SDC/E-SDS sends back an Invoice response to Accredited POS.
5. The receipt is delivered to the customer.

Expected Result

A fiscal receipt is the final result of this procedure. The receipt can be printed or shared via an email or other chat application message if a customer requires it.

Every receipt is digitally signed. Internal data is stored in the TaxCore database.

A valid QR code is at the end of the receipt (or the verification URL is displayed in a clickable hyperlink form). The 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).

Issue Normal Sale or Refund Invoice With Buyer Identification

Invoices with buyer identification can be issued in all invoice types and transaction types. At a beginning of the invoice creation cashier must ask the buyer for the Buyer's TIN, and optionally a Buyer Cost Center.

Special Cases

This section covers special cases when it comes to issuing fiscal invoices. Its goal is to help you provide clear guidelines for your customers on how to handle these situations.

NOTE:
This article has an advisory status and simply offers helpful examples. DO NOT COPY-PASTE THESE GUIDELINES TO YOUR USER MANUAL without checking the legal requirements regarding fiscal law that apply in your country or jurisdiction. It is the responsibility of each POS developer to create a POS solution that will enable their

customers to comply with the requirements.

General rules

All payments must conform to the following rules:

- If the payment is not considered a taxpayer income, 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.

A fiscal receipt is required when tax liability occurs which is in most cases when:

- payment is taken by a taxpayer
- goods/services are delivered to a customer

This is constituted by the "time of supply" rule and affects different business activities such as hospitality, construction, medical services, legal cases and others, whereby advance payment can be canceled or reduced due to the contract amendments. This provokes a refund or a credit note.

NOTE:
All the variations on the subject could be differently regulated differently, as jurisdiction-specific. Note that TaxCore is designed for monitoring, so the Tax Authority doesn't expect settlement of the amount registered by an EFD (POS+SDC) at the moment fiscal receipt is issued.

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 (it is not the taxpayer's income), then no fiscal receipt is issued for the Deposit (for example, cash necessary to start a business day in retail).
2. If the Deposit is considered an advance payment, then the fiscal receipt is issued as a NORMAL-SALE transaction. It is advisable the item name is descriptive enough to clearly state that this payment is partial ("Project X - advance payment 30%")

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

1. Create another transaction to complete the deliverable - new NORMAL-SALE. It is advisable the item name is descriptive enough to clearly state that this payment is partial and completes the final price ("Project X - final payment 70%"), or
2.

Create two new transactions: 1) NORMAL-REFUND to cancel the previously recorded amount, followed by 2) NORMAL-SALE with the full price (“Project X”).

Quotes / Pre-sale

A sales quote or pre-sale gives an estimated price of a product or service and allows a prospective buyer to see the costs that will be involved for desired work.

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

Installments/Layby

This is treated as a series of separate, successive supplies of services corresponding to the successive parts of the period of the lease or agreement, or as determined by the law. Each successive supply is treated as occurring on the earlier of the date on which the payment for that successive supply is due or received.

This means that each installment is treated as a separate payment and should be covered with a NORMAL-SALE invoice. It is advised to name each installment with a descriptive name, similar to “10% of the (Item Name)” or "(Service Name) - first installment".

When the last installment payment is made, POS must provide one or both of the following options:

1. Create another transaction for the last installment - “15% of the (Item Name)” or "(Service Name) - fifth/last installment", or
2. For each previously paid installment, create a NORMAL-REFUND invoice, followed with one NORMAL-SALE invoice (containing the item or service name and the full price).

Versioning

Applicability

These policies apply to all documents, products, components or web services provided by DTI as part of TaxCore product offering.

Versioning

This topic explains versioning policy, which is based on the semantic versioning standard.

To learn more, read Semantic Versioning 2.0.0 (http://semver.org/)

What is semantic versioning?

In summary, the semantic versioning standard specifies that each version should be uniquely labeled by an identifier made up of three components:

-

major version number
- **minor** version number
- **patch** version number

These components are separated by periods. For example: The version 1.2.3 has a **major** version of 1, a **minor** version of 2, and a **patch** version of 3.

For **pre-release versions** , we may suffix the version with an identifier indicating the version's pre-release status, like **-beta1** or **-alpha2**.

When we release a new version, we increment one of the major, minor, or patch components. Differentiating between new versions is based on the kinds of changes introduced in the new version.

As is conventional in the semantic versioning standard:

- The **major** version component increments when the version contains **breaking changes**.
- The **minor** version component increments when the version contains **new functionality that is backwards compatible**.
- The **patch** version component increments when the version contains **backwards compatible bug fixes**.

