## Change Log

<table>
<thead>
<tr>
<th>Author</th>
<th>Change</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTI</td>
<td>Review of the initial version of the document</td>
<td>1.0</td>
</tr>
</tbody>
</table>
| DTI    | 1. User Manual for Taxpayer Administration Portal is added to the document  
2. Revision based on the feedback from testers  
3. Added Status and Error Codes  
4. Common JSON Based Protocol for both HTTP and Serial connection  
5. Added Hash field into Sign Invoice request and response  
6. Added commands Attention and Get Signed Invoice  
7. POS to E-SDC Communication over Serial Port is JSON based  
8. TaxOnTotalAmount field removed from TaxCategory, enumeration Type added instead, with examples. | 2.0     |
| DTI | 1. Rename Cashier and Total Amount fields in textual representation of the invoice  
2. Cashier TIN optional changed to mandatory  
3. Authority to Service  
4. SDC returns four additional fields in `InvoiceFiscalizationResult`  
5. Added and removed Error Codes  
6. Province to District  
7. Swagger to OpenAPI-Specification  
8. Added Optional | Mandatory Table to Anatomy of an Invoice  
9. Create Verification Url | Total Amount from 4 to 8 bytes int32 to Uint64 bit  
10. Create Verification Url | Changed default version from 1 to 0x02  
11. Expected frequency of submission of new ARPs to Tax Service’s system is described in Audit section  
12. Added Error code: 2400 Device is not configured  
13. Modified description for Error code 0220 and 0210  
14. Statuses of call to SubmitAuditPackage are now documented  
15. `/api/SDC/StartProofOfAudit` is documented  
16. Added section Error Response for Serial protocol  
17. Replaced `InvoiceFiscalizationRequest` with Request and `InvoiceFiscalizationResponse` with Result as per the Model  
18. Added description for APDU protocols and IsoCases  
19. Added field `UnitPrice` to `InvoiceFiscalizationRequest`  
20. Added decimal format for `InvoiceFiscalizationRequest`  
21. Added description for MRC in `InvoiceFiscalizationResponse`  
22. Added Payment Method in Receipt example  
23. Added definition for Get Secure Element Version APDU command | 2.1 |
| DTI | 1. Changed Item Name length from 2gb to 2kb  
2. Deleted E-SDC shall Sign invoice... from High-level requirements  
3. Removed 384 px x 384 px dimensions for QR code  
4. Renamed Commands.json > *UID.commands + example  
5. Date and time - added definition for daylightsaving  
6. Tax amounts - added new section  
7. Fiscal invoices - improved definition and description  
8. Ordinal_Number - added definition  
9. Anatomy of fiscal receipt - improved definition and description  
10. Added new examples for NR, Training and Proforma  
11. Format of audit package - improved definition  
12. JSON based protocol - improved definition for Sign Invoice  
13. Added mapping Fiscal invoice to Fiscal Receipt | ver.2.3 |
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Log</td>
<td>2</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>5</td>
</tr>
<tr>
<td>Introduction</td>
<td>9</td>
</tr>
<tr>
<td>Interpretations</td>
<td>10</td>
</tr>
<tr>
<td>High-Level Requirements</td>
<td>12</td>
</tr>
<tr>
<td>Connectivity</td>
<td>13</td>
</tr>
<tr>
<td>Modes of Operation</td>
<td>13</td>
</tr>
<tr>
<td>Development Environment</td>
<td>14</td>
</tr>
<tr>
<td>Obtaining Test Certificates</td>
<td>14</td>
</tr>
<tr>
<td>Obtaining Smart Cards</td>
<td>14</td>
</tr>
<tr>
<td>Identification of the Environment</td>
<td>14</td>
</tr>
<tr>
<td>Using Taxpayer Admin Portal</td>
<td>14</td>
</tr>
<tr>
<td>Architecture</td>
<td>15</td>
</tr>
<tr>
<td>E-SDC Implementations</td>
<td>15</td>
</tr>
<tr>
<td>Standards</td>
<td>15</td>
</tr>
<tr>
<td>Power supply</td>
<td>15</td>
</tr>
<tr>
<td>Ports</td>
<td>16</td>
</tr>
<tr>
<td>TaxCore Backend.Api</td>
<td>17</td>
</tr>
<tr>
<td>Smart Cards</td>
<td>17</td>
</tr>
<tr>
<td>Secure element Applet</td>
<td>17</td>
</tr>
<tr>
<td>PKI Applet</td>
<td>18</td>
</tr>
<tr>
<td>E-SDC States</td>
<td>18</td>
</tr>
<tr>
<td>Authentication</td>
<td>18</td>
</tr>
<tr>
<td>Digital Certificates and PIN Codes</td>
<td>18</td>
</tr>
<tr>
<td>Digital Certificates for Testing Purpose</td>
<td>18</td>
</tr>
<tr>
<td>Authentication Token</td>
<td>19</td>
</tr>
<tr>
<td>Manuals</td>
<td>19</td>
</tr>
<tr>
<td>Data Structures</td>
<td>20</td>
</tr>
<tr>
<td>UID</td>
<td>20</td>
</tr>
<tr>
<td>Amounts</td>
<td>20</td>
</tr>
<tr>
<td>Date and Time</td>
<td>20</td>
</tr>
<tr>
<td>Tax Rates</td>
<td>20</td>
</tr>
<tr>
<td>Model</td>
<td>21</td>
</tr>
<tr>
<td>Example</td>
<td>22</td>
</tr>
<tr>
<td>Topic</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Tax Amounts</td>
<td>23</td>
</tr>
<tr>
<td>Fiscal Invoices</td>
<td>23</td>
</tr>
<tr>
<td>Unique Identification of a Fiscal Invoice</td>
<td>24</td>
</tr>
<tr>
<td>Key Elements</td>
<td>24</td>
</tr>
<tr>
<td>Json Representation of the Invoice</td>
<td>24</td>
</tr>
<tr>
<td>Anatomy of Fiscal Receipt</td>
<td>25</td>
</tr>
<tr>
<td>Normal Refund Receipt</td>
<td>26</td>
</tr>
<tr>
<td>Training or Proforma or Copy Receipt</td>
<td>27</td>
</tr>
<tr>
<td>Audits</td>
<td>27</td>
</tr>
<tr>
<td>Encryption of Audit Data</td>
<td>28</td>
</tr>
<tr>
<td>Format of the Audit Package</td>
<td>28</td>
</tr>
<tr>
<td>Commands</td>
<td>28</td>
</tr>
<tr>
<td>Configure Time Server URL Command</td>
<td>29</td>
</tr>
<tr>
<td>Set Tax Rates Command</td>
<td>29</td>
</tr>
<tr>
<td>Update Verification URL Command</td>
<td>30</td>
</tr>
<tr>
<td>Proof of Audit Command</td>
<td>30</td>
</tr>
<tr>
<td>Commands Results</td>
<td>30</td>
</tr>
<tr>
<td>Model</td>
<td>30</td>
</tr>
<tr>
<td>Example</td>
<td>30</td>
</tr>
<tr>
<td>Processes</td>
<td>31</td>
</tr>
<tr>
<td>Initialization</td>
<td>31</td>
</tr>
<tr>
<td>Standard Operation</td>
<td>31</td>
</tr>
<tr>
<td>Enter PIN to Unlock Secure element</td>
<td>31</td>
</tr>
<tr>
<td>Fiscalization of an Invoice</td>
<td>31</td>
</tr>
<tr>
<td>Calculate Taxes</td>
<td>32</td>
</tr>
<tr>
<td>Create Verification URL</td>
<td>34</td>
</tr>
<tr>
<td>Create QR Code</td>
<td>35</td>
</tr>
<tr>
<td>Create Textual Representation of an Invoice (Receipt)</td>
<td>35</td>
</tr>
<tr>
<td>Creating an Audit Package</td>
<td>37</td>
</tr>
<tr>
<td>Audit Process</td>
<td>37</td>
</tr>
<tr>
<td>Notifications</td>
<td>41</td>
</tr>
<tr>
<td>Sync Date and Time</td>
<td>41</td>
</tr>
<tr>
<td>Malfunctions and Non-serviceable devices</td>
<td>42</td>
</tr>
<tr>
<td>Dump Audit Packages Kept on E-SDC when Secure element is damaged</td>
<td>42</td>
</tr>
<tr>
<td>Protocols</td>
<td>43</td>
</tr>
<tr>
<td>Status and Error Codes</td>
<td>43</td>
</tr>
</tbody>
</table>
Submit Audit Package............................................................................................................. 68
Submit Audit Request Payload (ARP) .................................................................................... 69
File-Based Communication.................................................................................................... 70
  SD Cards or Flash memory drives format................................................................................. 70
  Tax Inspector Configures a new E-SDC using an SD Card..................................................... 70
  E-SDC Executes Commands Received via SD Card/USB drive ............................................ 70
  E-SDC Stores a command execution result to the SD Card/USB drive ................................ 70
  E-SDC Stores Audit Files on SD Card/USB drive................................................................... 71
Introduction

E-SDCs are software applications or hardware devices whose main function is to safeguard an invoice in offline mode and deliver audit packages to Tax Service. From the technical point of view, E-SDC is a middleware component that connects POS system to Secure element and enables standardized communication with TaxCore Backend.Api.

This document is based on several researches performed in different environments because technical specification has to be open and applicable to organizational and technological constraints and anticipated changes and to keep security tight in order to perform fiscalization of an invoice in safe and secure fashion.

The result is a rather complex set of rules and protocols E-SDC has to comply with in order to become an integral part of the fiscalization system. Good news is that most of rules are straightforward and simple to implement using any platform or programming language.

This Guideline sets standards that will enable simple integration of accredited E-SDCs with the Tax Service’s system. E-SDCs shall comply with the protocols described in the document.
Interpretations

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

**AES256** is a specification for the encryption of the electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.

**APDU command** (application protocol data unit) is the communication unit between a smart card reader and a smart card. The structure of the APDU is defined by ISO/IEC 7816-4 Organization, security and commands for interchange.

**Audit Data** – textual representation and machine-readable representation of a fiscal invoice with associated metadata.

**Audit Package** – encrypted Audit Data ready to be sent to Tax Service System.

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

**GUID** is Globally Unique Identification Number. In its canonical textual representation, the sixteen octets of a UUID are represented as 32 hexadecimal (base 16) digits, displayed in five groups separated by hyphens, in the form 8-4-4-4-12 for a total of 36 characters (32 alphanumeric characters and four hyphens). For example: 123e4567-e89b-12d3-a456-42665440000

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

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

**Invoice**, see Receipt.

**Non-volatile memory** is a type of computer memory that can retrieve stored information even after having been power cycled (turned off and back on) e.g. USB Flash drive.

**Proof of Audit (PoA)** is a confirmation generated by a Tax Service System once all expected audit packages have been received from an E-SDC.

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

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

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

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

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

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

TaxCore is a set of web services, sites, databases and management software installed on the tax Service side, designed for communication with POS and SE devices.

UID – Unique Identifier (8 alphanumeric characters) assigned to each Smart card and embedded in the subject field of a digital certificate.

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

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

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

Volatile memory is computer memory that requires power to maintain the stored information; it retains its contents while powered on but when the power is interrupted, the stored data is lost immediately or very rapidly e.g. RAM.
High-Level Requirements

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

1. E-SDC shall be able to generate an invoice without the Internet connection.
2. E-SDC shall rely on a Secure Element (delivered as Smartcard) to safeguard an invoice.
3. E-SDC shall calculate tax liabilities based on received Invoice items and defined tax rates.
4. Initial E-SDC configuration could be performed via a file (e.g. on USB disk) or the Internet connection.
5. E-SDC shall expose at least one standardized protocol to a POS (Json via HTTP or serial).

6. E-SDC shall process all commands received from the Tax Service’s system in a consecutive order. These commands can include time synchronization, update of tax rates, and so forth.
7. E-SDC shall encrypt an audit data and store it locally in an encrypted form.
8. E-SDC shall periodically send stored audit data to a Tax Service and this process is called an audit.
9. E-SDC shall keep an audit data locally until a proof of audit (PoA) has been received from the Tax Service’s system stating the audit data has been securely stored on the Tax Service’s system.
10. E-SDC does not have to keep an audit data submitted and successfully stored on the Tax Service’s system.
11. E-SDC shall submit 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 shall not store the Secure Element’s PIN code except in the working memory. Once the E-SDC is restarted, the cashier will be required to enter the PIN code again.
13. E-SDC shall store in its memory up to 50,000 Audit packages.
Figure 1 shows the high-level architecture of Fiscalization System, involving an Accredited POS, E-SDC and Tax Service’s system (TaxCore).

![Diagram](image)

**Connectivity**

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

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

**Modes of Operation**

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 a poor network coverage or the Internet unavailability.

E-SDC can work in the following modes:

**Offline (Required)**

Offline mode is the only mode of operation required by these Instructions. In the offline mode, the Secure Element signs an invoice and the E-SDC device stores an audit package locally in a secure manner.

**Semi-offline (Optional)**

In the semi-offline mode, the Secure Element signs an invoice and the E-SDC device will immediately try to contact the Tax Service’s system and perform a remote audit. If the Service’s system is not accessible, the E-SDC will switch to the offline mode.

In case the Internet connection is interrupted, the E-SDC switches to the Offline mode.
Development Environment

Development environment is accessible to all developers of the Accredited E-SDCs. Development Environment exposes the same APIs and uses the same protocols as a production environment.

Obtaining Test Certificates

Every developer or a company registered as a developer of an Accredited E-SDC on Tax Service’s web site shall receive a set of test certificates and technical documentation. Test certificates shall make it possible to test both successful and failing scenarios, like trying to fiscalize an invoice with the expired certificate.

Obtaining Smart Cards

Accredited E-SDC vendors will be able to apply to a Tax Service and get test smart cards to use for development, integration and testing purposes. The process for obtaining smart cards is the same as the process for the actual taxpayers:

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

Identification of the Environment

Tax Service shall publish the URL of the Staging and Production environment on their web site.

Using Taxpayer Admin Portal

Once a test smart card is delivered to a Primary Contact you can use it to login to the Taxpayer Admin portal to search and view an invoice details submitted to the backend of the staging environment.
Architecture

Figure 2 shows components of a fiscalization system and their mutual relationship.

![Fiscalization system component diagram](image)

### E-SDC Implementations

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.

### Standards

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


A device 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 case of the AC voltage, a device shall work with the frequency range 50-60 Hz. Power supply circuit used by the E-SDC shall be protected with an automatic circuit breaker, suitable for electronic devices (type I).

Beside the mandatory basic protection, in order to protect sensitive electronic components, a smart card and data stored on non-volatile memory, 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.

Ports

POS Communication

POS shall 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. Minimum speed of Ethernet port is at least 10 Mb/s.

Wireless

Wireless connection in compliance with IEEE 802.11 to a POS device and a local network.

Serial

For the serial communication with a POS, an E-SDC shall have one connector with minimum 3 pins: Tx, Rx and GND. Preferred type of connector is DE-9 connector compliant with standards IEC 60807-3 / DIN 41652. Alternatively, Euro Style Connectors, pin header connectors, or similar can be used. In that case PIN usage shall be clearly specified.

<table>
<thead>
<tr>
<th>Pin Number</th>
<th>Pin Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Unused</td>
</tr>
<tr>
<td>2</td>
<td>RXD</td>
</tr>
<tr>
<td>3</td>
<td>TXD</td>
</tr>
<tr>
<td>4</td>
<td>Unused</td>
</tr>
<tr>
<td>5</td>
<td>Logic GND</td>
</tr>
<tr>
<td>6</td>
<td>Unused</td>
</tr>
<tr>
<td>7</td>
<td>Unused</td>
</tr>
<tr>
<td>8</td>
<td>Unused</td>
</tr>
<tr>
<td>9</td>
<td>Unused</td>
</tr>
</tbody>
</table>

The Accredited POS and the E-SDC shall be able to communicate via a «Null modem» cable.

Audit

If the E-SDC device uses a USB flash drive for a Local Audit, USB connectors "USB Type B female" shall be used. If a serial over USB protocol is used for communication between a POS and an E-SDC, additional USB port of the same type shall be available. Applied USB communication protocol shall be
“USB 2.0” or higher. Same USB port shall not be used for a Local audit and serial communication with a POS.

In case when an SD Flash memory card is used for a Local audit, a device shall have an easily accessible Micro SD card connector.

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

**Smart Card**

E-SDC shall have a smart card reader in compliance with ISO/IEC 7810 and ISO/IEC 7816 standard. Supported Smart Card sizes are 1FF (credit card size) and 2FF (mini SIM card size)

<table>
<thead>
<tr>
<th>SIM card</th>
<th>Standard reference</th>
<th>Length (mm)</th>
<th>Width (mm)</th>
<th>Thickness (mm)</th>
<th>Volume (mm³)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-size (1FF)</td>
<td>ISO/IEC 7810:2003, ID-1</td>
<td>85.60</td>
<td>53.98</td>
<td>0.76</td>
<td>3511.72</td>
</tr>
<tr>
<td>Mini-SIM (2FF)</td>
<td>ISO/IEC 7810:2003, ID-000</td>
<td>25.00</td>
<td>15.00</td>
<td>0.76</td>
<td>285.00</td>
</tr>
</tbody>
</table>

**TaxCore Backend.Api**

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

E-SDC is authenticated by Backend.Api using a client digital certificate and an authentication Token.

**Smart Cards**

For standard operations, each E-SDC shall require a Smart card issued by a Tax Service, which consists of two applets:

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

Both applets share the same PIN code. PIN is chosen by a Taxpayer’s authorized person on a Tax Service’s portal during the fiscalization enrolment process or while requesting an additional POS Smart cards.

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

**Secure element Applet**

**Secure element (SE)** is a fiscal component implemented as a special software or a device designed to receive an invoice data, perform signing and data processing and generate response which is sent back to caller for further actions. Response provides authenticity of invoice data. Secure Element is issued and controlled by the Tax Service. Main purpose of the Secure element is to sign invoices using a taxpayer’s digital certificate, to control audits and to 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 Service system. Likewise, SE will continue to produce fiscal invoices once it receives proof from TaxCore Backend.Api that audit has been received and stored on the Tax Service’s system.

PKI Applet

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

E-SDC States

The diagram on Figure 5 shows basic E-SDC states and transitions.

Authentication

Authentication against the Tax Service’s system is performed using the taxpayer’s digital certificate.