Understanding Versioning Policy

The semantic versioning standard is a set of guidelines, not rigid rules. Different products and companies interpret the standard in ways that make sense to them.

We version our APIs based on the following criteria:

What qualifies as "documented" behavior?

"Documented" behavior is behavior that is referenced or explained in technical and user documentation we provide about TaxCore, or behavior that exists in the public API.

Modifying unintentional behavior (patch version increment)

We release a patch version to modify a behavior if correcting that behavior does not change any documented types, properties, methods, or parameters.

We release a **patch** version to modify a behavior when:

- the behavior is unintended and does not work as documented (a "bug"), or
- the behavior works as intended at the time of release but is later found to cause problems

Adding new functionality (minor version increment)

"New functionality" is not a term that applies to all new behavior.

It means providing you the ability to do something with the SDK that you could not do before and that involves a new type, property, method, optional parameter, or supported parameter value.

New functionality qualifies as a **minor** version release.

Introducing breaking changes (major version increment)

A "breaking change" occurs when a type, property, method, parameter, or allowable parameter value is no longer defined or no longer produces the results or behavior you want when you use it according to the documentation we provide.

In cases like this, internal methods or properties can exist that DTI cannot prevent application code from accessing, including methods and properties which are excluded from all documentation and from explicit interface declarations. We consider methods and properties like that, such as TypeScript declarations, to be internal. We do not consider breaking these internal references to be breaking changes if the underlying behavior persists. We will not release a major version solely to resolve an internal breakage.

Examples of breaking changes that would qualify for a **major** version release include:

- the application code no longer compiles (in a compiled language)
- the application or SDK is unable to do the thing you want it to do, even when you use it correctly

Alpha and canary versions (alpha and prerelease suffixes)

We provide alpha versions of React as a way to test new features early, but we need the flexibility to make changes based on what we learn in the alpha period. If you use these versions, note that APIs may change before the stable release.

Commitment to Stability

Hundreds of developers are building, testing, accrediting, and maintaining E-SDC and POS solutions that depends on different TaxCore APIs. In addition to that, dozens of applications are depending on TaxCore.API.

Breaking changes are inconvenient for everyone, so we try to minimize the number of **major** releases. Instead, we release new features in minor versions. That means that minor releases are often more interesting and compelling than majors, despite their unassuming name.

As we change TaxCore over time, we try to minimize the effort required to take advantage of new features. When possible, we'll keep an older API working. That means we need to make it as easy as possible to upgrade to new versions; if we make large changes without a migration path, people will be stuck on old versions.

Supported versions

TaxCore APIs and SDKs follow End of Life (EOL) policy to determine each version's maintenance window. The end-of-life policy also defines when DTI can drop support for older versions of their underlying platforms.

End Of Life Policy

End OF Life Policy

General Policy

API versions will be deprecated **12** months after the release of a new version.

- At the time of the new release, a notification will be sent to all interested parties that a new version is available including the change log of what was added or fixed.
- After 12 months the deprecated API version may be blocked from connecting to the service.

TaxCore web application supports the latest stable version of the following browsers:

- Chrome (Windows, Mac, Linux, and Android)
- Safari (Mac and iOS)
- Edge (Windows)

DTI may request that a customer or end users change browsers or upgrade their browser version if the customer encounters an issue in an unsupported browser version.

Priority Upgrade Policy

A **priority upgrade** will be required if there is an issue discovered that impacts ability to maintain quality of delivery of our service. If a priority upgrade is required for the API or an SDK, the old versions will be supported up to 6 months from the time of release of the new version, if a solution can be put in place to protect customer data and DTI service delivery.

- At the time of the new release, a notification will be sent to all teams using the existing API version or specific SDK that a new version is available to address a critical issue and include appropriate details of the impact of the critical issue. This notification will also outline the timeline for required action. The request will be made to upgrade to the new version as soon as possible with an offer of assistance from Support.
- **After 6 months** the deprecated API version may be blocked from connecting to the service. Deprecated SDK versions will no longer be supported and applications using the older version may no longer receive flag updates.

Critical Upgrade Policy

A **critical upgrade** will be limited to changes that are required to remediate security and/or data protection or loss issues. If a critical upgrade is required for the API or an SDK, the old versions will be immediately deprecated at the time of release of the new version.