Digital Certificates and PIN Codes

The Tax Service’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 Service will issue the requested number of test digital certificates to each accredited supplier and each accredited taxpayer.
Authentication Token

E-SDC uses authentication token when calling the TaxCore API web services.

Authentication token is obtained from TaxCore API by calling the RequestAuthenticationToken web service and providing a Taxpayer’s digital certificate.

Manuals

E-SDC shall have a user manual that explains in detail some of the following topics.

1. Installation instructions for the technicians performing the installation and integration of an E-SDC device or software on a sales point.
2. User instructions for the operator (cashier or shopkeeper) explaining normal operations in detail.
3. Local and/or remote audit instructions
Data Structures

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 SERIALNUMBER parameter.

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

Amounts

TaxCore works with amounts presented as decimal numbers with 4 decimal places. For the sake of simplicity, all binary based protocols (POS to E-SDC Communication over Serial Port and E-SDC to Secure element (APDU commands)) use integers representing the amount multiplied by 10,000. For example, the amount 123,4567 will be converted into 1234567.

JSON based protocols use the standard decimal format.

Date and Time

E-SDC shall have access to the current time. Real Time Clock or similar component shall be installed and used to maintain the correct time while the power is off.

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

UTC is the default time used by E-SDC for all purposes, except:

- Date and time sent by a POS to an E-SDC is the local time.
- Date and time printed on a journal (textual representation of an invoice) generated by an E-SDC is local time. (Fiji Winter Time is used as a local time if E-SDC cannot track daylight saving time).
- Date and time sent by an E-SDC to a POS is the local time (Fiji Winter Time is used as a local time if E-SDC cannot track daylight saving time).

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

All binary based protocols (POS to E-SDC Communication over Serial Port and 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).

In case of a power outage, the Real Time Clock shall be able to operate uninterruptedly up to 6 months.

Tax Rates

Tax rate is a sales tax expressed as 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 an ID, which represents taxes revision and a date / time which defines the moment when the new taxes shall take effect.

E-SDC receives the tax groups via Set Tax Rates Command on the Initialization process or an information if taxes have been changed.

### Model

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TaxRateGroup</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValidFrom</td>
<td>Date time</td>
<td>Date when a tax rates group shall enter into force</td>
</tr>
<tr>
<td>GroupId</td>
<td>32 bit integer</td>
<td>Revision number for all taxes under a tax rates group</td>
</tr>
<tr>
<td>TaxCategories</td>
<td>Array of TaxCategory</td>
<td>All tax categories under one tax rates group</td>
</tr>
<tr>
<td><strong>TaxCategory</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CategoryId</td>
<td>32 bit integer</td>
<td>Unique ID of a category (system-wise)</td>
</tr>
<tr>
<td>Name</td>
<td>Unicode String</td>
<td>Name of a tax (tax category).</td>
</tr>
<tr>
<td>Type</td>
<td>32 bit integer</td>
<td>One of the following tax category types:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 (tax-on-net) - all tax rates from this category are proportional, and shall be applied on the net price;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 (tax-on-total) - all tax rates from this category are proportional, and shall be applied on the total amount;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 (amount-per-quantity) - all tax rates from this category are fixed tax amounts, which shall be multiplied with item quantity.</td>
</tr>
<tr>
<td>OrderId</td>
<td>32 bit integer</td>
<td>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</td>
</tr>
<tr>
<td>TaxRates</td>
<td>Array of TaxRate</td>
<td>All tax rates for a tax (category)</td>
</tr>
<tr>
<td>RateId</td>
<td>32 bit integer</td>
<td>Unique ID of a tax rate (system-wise)</td>
</tr>
<tr>
<td>--------</td>
<td>----------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>Label</td>
<td>Unicode String</td>
<td>Label for a tax rate, unique within a tax group, always belongs to one tax category.</td>
</tr>
<tr>
<td>Rate</td>
<td>Decimal</td>
<td>Rate percentage for a proportional tax, or tax amount for a fixed tax.</td>
</tr>
</tbody>
</table>

Table 2 Tax rates data structure

Example

```
"TaxRateGroup": {
   "ValidFrom": "2017-07-02T00:00:00",
   "GroupId": "1",
   "TaxCategories": [{
      "CategoryId": 1002,
      "Name": "VAT",
      "Type": 0,
      "TaxRates": [{
         "RateId": 1002,
         "Rate": 6.0,
         "Label": "A"
      },
      { "RateId": 1003,
        "Rate": 10.0,
        "Label": "B"
      },
      { "RateId": 1004,
        "Rate": 20.0,
        "Label": "C"
      }],
   "OrderId": 1,
  },
  { "CategoryId": 1003,
    "Name": "STT",
    "Type": 1,
    "TaxRates": [{
       "RateId": 1005,
       "Rate": 25.0,
       "Label": "D"
    }],
   "OrderId": 2,
  },
  { "CategoryId": 1004,
    "Name": "ECAL",
    "Type": 2,
    "TaxRates": [{
       "RateId": 1006,
       "Rate": 10.0,
       "Label": "E"
    }],
   "OrderId": 3,
  }
}
```
Tax Amounts

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

<table>
<thead>
<tr>
<th>Tax Type</th>
<th>Formula</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tax</td>
<td>( \text{Tax Amount} = \sum (\text{Base price} \times (\text{Rate/100})) )</td>
<td>for each item with this label</td>
</tr>
<tr>
<td>Tax on Total</td>
<td>( \text{Tax Amount} = \sum (\text{Total price} \times (\text{Rate/100})) )</td>
<td>for each item with this label (Total price here is item base price with all other regular taxes included)</td>
</tr>
<tr>
<td>Amount per Quantity</td>
<td>( \text{Tax Amount} = \sum (\text{Fix amount} \times \text{quantity}) )</td>
<td>for each item with this label, item quantity is multiplied with fix amount as defined by tax Service. If other tax labels are defined on item, those taxes are calculated on the remainder</td>
</tr>
</tbody>
</table>

Fiscal Invoices

This section defines the minimum set of attributes required to produce a fiscal invoice.

Fiscal invoice is, by definition, a digitally signed acknowledgment that a specified payment has been received or refunded. Fiscalization of an invoice is a process of applying a digital signature on the electronic content of the invoice by the Secure Element.

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

Each invoice type is associated with one of following transaction types:
- sale;
- refund.

Each invoice type is associated with one of following payment types:
- cash
- card
- check
- wire transfer
- mobile money
- voucher
- other.
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.

Key Elements

Fiscal invoice must contain the following parts. A fiscal invoice may 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 standard, publicly available protocol for communication to E-SDC, using 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

Digital signature applied on the content of an electronic invoice by the Secure Element.

Internal Data

Internal data contains encrypted fiscal data. Content of the internal data is readable by the Tax Service 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).

Json Representation of the Invoice

As previously described, each Invoice consist of two separate data structures.

InvoiceFiscalizationRequest - this object is created by POS and submitted to E-SDC

InvoiceFiscalizationResponse - 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 (InvoiceFiscalizationRequest),
  Result (InvoiceFiscalizationResult)}
Anatomy of Fiscal Receipt

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

<table>
<thead>
<tr>
<th>Textual representation of Fiscal Invoice</th>
<th>Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title line – marks the beginning of the fiscal part of receipt</td>
<td></td>
</tr>
<tr>
<td>TIN: 502579006 Company: Golf Store: Sun Store Address: 7 Someplace District: Suva</td>
<td></td>
</tr>
<tr>
<td>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 Invoice Response). Values are extracted from the subject field of digital certificate stored in the Secure Element Applet.</td>
<td></td>
</tr>
<tr>
<td>Cashier TIN: 1234567890</td>
<td></td>
</tr>
<tr>
<td>Cashier’s identification. Local regulations might mandate POS to send particular data instead of cahier’s name like Employee ID or some other information that uniquely identifies the POS cashier.</td>
<td></td>
</tr>
<tr>
<td>Buyer TIN: 5123456789 Buyer Cost Centre: 123 POS number: POS2017/998 POS time: 15/6/2017 8:56:23AM</td>
<td></td>
</tr>
<tr>
<td>Buyer TIN is mandatory only in case of B2B transaction and in that case, it must be printed on the receipt. Buyer Cost Center is optional and reserved for further use, it must be present only for B2B transactions. POS (Invoice) Number and POS (Invoice) time are optional fields.</td>
<td></td>
</tr>
<tr>
<td>Ref no: P22VC8VR-3TJC5V65-114906</td>
<td></td>
</tr>
<tr>
<td>Reference (Document) Number is required only for Refund or Copy invoice type. In that case, Ref no must be printed on the receipt (and on journal), containing SDC Invoice No of any issued Invoice or Refund, in format RequestedBy-SignedBy-OrdinalNumber. For any other invoice/transaction type (for example Normal Sale invoice referencing to Proforma Sale invoice) this field is optional.</td>
<td></td>
</tr>
<tr>
<td>Total Purchase: 3249.52 Payment Method: Cash</td>
<td></td>
</tr>
<tr>
<td>Total Purchase, Tax items and Total Tax are calculated by V-SDC or E-SDC during fiscalization of the invoice and are returned to POS as a part of the response. Payment Method: Cash, Card, Check, Wire Transfer, Voucher, Mobile Money, or Other. Taxpayer's tax liability is based on these tax amounts, calculated by V-SDC or E-SDC. Calculation is explained in the section Tax Amounts.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Price</td>
</tr>
<tr>
<td>Sport-100 Helmet, Blue (E)</td>
<td>34.99</td>
</tr>
<tr>
<td>Mountain Bike Socks, M (A)</td>
<td>9.03</td>
</tr>
<tr>
<td>HL Road Frame - Red, 58 (F, A)</td>
<td>1431.50</td>
</tr>
<tr>
<td>Plastic bag (P)</td>
<td>0.10</td>
</tr>
<tr>
<td>Total Purchase: 3249.52 Payment Method: Cash</td>
<td></td>
</tr>
<tr>
<td>Total Purchase, Tax items and Total Tax are calculated by V-SDC or E-SDC during fiscalization of the invoice and are returned to POS as a part of the response. Payment Method: Cash, Card, Check, Wire Transfer, Voucher, Mobile Money, or Other. Taxpayer's tax liability is based on these tax amounts, calculated by V-SDC or E-SDC. Calculation is explained in the section Tax Amounts.</td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Name</td>
</tr>
<tr>
<td>E</td>
<td>STT</td>
</tr>
<tr>
<td>A</td>
<td>VAT</td>
</tr>
<tr>
<td>F</td>
<td>ECAL</td>
</tr>
<tr>
<td>P</td>
<td>PB</td>
</tr>
<tr>
<td>Total Tax: 488.41</td>
<td></td>
</tr>
<tr>
<td>Fiscal metadata added to the invoice through fiscalization. SDC Invoice No - Combination of Requested By (7AF4D923), Signed By (E3B30A31) and Ordinal Invoice Number (234) is system-wide unique identification of fiscal invoice. It may be used instead of current receipt/invoice number generated by POS.</td>
<td></td>
</tr>
</tbody>
</table>
SDC Time is the official date and time relevant to the tax calculation and reporting. Invoice Counter is generated by V-SDC or E-SDC and explained in section Invoice Response, field IC.

QR Code contains Invoice verification URL. QR Code also contains Internal data and digital signature used for the invoice verification. Invoice is verifiable by customer immediately after fiscalization.

In case an invoice/receipt is delivered as an electronic document (email), QR Code shall be substituted with Invoice verification URL in (clickable) hyperlink format.

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

Title line – marks the end of the fiscal part of a receipt
This is custom message

-------- END OF FISCAL INVOICE --------

Title line – marks the end of the fiscal part of a receipt
This is custom message

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

For Refund transaction type Ref no element is mandatory.

Example:

============= FISCAL INVOICE =============
TIN: 502579006
Company: Golf V Store: Sun Store
Address: 7 Someplace District: Suva
Cashier TIN: 12345678
POS number: 89347415-2817
POS time: 2018-03-09 14:57:25
Ref no: 7AF4D923-E3B30A31-234

------------- NORMAL REFUND ------------

Items
----------------------------------------
Name                   Price       Qty.       Total
Sport-100 Helmet, Blue (E)
34.99                   10          -349.90
Mountain Bike Socks, M (A)
9.03                    4           -36.12
----------------------------------------
Total Purchase: 386.02
Payment Method: Cash

----------------------------------------
Label      Name  Rate       Tax
E          STT  6.00%      19.81
A          VAT  9.00%      2.98
----------------------------------------
Total Tax: 22.79

============= FISCAL INVOICE =============
SDC Time: 2018-03-09 14:57:46
SDC Invoice No: 7AF4D923-E3B30A31-235
Invoice Counter: 4/235NR

-------- END OF FISCAL INVOICE --------

---- QR code omitted for simplicity ----
Training or Proforma or Copy Receipt

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

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

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

For Copy invoice type Ref no element is mandatory.

Example:

===== THIS IS NOT A FISCAL RECEIPT =====
TIN:  502579006
Company:  Golf V Store
Address:  7 Someplace
District:  Suva
Cashier TIN:  
POS number:  89347415-2017
POS time:  2018-03-09 14:57:25
---------- TRAINING SALE -----------
Items
========================================
Name    Price        Qty.          Total
Sport-100 Helmet, Blue (E)  
34.99          10         349.90
Mountain Bike Socks, M (A)  
9.03           4          36.12
----------------------------------------
Total Purchase:                   386.02
Payment Method:                     Cash
========================================
THIS IS NOT A FISCAL INVOICE
========================================
Label        Name     Rate           Tax
E             STT    6.00%         19.81
A             VAT    9.00%          2.98
----------------------------------------
Total Tax:                         22.79
SDC Time:            2018-03-08 14:57:46
SDC Invoice No:    7AF4D923-E3B3A31-236
Invoice Counter:                 1/236TS
========================================
---- QR code omitted for simplicity ----
===== THIS IS NOT A FISCAL RECEIPT ====

Audits

Audit data represents machine readable formatted fiscal invoice signed by a taxpayer’s private key followed by journal data. Journal data is 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 Service’s system, after successful audit process.
Encryption of Audit Data

Encryption of audit data prevents access to sales data by unauthorized persons. The only one that can decrypt audit data is the Tax Service’s system software running on the Tax Service premises and used by the authorized personnel only.

Format of the Audit Package

The Audit Package is a textual file in JSON format

```
AuditData {
    Key (string),
    IV (string),
    Payload (string)
}
```

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>Base64 Encoded String</td>
<td>One-time symmetric key (256Bit long) encrypted using Taxcore public key</td>
</tr>
<tr>
<td>IV</td>
<td>Base64 Encoded String</td>
<td>Initialization vector Key encrypted using TaxCore public key ------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Payload</td>
<td>Base64 Encoded String</td>
<td>Base64Encoded JSON format of an invoice, as described in section Json Representation of the Invoice, encrypted with Key and IV using AES256 algorithm.</td>
</tr>
</tbody>
</table>

Commands

Commands are means of communication between the Tax Service’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 Service’s system using a remote or a Local audit.

Commands are always delivered as an array structure. Commands shall be executed in a 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)
}
```

```
enum CommandsType {
    UpdateTaxRates = 0,
    UpdateNTPServiceUrl = 1,
    UpdateVerificationURL = 2,
    UpdatePAC = 3, // reserved for later use
    TaxCorePublicKey = 4 // deprecated,
    EndProofOfAudit = 5
}
```

CommandId is a unique identifier assigned by Tax Service’s system. Once a command is successfully executed, Backend.Api shall be notified as described in the Process Commands section of this document.
Type defines 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 Backend.Api to an E-SDC in Json format. Format of each type is described in the following sections.

Configure Time Server URL Command

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

Set Tax Rates Command

Payload contains a group of all new tax rates which shall be applied from the specified date and time. The structure of a group is defined in section Tax Rates. Date can be in the past or (more likely) in the future. Tax rates with the future date shall be stored in non-volatile memory and applied starting from the specified moment. If more than one group has the same date, the one with higher GroupId shall be applied.

Payload of the command is structured as following:

```json
{
    "TaxRateGroup": {
        "ValidFrom": "2017-07-02T00:00:00",
        "GroupId": "1",
        "TaxCategories": [{
            "CategoryId": 1002,
            "Name": "VAT",
            "Type": 0,
            "TaxRates": [{
                "RateId": 1002,
                "Rate": 6.0,
                "Label": "A"
            },
            {"RateId": 1003,
            "Rate": 10.0,
            "Label": "B"},
            {"RateId": 1004,
            "Rate": 20.0,
            "Label": "C"}],
        "OrderId": 1,
    },
    {"CategoryId": 1003,
    "Name": "STT",
    "Type": 1,
    "TaxRates": [{"RateId": 1005,
    "Rate": 25.0,
    "Label": "D"}],
    "OrderId": 2,
    }
}
```
Update 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 the Create Verification URL section.

Example: https://fe.staging.vms.frcs.org.fj/v/?vl={Query String Parameter Constructed By E-SDC}

Proof of Audit Command

Proof of audit command payload is transmitted to the Secure Element applet on the smart card (End Audit APDU command) once audit process is completed successfully on the Tax Service’s system.

Payload is a byte array encoded as a base64 string.

Commands Results

Commands Results is a confirmation to Backend.Api that a certain command is executed.

Model

```json
CommandResults {
    "CommandId": (GUID),
    "Success": (boolean),
    "DateAndTime": (string)
}
```

Example