- **Connections** from previous version **may be refused immediately**.
- At the time of the new release, a notification will be sent to all teams using the existing API version or specific SDK that a new version is available to address a critical issue and include appropriate details of the impact of the critical issue. Instructions will be provided on how to upgrade to the new version as soon as possible with an offer of assistance from support.
- Technical support and Customer Success will work with all affected teams until upgrades are completed.

Current Supported API Versions

`V1`

POS an E-SDC Upgrades and Reaccreditation

POS and E-SDCs are accredited for the version of specs that are current at the moment of accreditation

If the Sandbox environment is upgraded to the newer version during the technical review process POS and E-SDC vendors will be required to upgrade their solutions to the latest specification.

Licenses

This section describes the different licenses that are used within the TaxCore system.

1. Microsoft Public License
   The Microsoft Public License is a free an open source software license released by Microsoft, having been for its projects that were released as open source.

2. ANTLR 2 License
   See https://www.antlr2.org/license.html

3. NUnit License
   See https://nunit.org/nuget/license.html

4. Apache 2.0 License
   The Apache software license gives users permission to reuse code for nearly any purpose, including using the code as part of a proprietary software. As with other open source licenses, the Apache license governs how end-users can utilize the software in their own projects. This license is a widely-used open source license, and like other permissive licenses, it continues to grow in popularity because it encourages the use of open source software within proprietary projects.

5. Mozilla 2.0 License
   See https://www.mozilla.org/en-US/MPL/2.0/

6. BSD License
   BSD Licenses represent a family of open source licenses. The original BSD License was created as a part of the Berkeley Software Distribution (BSD) Operating System with its first release done in 1990.

7. MIT License
   The MIT License represents an agreement commonly used by software developers whenever they release Free and Open Source software. This agreement is more precisely named the **Expat License**, as it helps disambiguate it from other similar licenses. It has been reported that the use of the MIT License amounts for 27% of the total usage, making it the most used open source license type with more than 4 million public repository search results on Github.

Microsoft Public License

What is the Microsoft Public License?

The Microsoft Public License is a free an open source software license released by Microsoft, having been for its projects that were released as open source.

Within the TaxCore system, we use the Microsoft Public License along with Microsoft .NET Library and Microsoft ASP.NET MVC 3 Tools Update with the necessary modifications in place, always adhering to the prerequisites and standard procedures related to the use of these licenses.

Prerequisites

A developer is free to reproduce and distribute original or derivative works of any software licensed under the Ms-PL license. However, it is not allowed to use any contributors' name, logo, or trademarks when doing so.

The Ms-PL protects the authors by explicitly not offering any express warranties or guarantees for using their code. So the author is not liable if the code does not work well in some usage.

When a developer distributes software (or its portion) under the Ms-PL, they are not required to distribute its source code. They may do so if they want to, but they are not obliged. However, a developer is required to:

Retain all copyright, patent, trademark, and attribution notices that are originally present in the software.

Additionally, if they distribute any portion of the software in its source code form, they may do so only under the Ms-PL by including a complete copy of this license with their distribution.

If a developer distributes any portion of the software in its compiled or object code form, they may only do so under any other license that complies with the Ms-PL.

NOTE:
Here you can check the information related to the terms and conditions related to the use of this license with more detail: Open Source and Microsoft Open License.

ANTLR 2 License

See https://www.antlr2.org/license.html

NUnit License

See https://nunit.org/nuget/license.html

Apache 2.0 License

What is the Apache 2.0 License?

The Apache software license gives users permission to reuse code for nearly any purpose, including using the code as part of a proprietary software. As with other open source licenses, the Apache license governs how end-users can utilize the software in their own projects. This license is a widely-used open source license, and like other permissive licenses, it continues to grow in popularity because it encourages the use of open source software within proprietary projects.

Within the TaxCore system, we use the Apache 2.0 License with the necessary modifications in place, always adhering to the prerequisites and standard procedures related to the use of these licenses.

NOTE:
You can browse and check for more information on this topic on the Open Source Initiative's website.

Prerequisites

In order to apply the Apache 2.0 License to any project, a developer has to include a copy of the Apache License, typically in a file called "LICENSE", together with a "NOTICE" file that references the License.

You can browse through the terms and conditions applicable to the Apache licenses by accessing the Apache License's website.

What is the Apache 2.0 License text?

To apply the Apache License to specific files in the developer's work, the following boilerplate declaration has to be attached, replacing the fields enclosed by brackets "[]" with your own identifying information, without including the brackets.