```json
{
    "CommandResults": [
        {
            "CommandId": "945bb863-5c7f-4826-9ae3-26debcac331a",
            "Success": 1,
            "DateAndTime": "2017-06-17T04:33:47+00:00"
        }
    ]
}
```
Processes

This section describes processes performed by an E-SDC.

Initialization

Prior to the first use, the E-SDC has to be initialized. E-SDC shall have access to the Secure Element during the initialization process in order to establish a secure connection with the TaxCore Backend.Api to obtain a set of initialization commands. Commands are explained in the section Commands. In case of the poor or no internet connection, configuration commands can be uploaded via file-based communication as explained in section E-SDC Stores Audit Files on SD Card.

Standard Operation

Enter PIN to Unlock Secure element

Before the Secure Element applet can be used, a valid PIN code shall be supplied from the POS using the serial or the Ethernet connection. Once the E-SDC receives a PIN code, it will try to execute Pin Verify APDU command. Depending on the provided PIN, the SE will remain either unlocked for further use, or locked until a valid PIN is entered. E-SDC will send response to the POS based on the result of the PIN Verify command execution.

It is important to note the SE interprets data as byte containing digits, so appropriate conversion shall be done by E-SDC 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.

Fiscalization of an Invoice

Invoice fiscalization is the main function of an E-SDC. The following steps shall be executed by the E-SDC once a request data is received from an Accredited POS:

1. POS generates a request data and sends it as a request to the E-SDC using Json via HTTP or Serial protocol;
2. E-SDC verifies format of the invoice;
3. E-SDC calculates taxes based on the current tax rates;
4. E-SDC sends the invoice data to the Secure Element for fiscalization providing current date and time and PIN code/password if required;
5. Secure element signs the invoice and returns the data to the E-SDC;
6. E-SDC produces a journal file – a textual representation of an invoice;
7. E-SDC generates a verification URL
8. [optionally] E-SDC creates QR Code – a graphical representation of a verification URL;
9. E-SDC creates an invoice with all mandatory elements (receipt data, previously generated signature, verification URL and journal), generates one-time key and encrypts the invoice using symmetric algorithm. The E-SDC encrypts one-time symmetric key using the Tax Service’s system public key and adds it to the package so the Tax Service’s system shall decrypt the symmetric key and access the package content once it arrives to the Service’s system.
10. E-SDC returns response to the POS and optionally generated journal data.

The previous process is illustrated on Figure 6.
Calculate Taxes

Taxes are calculated by an E-SDC after a POS has sent a valid request. Tax amount for particular items on an invoice are defined by the tax labels associated with an item.
Process of a tax calculation depends on:
- Invoice and Transaction Type
- the tax rates for each label associated with an item on an invoice
- the Type value of tax category to which the label belongs

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

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

          Example 1: An item has a total price 10$ and applied labels: A(5%) and B(6%).
          \[ A = \frac{10 \times 5}{(100 + \Sigma(5+6))} \]
          \[ B = \frac{10 \times 6}{(100 + \Sigma(5+6))} \]
          Tax amount for label A=0.4505$ and for label B=0.5405$.

      iii. If any of labels’ tax category is tax-on-total (category 1):
          - Tax amount for every label whose category type is tax-on-total (category 1) is:

          \[ \frac{item\ total\ amount}{(1 + \Sigma (all\ tax-on-total\ rates)/100)} \times \frac{label\ rate}{100} \]

          - Tax amount for every other label from category 0 is:

          \[ \frac{item\ total\ amount}{(1 + \Sigma (all\ tax-on-total\ rates)/100)} \times \frac{label\ rate}{(100 + \Sigma (all\ tax-on-net\ rates\ on\ item))} \]

          Example 2: Item has a total price 10$ and applied labels: A(5% tax-on-net), B(6% tax-on-net), C(3% tax-on-total) and F(4% tax-on-total).
          \[ C = \frac{10}{(1 + \Sigma(3+4)/100)} \times \frac{3}{100} \]
          \[ F = \frac{10}{(1 + \Sigma(3+4)/100)} \times \frac{4}{100} \]
          \[ A = \frac{10}{(1 + \Sigma(3-4)/100)} \times \frac{5}{(100 + \Sigma(5+6))} \]
          \[ B = \frac{10}{(1 + \Sigma(3+4)/100)} \times \frac{6}{(100 + \Sigma(5+6))} \]
          Tax amount for label A=0.4210$, for label B=0.5052$ for label C=0.2804$ and for label F=0.3738$.

   iv. Add calculated labels’ tax amounts to the label’s total amount sum.

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

   v. Add fixed tax amounts, multiplied with quantity, to the respective labels’ total amount sum.
3. After all items have been processed, calculate the tax amount for all tax categories found in the request. One tax category can be consisted of one or more tax labels (e.g. A, B...). Tax amount for a tax category is a sum of all label tax amounts related to the category.

Example 6: the request contains two items from the Example 1 and Example 2. Labels A and B are VAT category, C is STT category and F is ET category. Total VAT=1.9172$, STT=0.2804$, ET=0.3738$.

Once the Tax calculation is completed, assign GroupId of the active tax rate group to the field TaxGroupRevision of InvoiceFiscalizationResult.

**Rounding**

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

Examples:

3.44455555666 -> 3.4445
3.4440912345 -> 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:

<table>
<thead>
<tr>
<th>Start Offset</th>
<th>Invoice Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Bytes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Version</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>RequestedBy</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>SignedBy</td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>TotalCounter</td>
</tr>
<tr>
<td>21</td>
<td>4</td>
<td>DocTypeCounter</td>
</tr>
<tr>
<td>25</td>
<td>8</td>
<td>TotalAmount</td>
</tr>
<tr>
<td>29</td>
<td>8</td>
<td>DateAndTime</td>
</tr>
<tr>
<td>37</td>
<td>1</td>
<td>InvoiceType</td>
</tr>
<tr>
<td>38</td>
<td>1</td>
<td>TransactionType</td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>BuyerId Length</td>
</tr>
<tr>
<td>40</td>
<td>?</td>
<td>BuyerId</td>
</tr>
</tbody>
</table>
Encrypted Internal Data received from SE after Invoice Sign APDU command, 256 or 512 bytes long

Signature received from SE after Invoice Sign APDU command, 256 bytes long

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 Update Verification URL Command

Create QR Code

1. QR code contains a verification URL created in the previous section (Create Verification URL)
2. Base64 encoded string is created from GIF image bytes and attached to the Response

Create Textual Representation of an Invoice (Receipt)

Textual representation of a Receipt shall be created as described in the chapter Anatomy of 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 local time based.

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

How to Obtain an Tax Identification Number (TIN)

Digital certificate exported using 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 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.
Mapping Subject 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 mapping between a subject name/value pairs and invoice fields.

Subject Field **(bolded parameters are always present in the subject):**

- **E** = someone@test.taxcore.dti.rs
- **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

<table>
<thead>
<tr>
<th>Invoice Field</th>
<th>Subject Parameter Name</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TIN</strong></td>
<td>N/A</td>
<td>obtained by OID as explained in <a href="#">How to Obtain Tax Identification Number (TIN)</a></td>
</tr>
<tr>
<td><strong>Business Name</strong></td>
<td><strong>O</strong></td>
<td>Legal name under which business operates</td>
</tr>
<tr>
<td><strong>Shop Name</strong></td>
<td><strong>OU</strong></td>
<td>It may be the same as Business Name if Company HQ and sales location are the same.</td>
</tr>
<tr>
<td><strong>Address</strong></td>
<td><strong>STREET</strong></td>
<td>Street name and number</td>
</tr>
</tbody>
</table>
Creating an Audit Package

Once an invoice is created (InvoiceFiscalizationRequest and InvoiceFiscalizationResult) 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 all Date and Time data to UTC;
2. Generate a random one-time symmetric key for AES256;
3. Encrypt string JSON representation of the invoice using the one-time key;
4. Convert the encrypted invoice to base64 string and store it in the Payload field of Json Representation of the Invoice;
5. Get the TaxCore Public key using Export TaxCore Public Key APDU command.
6. Encrypt the one-time key using TaxCore public key, convert it to base64 string and store it in the Key field;
7. Encrypt Initialization Vector (IV) using the 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}-{Ordinal_Number}.json;
9. (Optional) Generate an QR code, and attach it to InvoiceFiscalizationResult (make sure that the QR code is not stored in the Audit Package);
10. Return InvoiceFiscalizationResult to the POS;
11. If the internet connection is available try to send the AuditData to Backend.Api as explained in the section Remote Audit;

Audit Process

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

There are two specific scenarios: Remote Audit and Local Audit. Basic rules and processes described in this section apply to both scenarios. Details will be explained in the separate sections.

An audit is always a synchronous 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 response (signed invoice and journal) from the Secure Element, it shall be encrypted and stored in the non-volatile memory.

An E-SDC device shall be fully functional during audit. The POS shall be able to sign new invoices as long as the Secure Element permits. There shall be a mechanism in place that is responsible for continuous operation of the Secure Element and E-SDC while audit packages are on its way to the Tax Service’s system.

Depending on the connection availability, audit may be triggered by the arrival of a signed invoice from the Secure Element or upon the insertion of an external memory device into the E-SDC. Regardless of the event which has triggered the audit, the following conversation shall take place between the E-SDC, the Tax Service’s system and the Secure Element:

1. E-SDC signals the beginning of the audit to the Secure Element (Invokes Start Audit APDU command);
2. The Secure Element returns ARP (256 bytes) to the E-SDC;
3. E-SDC starts the audit by sending audit data (over HTTPS) or dumping them on external memory (e.g. SD card, USB flash drive), starting with the oldest unaudited package, in piecemeal fashion. ARP is sent to the Tax Service’s system using the same communication channel;
4. If verification is successful, the Tax Service’s system shall generate a proof of audit (PoA) and return it as a Proof of Audit Command;
5. E-SDC receives the proof of audit command and passes the payload to the End Audit APDU command;
6. The Secure Element verifies if proof of audit is valid, meaning the audit data has been successfully received by the Tax Service’s system;
7. If proof of audit is valid, the Secure Element will conclude the audit process;
8. Consequently:
   a) audit packages created prior to the beginning of the final audit, are considered safe for deleting, because the Proof of Audit has been received by the Tax Service’s system;
   b) audit packages created after beginning of audit, are considered unaudited and E-SDC is responsible to preserve those audit packages and to submit them to the Tax Service’s system in the next audit cycle;
   c) audit packages created after end of audit are considered unaudited and E-SDC is responsible to preserve those audit packages and to submit them to the Tax Service’s system in the next audit cycle.

ARP should be generated and sent to Tax Service’s system periodically (for example, after each 200 audit packages are submitted to backend API or once per day regardless of number of submitted audit packages) if HTTP-based communication is used to submit audit packages.

ARP should be generated and saved as file each time at least one audit package is submitted to the Tax Service’s system using USB memory stick (File-based communication).

Audit process sequence is illustrated on Figure 7.
Remote Audit

Remote audit is the process of transferring data to the Tax Service’s system using the internet connection. It is the most common way to perform audit for any occasionally connected device.

An E-SDC checks if Backend.Api is reachable. If Backend.Api is reachable, the E-SDC authenticates the Tax Service’s system by using a server-side certificate installed on the Backend.Api endpoint, enabling HTTPS protocol. The Tax Service’s system authenticates the E-SDC using digital certificate issued on the Secure Element. The E-SDC starts sending audit packages, performing a series of audits until no more unaudited data is stored on its non-volatile memory.

Not all E-SDC devices are required in order to perform remote audit. If the network connection is not available due to the interruption of the service or a missing GPRS modem or network card, E-SDC will still be able to perform Local audit.

Local Audit

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

An audit is initiated by plugging an SD card or a USB Flash drive to an E-SDC device.
During the Local Audit the E-SDC performs the same steps as the Remote audit, but instead of submitting the ARP and Audit packages to Backend.Api, those files are saved to an SD Card or a USB Flash Drive.

Data formats for the file-based Local audits are described in the section File-Based Communication.

Submitting Data Using a Web Application

Audit packages (up to 30Mb) could be sent to a Tax Service using a public web site. The Service’s system shall verify received audit packages and generate the proof of audit as a response. A user will be required to manually delete audit packages from the media and save received proof of audit for a later use.

Completing an Audit in Progress

A taxpayer inserts media with proof of audit file into an E-SDC. 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 for processing.

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.

Proof of Audit

Proof of audit is generated by the Tax Service’s system once all expected audit packages have been received and securely stored on the Tax Service’s system.

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

Figure 8 illustrates the simplest case, where no additional audit packages are generated during the whole audit process, as following:

1. Create an audit package
2. Initiate the Audit process by invoking BeginAudit APDU command
3. Receive a proof of audit and pass it to EndAudit APDU command
4. If EndAudit returns value true you can safely delete the audit package
5. If EndAudit returns value false, continue until a valid proof of audit is received

Case 2 – Audit is performed after multiple audit packages have been created (Figure 9 Audit process timeline with packages created after audit start)

1. Create audit packages 1-3
2. Initiate the audit process by invoking BeginAudit APDU command
3. Continue to fiscalize invoices and create audit packages 4-6
4. Receive a proof of audit and pass it to EndAudit APDU command
5. If EndAudit APDU command returns value true you can safely delete 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.

**Case 3 – Audit is started multiple times before the first proof of audit arrived** (Figure 10 Audit process timeline with multiple audit starts)

1. Create Audit Packages 1-3
2. Initiate the Audit process by invoking BeginAudit APDU command
3. Continue to fiscalize invoices and create Audit Packages 4 and 5
4. Initiate another audit process by invoking BeginAudit APDU command – the previous audit is canceled
5. Continue to fiscalize invoices and create Audit Package 6
6. Receive the Proof of Audit and pass it to EndAudit APDU command
7. If EndAudit returns value true you can safely delete audit packages 1-5 because it is the last BeginAudit being invoked by E-SDC. The Audit package 6 is created after the last call to BeginAudit APDU command so it is not audited in this run
8. If EndAudit APDU command returns value false, continue until a valid proof of audit is received.

**Notifications**

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

The 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 the visual notifications.
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 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)

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, manufacturer shall provide and document a simple way to check, set and keep date and time in sync on the E-SDC.

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 Service system shall be able to dump data from 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 (Backend.Api, Secure element Applet, PKI Applet or SD Card/USB Flash Drive) required to fulfil its primary role – to safeguard a transaction and to transfer the audit packages to the Tax Service’s system.

Accredited POS systems can communicate with the E-SDC using the Serial protocol or JSON via HTTP protocol. It is up to an E-SDC developer to choose if only one or both of these protocols will be implemented by an E-SDC.

Status and Error Codes

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

<table>
<thead>
<tr>
<th>Code</th>
<th>0-Info 1-Warning 2-Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000</td>
<td>All OK</td>
<td>Command is executed without warnings or errors</td>
</tr>
<tr>
<td>0100</td>
<td>Pin OK</td>
<td>This code indicates that the provided PIN code is correct</td>
</tr>
<tr>
<td>0210</td>
<td>Internet Available</td>
<td>Internet connection to Backend.Api is available (optional)</td>
</tr>
<tr>
<td>0220</td>
<td>Internet Unavailable</td>
<td>Internet connection to Backend.Api is not available (optional)</td>
</tr>
<tr>
<td>WARNINGS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td>Storage 90% Full</td>
<td>Storage used to store audit packages is 90% percent full. It is time to perform the audit.</td>
</tr>
<tr>
<td>1300</td>
<td>Smart Card is not present</td>
<td>Secure element card is not inserted in the E-SDC smart card reader</td>
</tr>
<tr>
<td>1400</td>
<td>Audit Required</td>
<td>Total Sale and refund amount reached 75% of SE limit. It is time to perform the audit.</td>
</tr>
<tr>
<td>1500</td>
<td>Pin Code Required</td>
<td>Indicates that POS shall provide the PIN code</td>
</tr>
<tr>
<td>1999</td>
<td>Undefined Warning</td>
<td>Something is wrong but specific warning is not defined for that situation. Manufacturer can use manufacturer-specific codes to describe warning in more details</td>
</tr>
<tr>
<td>ERRORS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2100</td>
<td>Pin Not OK</td>
<td>PIN code sent by the POS is invalid</td>
</tr>
<tr>
<td>2210</td>
<td>SE Locked</td>
<td>Secure element is locked. No additional invoices can be signed before the audit is completed</td>
</tr>
<tr>
<td>2220</td>
<td>SE Communication Failed</td>
<td>E-SDC cannot connect to the Secure element applet</td>
</tr>
<tr>
<td>2230</td>
<td>SE Protocol Mismatch</td>
<td>Secure element does not support requested protocol version</td>
</tr>
</tbody>
</table>
### JSON Based POS to E-SDC Protocol

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

- Get Status
- Verify PIN
- Sign Invoice
- Attention
- Get Last Signed Invoice

#### Get Status Command

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

**Request Data**

JSON data field with predefined string value.

**Example**