The text needs to be enclosed in the appropriate comment syntax for the file format. It is also recommended to include a file or class name and description of purpose on the same "printed page" as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Mozilla 2.0 License

See https://www.mozilla.org/en-US/MPL/2.0/

BSD License

What is the BSD License?

BSD Licenses represent a family of open source licenses. The original BSD License was created as a part of the Berkeley Software Distribution (BSD) Operating System with its first release done in 1990.

The BSD License has been modified and revised several times since its initial creation and there are currently **4 different versions of the license in use**. In addition to being a specific license, the BSD is also used to describe a class of licenses, generally referred to as BSD-like, which are based on one of the BSD License iterations.

Within the TaxCore system, we are currently using the B2D 2 (FreeBSD) as well as the BSD3 licenses, making the necessary modifications to adapt them to our products.

NOTE:
Since there are several different versions of the BSD License, to avoid confusion, it is recommended to specify exactly which B2D license is in use.

Prerequisites for the B2D 2 (FreeBSD) License

The basic difference between B2D 2 (FreeBSD) and BSD 3-clause License is that the B2D 2 (FreeBSD) version omits the non-endorsement clause. It also adds a further disclaimer about views and opinions expressed in the software.

What is the B2D 2 (FreeBSD) License text?

This is the information that any developer has to place in their code with or without modification. In the TaxCore system, it is used as described in the user instructions section above:

Copyright (c) ,

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Prerequisites for the BSD 3 License

The 3 clause BSD License was released by William Hoskins, Director of the Office of Technology Licensing for UC Berkeley in 1999. The 3 clause BSD License (BSD-3) removed the advertising clause. Also referred to as the "Modified BSD License" or "New BSD" is compatible with the GNU GPL and has been vetted by the Open Source Initiative.

NOTE:
The BSD family of licenses are highly compatible the BSD 3 and 2 licenses and these are considered compatible with the Apache 2.0, as well as the MIT License.

What is the B2D 3 License text?

This is the information that any developer has to place in its code with or without modification. In the TaxCore system, it is used as described in the user instructions section above:

Copyright

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NOTE:
You can check here for more information concerning the use of the BSD Licenses: BSD 2, BSD 3.

MIT License

What is the MIT License?

The MIT License represents an agreement commonly used by software developers whenever they release Free and Open Source software. This agreement is more precisely named the **Expat License**, as it helps disambiguate it from other similar licenses. It has been reported that the use of the MIT License amounts for 27% of the total usage, making it the most used open source license type with more than 4 million public repository search results on Github.

What does it mean?

It means that when an author makes software available under the MIT License, they are granting permission to developers to make use of, share or modify the software at no extra cost. However, this is subject to two conditions:

- Developers agree not to blame the author if the software does not work as intended.
- The author will still own the copyright to their work, so the developer cannot remove the license. If the developer modifies the software in any way, they need to be clear about the changes that were made, and what changes these licenses are distributing.

It can happen that only part of a software package is MIT Licensed. For instance, part of the software may include graphics under a different license, or may come with some non-free additions. It is the task of the author to make the scope of the license as clear as possible. The MIT License code can be used commercially and in cloud-source software, although the copyright notice and the permission notice need to be included in the software's documentation.

Within the TaxCore system, we use the MIT License with the necessary modifications in place, always adhering to the prerequisites and standard procedures related to the use of these licenses.

You can browse and check for more information about this topic on the Open Source Initiative's website.

NOTE:
It is important to highlight that the MIT License code cannot be redistributed under another license, as the code has to remain under the MIT License, but the developer may be able to redistribute it as a part of another project under any desired license. In this case, any changes made to the code are released under the new license.

Prerequisites

There are a couple of instructions to take into account while using the MIT License in the software projects, here is a summary of the most important aspects to consider:

- When the developer releases the source code, they have to put a copy of the MIT License text at the top of each source file as a comment.
- When the developer releases a software package, they have to include a copy of the MIT License in the

root directory of the package. The file needs to be named either "COPYING" or "LICENSE".
- The developer has to display the MIT License as part of its software's End User License Agreement (EULA).
- The developer has to display the MIT License in any documentation.

When the developer displays the MIT License, it should also display any copyright information and, if appropriate, make clear what the license applies to.

NOTE:
Here you can see examples of other projects that are currently using MIT License: Babel, Dotnet Runtime, Ruby on Rails, Visual Studio Code, Vue.js or React.

What is the MIT License text?

This is the information that any developer has to place in its code with or without modification, in this case, in our TaxCore system, as described in the user instructions section above:

Copyright

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.