```json
{
   “GS” : “GetStatus”
}
```

**Response Data**

JSON formatted data in accordance with Table 4 Get Status response data.
<table>
<thead>
<tr>
<th>Filed</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsPinRequired</td>
<td>If PIN is not entered, or if wrong PIN is entered in the previous attempt, this field shall be set to true; otherwise set to false</td>
<td>true</td>
</tr>
<tr>
<td>AuditRequired</td>
<td>If Audit is required, this field shall be set to true. Audit is required if Total Amount of all invoices is 75% or more of Maximum Limit. Maximum Limit and Total Amount are obtained from the Secure element using Amount Status APDU command</td>
<td>False if Total Amount is 1554879 Maximum Limit is 9000000 True if Total Amount is 7504899 Maximum Limit is 10000000</td>
</tr>
<tr>
<td>DT</td>
<td>Current Local Date and Time in ISO 8601 format</td>
<td>2017-08-30T11:53:05+00:00</td>
</tr>
<tr>
<td>LastInvoiceNumber</td>
<td>Invoice number of the last invoice signed by this E-SDC.</td>
<td>ORG674J1-ORG674J1-98637</td>
</tr>
<tr>
<td>ProtocolVersion</td>
<td>Always return 1.0.0.0</td>
<td>1.0.0.0</td>
</tr>
<tr>
<td>SecureElementVersion</td>
<td>Version obtained from the Secure element using Get Secure Element Version APDU command</td>
<td>1.0.0.0</td>
</tr>
<tr>
<td>HardwareVersion</td>
<td>Manufacturer-specific hardware version, if applicable</td>
<td>1.2.7.21</td>
</tr>
<tr>
<td>SoftwareVersion</td>
<td>Manufacturer-specific software version</td>
<td>1.7.6.5</td>
</tr>
<tr>
<td>DeviceSerialNumber</td>
<td>Manufacturer specific serial number</td>
<td>1289A24EB67F22C1</td>
</tr>
<tr>
<td>Make</td>
<td>Manufacturer specific Make Name</td>
<td>Acme</td>
</tr>
<tr>
<td>Model</td>
<td>Manufacturer specific Model Name</td>
<td>The Device 442</td>
</tr>
<tr>
<td>MSSC</td>
<td>Manufacturer Specific Errors, Warnings and info messages</td>
<td>Array of error codes</td>
</tr>
<tr>
<td>GSC</td>
<td>General Errors, Warnings and info messages defined in Status and Error Codes section</td>
<td>Array of error codes</td>
</tr>
</tbody>
</table>

Table 4 Get Status response data

Example

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

This command is used to verify a PIN entered by a cashier on a POS. Once the PIN is entered, **Pin Verify** APDU command shall be invoked and PIN passed to the Secure element. If command is successfully executed, the E-SDC shall store the PIN in the volatile memory (RAM) until the E-SDC is switched off or the smart card is removed from the reader.

**Request Data**

JSON string with PIN code sent from POS.

Example

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

**Response Data**

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

Example

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

Sign Invoice Command

**Invoice Request**

**Data Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DateAndTimeOfIssue</td>
<td>Current <strong>Local</strong> Date and Time in ISO 8601 format.</td>
</tr>
<tr>
<td>IT</td>
<td>Invoice Type enumeration value.</td>
</tr>
<tr>
<td>TT</td>
<td>Transaction Type enumeration value.</td>
</tr>
<tr>
<td>PaymentType</td>
<td>Payment Type enumeration value.</td>
</tr>
<tr>
<td>Cashier</td>
<td>Cashier’s identification.</td>
</tr>
<tr>
<td>BD</td>
<td>Taxpayer ID of the Buyer. <strong>It is mandatory for B2B transactions, otherwise it’s optional.</strong></td>
</tr>
<tr>
<td>BuyerCostCenterId</td>
<td>Cost Center ID provided by buyer to the cashier in case Buyer’s company wants to track spending in Taxpayer Portal. <strong>It is optional and may exist only for B2B transactions, otherwise it shall be ignored by E-SDC.</strong></td>
</tr>
<tr>
<td>InvoiceNumber</td>
<td>Invoice number generated by a POS.</td>
</tr>
<tr>
<td>ReferentDocumentNumber</td>
<td>Mandatory only in case Invoice Type is <strong>Refund or Copy</strong>. In both cases, this field must contain Invoice Number of previously issued Invoice or Refund. In any other case (for example Normal Sale invoice) this field is optional. <strong>ASCII, in RequestedBy-SignedBy-OrdinalNumber format</strong></td>
</tr>
<tr>
<td>Items (n)</td>
<td>Each invoice contains at least one Item in Items collection</td>
</tr>
<tr>
<td>GTIN</td>
<td>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.</td>
</tr>
<tr>
<td>Name</td>
<td>Human readable name of the product or service.</td>
</tr>
<tr>
<td>Quantity</td>
<td>Quantity of an item, e.g. 2 (pieces), 0.100 (grams).</td>
</tr>
</tbody>
</table>
**UnitPrice**  
Unit price of the line item. Does not take part in tax calculation.

**Discount**  
Discount applied, before **TotalAmount** is accounted for. It's not part of tax calculation.

**Labels**  
Array of labels. Each Label represents one of the Tax Rates applied on invoice item. Tax Items are calculated based on TotalAmount and applied Labels as described in Calculate Taxes section. In case no taxes are applicable on item this field is optional.

**TotalAmount**  
Gross price for the line item.

**Options**  
Key/value collection defines output of V-SDC/E-SDC invoice fiscalization, to optimize resources.  
*Key: OmitQRCodeGen*  
Value: "1" to omit QR Code generation by E-SDC and "0" to generate and return QR code.  
*Key: OmitTextualRepresentation*  
Value: "1" to omit generation of textual representation by E-SDC and "0" to generate return textual representation to POS.

**Hash**  
Base64 encoded MD5 hash of the request data. It is used only for later invoice search.

---

**Model**  
**InvoiceFiscalizationRequest**

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

**inline_model**

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

**Item**

```json
Item {
  GTIN (string, optional) MinLength:8 MaxLength:14,
  Name (string) Unicode MaxLength:2048,
  Quantity (number) Decimal(14,3) MinValue:0.001,
  UnitPrice (number, optional) Decimal(14,2),
  Discount (number, optional) Decimal(14,2),
  Labels (Array[string], optional) MinLength:0,
  TotalAmount (number) Decimal(14,2)
}
```

**Example**

```json
{
  "DateAndTimeOfIssue": "2017-06-15T08:56:23.286Z",
  "Cashier": "123456789",
```
"IT": "Normal",
"TT": "Sale",
"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
  }
],
"Hash": "W33lEEgkSRsqTFMO86a8Og=="

Invoice Response
Data Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RequestedBy</td>
<td>UID of Secure Element digital certificate.</td>
</tr>
<tr>
<td>SignedBy</td>
<td>UID of Secure Element digital certificate.</td>
</tr>
<tr>
<td>DT</td>
<td>Local date and time in ISO 8601 format provided by E-SDC.</td>
</tr>
<tr>
<td>IC</td>
<td>Invoice Counter in format TransactionTypeCounter/TotalCounter InvoiceCounterExtension For Example: 14/17NS</td>
</tr>
<tr>
<td>InvoiceCounterExtension</td>
<td>First letters of Transaction Type and Invoice Type of the invoice. NS for Normal Sale, CR – Copy Refund, TS – Training Sale etc.</td>
</tr>
<tr>
<td>IN</td>
<td>Invoice number in format RequestedBy-SignedBy-TotalCounter</td>
</tr>
<tr>
<td>VerificationUrl</td>
<td>VerificationURL generated in Create Verification URL process</td>
</tr>
<tr>
<td>VerificationQRCode</td>
<td>Base64 encoded byte array of GIF image created in Create QR Code process</td>
</tr>
<tr>
<td>Journal</td>
<td>Textual Representation of the invoice created in Create Textual Representation of an Invoice (Receipt) process</td>
</tr>
<tr>
<td>Messages</td>
<td>Custom human readable message that shall be printed or displayed by POS.</td>
</tr>
<tr>
<td>TotalCounter</td>
<td>Total number of invoices signed by Secure Element. Returned by Sign Invoice APDU command</td>
</tr>
<tr>
<td>TransactionTypeCounter</td>
<td>Total number of invoices of a requested type. Returned by Sign Invoice APDU command</td>
</tr>
<tr>
<td>TotalAmount</td>
<td>Sum of all Items – total payable by customer</td>
</tr>
<tr>
<td>ID</td>
<td>Encrypted Internal Data - Base64 encoded byte array returned by Sign Invoice APDU command</td>
</tr>
<tr>
<td>S</td>
<td>Signature - Base64 encoded byte array returned by Sign Invoice APDU command</td>
</tr>
<tr>
<td>TaxItems</td>
<td>Array of TaxItem entities</td>
</tr>
<tr>
<td>Label</td>
<td>Tax Label (A, F, G)</td>
</tr>
<tr>
<td>Name</td>
<td>Tax Category Name (i.e. VAT, Consumption)</td>
</tr>
<tr>
<td><strong>Rate</strong></td>
<td>Tax rate percentage for Label (i.e. 12.50%)</td>
</tr>
<tr>
<td><strong>Amount</strong></td>
<td>Tax amount calculated by E-SDC during invoice fiscalization</td>
</tr>
<tr>
<td><strong>Hash</strong></td>
<td>Hash received from POS in request field Hash</td>
</tr>
<tr>
<td><strong>BusinessName</strong></td>
<td>Taxpayer Business Name obtained from digital certificate subject field</td>
</tr>
<tr>
<td><strong>LocationName</strong></td>
<td>Location Name obtained from digital certificate subject field</td>
</tr>
<tr>
<td><strong>Address</strong></td>
<td>Street address obtained from digital certificate subject field</td>
</tr>
<tr>
<td><strong>TIN</strong></td>
<td>Tax Identification Number obtained from digital certificate subject field</td>
</tr>
<tr>
<td><strong>District</strong></td>
<td>District obtained from digital certificate subject field</td>
</tr>
</tbody>
</table>
| **MRC** | Manufacturer registration code (in format MakeCode-SW-Serial), mandatory for audit package, optional for POS response.  
- MakeCode: unique 2 characters obtained on Tax Service Accreditation  
- SW: software version  
- Serial: manufacturer serial number (max 32 characters) |

**Model**

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

**TaxItem**

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

**Example**

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

],
"VerificationUrl": "https://staging.vms.frcs.org.fj/v/?vl=AVAyMlZDOFZSSlRKQzVWiUBAAAA AQAAAAAuGQ4AAAFe%2BWC3gAAAAAbDPPkm7R91INLP1erP%2Bh3UQswb%2F2x5XyiEljyCI1QyqHx626Fru PNvNksB7wMoG2p85uvb9txF2cdNy15Jzhb3q7TLF%2B2q0mRs3EcAykuVF05mFbTrtgQmUR0ZE761ciqa xvAEGK831lqZ2HVz8myqHnaA1iu2FuNq4q2wQ4q39hb%2B6pvyhY%2B2B8c3539GNgPm4eQeB1%2BCThtjJ pPCYL1hrVKPjQbQEE6Fsm2II0YeeQwENGCQaxthMhm5sJTT4B3J2F1h2hTQyufWoiSko3oaAmBAZseDxqS4 oEEQ3Lum3%2F75tGcuUweEnRGfOEPYeiqa0kts25dSg18hrjD4PUdZ2QksSeIDmjLmsqZolGmicyda jhMN2eMeo2B9LZ2FHlxs0RkboOlrArVfGQ%2B9MFbmy3s1LCE1T6myTAC2H2CvQ%2Bc0ME%2FBeuynC kQCO6BBV39ZNm8ynRagmsE0s1kQdty66QmphemBCCOx76u2F41CjpxO00xc2F6zNR8Se1MAoDPVH3PU7 l1QdToxFXY3pvwSqtK%2FUY5QvQpuQpL6kUXr1qEcjt2uj6Qslh%2BbgwjEZvPb%2ByhmEoQc19A4ND UtoUjPjPITB0xP04VvmVe2Bgv4UnpAKQigAv%2FyWeDeD9noHDGciFsFLZCJ0IXMSle0%2B1f%2B1FE2YYX 84gH7n7Ncnpn",
"VerificationQRCode": null,
"Journal": "============ FISCAL INVOICE ============

TIN:                           502579006
Company:                          Golf V
Store:                         Sun Store
Address:                     7 Someplace
District:                           Suva
Cashier TIN:                   123456789
POS number:                  POS2017/998
POS time:             2017-06-15 8:56:23

--------------- NORMAL SALE ---------------

Name    Price        Qty.          Total
Sport
-100 Helmet, Blue (A)
34.23           2          68.46

----------------------------------------
Total Purchase:                    68.46

----------------- Payment Method: Cash -----------------

Label        Name     Rate           Tax
A             VAT    9.00%          5.65

----------------------------------------
Total Tax:                          5.65

SDC Invoice No:    7AF40923-E3830A31-234
Invoice Counter:               230/234NS

---------------- END OF FISCAL INVOICE ----------------

"Messages": "Success",
"SignedBy": "E3830A31",
"ID": "BsMM+SbtH0jU88uJ6s/6HdRcZBv9drzFgqIScuPIKUEqHPfoWw49U2SwHvAygbYuKdzm69sb23F/YKd1i XKlmyEEmytrMTsxK6Od2gcRrKRRV/TmVYo0uCBzE5kTqvYkprG9pUQyrZeJZwtvYdXP5aVkoedro117+5Of irbBd1Antzt4QPm/KFj5hxzdKBg0+y+bsrpxY2mMkg8JjguUetUo+NtAQToVkbYgjRgR5cPYQYIeqp3Ee aNLywiNpeEkn+wHNNDK4VqgjmSjegCbw8mWr2DHNigQQ2vcstBcm93vm0ZxSb45eBF+gq/O5Kpo63x3ax1 KDXyGuN3gQQ=",
"S": "HWfEFJLEniA5oyzLkmaC5hoqsHw04TDdndjHqPv5sF4S58Q7ETpGzlPKQ1Rn0PvTHw3sibCChwCE+pskwa th2qP0PNDGBV9HwspwPAbDwgQqb9/cz2/Mjwu0rJ8DrJ2EMbcu0wXyvungse+r+Q3Qb0TjzXP+szUfEg HtTDw1g2r10yJUHu65M6X216bikQsv1G50R165jKix+2pPF69ajgo7d11iK/LB/4MIXGaV3qsoe2E9e HCPODQ31af1Z6SEZ2z1MTzuL8p3vol+FJ6QCKiOAL/8sCng/Z6Bw4AohUny2QidCFZEpXqPoh/H1txNmF1/ OIB+s+2XZw==",
"TotalCounter": 234,
"TransactionTypeCounter": 230,
"TotalAmount": 68.46,
"Hash": "W33lEEgkSRsqTFMO86a8Og==",
"BusinessName": "Golf V",
"TIN": "502579006",
"LocationName": "Sun Store",
"Address": "7 Someplace",
"District": "Suva"
}

Mapping Fiscal Invoice to Fiscal Receipt

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

This command is used by POS to verify if ESDC is available. The command shall be used prior to Sign Invoice or Send Pin requests. This significantly lowers possibility for communication errors, including timeout errors. Only if a valid response is received, the POS shall immediately send the next command.
Request Data
JSON data field with predefined string value.

Example
{
   "ATT": "Attention"
}

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

Example
{
   "ATT_GSC": "0000"
}

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

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

Example
{
   "GI": "MDNDN0MwQUNFMzk1RDgwMQ=="
}

Response Data
Refer to response for Sign Invoice Command.

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

Data Fields

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

Model

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

ModelError {
    FieldName (Array[ErrorCode (string)])
}
Implementation example in C# programming language:

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

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

POS to E-SDC Communication over HTTP Protocol

E-SDC device 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.

Physical connection to a network can be done 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 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. Technical implementation of these features is in the scope of E-SDC manufacturer.

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

Get Status Command

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

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

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

Verify PIN Command

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

Example: http://192.168.88.112:8888/api/Status/VerifyPin
Attention Command
HTTP POST request data is sent to: http://<ESDC_ip_address>:<port>/api/Status/Attention
Example: http://192.168.88.112:8888/api/Status/Attention

Sign Invoice Command
HTTP POST request data is sent to: http://<ESDC_ip_address>:<port>/api/Sign/SignInvoice
Example: http://192.168.88.112:8888/api/Sign/SignInvoice

Get Last Signed Invoice Command
HTTP POST request data is sent to: http://<ESDC_ip_address>:<port>/api/Sign/GetSignedInvoice
Example: http://192.168.88.112:8888/api/Sign/GetSignedInvoice

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Databits</td>
<td>8</td>
</tr>
<tr>
<td>Parity</td>
<td>None</td>
</tr>
<tr>
<td>Stopbits</td>
<td>1</td>
</tr>
<tr>
<td>Baudrate</td>
<td>115200 b/s</td>
</tr>
<tr>
<td>Handshake</td>
<td>None</td>
</tr>
</tbody>
</table>

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

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

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

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

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

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

SLIP defines the following special bytes to be used:

<table>
<thead>
<tr>
<th>Hex value</th>
<th>Dec Value</th>
<th>Oct Value</th>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xC0</td>
<td>192</td>
<td>300</td>
<td>END</td>
<td>Frame End</td>
</tr>
<tr>
<td>0xDB</td>
<td>219</td>
<td>333</td>
<td>ESC</td>
<td>Frame Escape</td>
</tr>
<tr>
<td>0xDC</td>
<td>220</td>
<td>334</td>
<td>ESC_END</td>
<td>Transposed Frame End</td>
</tr>
<tr>
<td>0xDD</td>
<td>221</td>
<td>335</td>
<td>ESC_ESC</td>
<td>Transposed Frame Escape</td>
</tr>
</tbody>
</table>

SLIP modifies a standard TCP/IP datagram by

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

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

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

Request

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

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

Example

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

507B225650494E223A223334227DB6C4C0

Response

Every response sent by E-SDC to POS over a serial communication protocol is a SLIP packet consisted of the following segments:
Payload is a UTF-8 encoded JSON based command response as defined in section JSON Based POS to E-SDC Protocol.

Checksum is Fletcher-16 checksum calculated on <Payload> segment, as defined in section Fletcher-16 checksum.

Example

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

7B225650494E5F475343223A2230313030227D67F8C0

Error Response

In case E-SDC fails to process a command, error response sent by E-SDC to POS over a serial communication protocol is a SLIP packet consisted of the following segments:

0xFF<Payload><Checksum><SLIP End>

Payload is a UTF-8 encoded JSON based error message format as defined in section Error Messages Format.

Checksum is Fletcher-16 checksum calculated on <Payload> segment, as defined in section Fletcher-16 checksum.

Fletcher-16 checksum

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

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

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

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

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

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

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

In case of timeout error on Get Signature request, POS shall repeat this request.

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

GetStatus Command

GetStatus Command is used by a POS to gather information on state of the connected E-SDC device.
Command Identifier: S (“0x53” in hexadecimal).

Verify PIN Command

This command is used to provide a PIN code to the Secure element.
Command Identifier: P (“0x50” in hexadecimal).

Sign Invoice Command

Sign Invoice Command performs the tax calculation, creates a verification URL, applies a digital signature and optionally generates a QR code and a textual representation of the invoice.
Command Identifier: I (“0x49” in hexadecimal).

Attention Command

This command is used by POS to verify if E-SDC is available.
Command Identifier: A (“0x41” in hexadecimal).

Get Signed Invoice Command

This command is used by POS to get the last signed invoice.
Command Identifier: G (“0x47” in hexadecimal)

POS to E-SDC Communication over TCP

Will be supported in the future revision of the document.
E-SDC to Secure element (APDU commands)

Communication with a Secure element shall be 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 type of usage:

1. Personalization
2. Fiscalization
3. Audit

Notes:

1. P1 and P2 values are not considered in request processing, except for the Select Applet Command
2. All APDU commands are sent to the Smart Card using T1 communication protocol
3. All values are submitted to 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)

General Commands

Secure element Applet is installed as a non-default applet on a smart card. Before any APDU command is invoked, the applet shall be selected using the standard Select command.

Select Applet

As previously mentioned, the Smart Card has two applets installed. This command shall select the Secure element applet and route a subsequent APDU commands to it.

IsoCase: Case4Short
Class: 0x00
Instruction: 0xa4
P1: 0x04
P2: 0x00
Example: 0x00 0xA4 0x04 0x00 0x10 0x40 0x00 0x07 0x48 0x46 0x4A 0x49 0x2D 0x61 0x78 0x43 0x6F 0x72 0x65 0x00

Export Certificate

IsoCase: Case2Extended
Class: 0x88
Instruction: 0x04
Example: 0x88 0x04 0x04 0x00 0x00 0x00
Exports taxpayer certificate in a DER format. This certificate contains a location data that shall be present on the textual representation of an invoice.

Get Secure Element Version

IsoCase: Case2Short
Class: 0x88
Instruction: 0x09
Example: 0x88 0x09 0x04 0x00 0x00

**Fiscalization**

**PIN Verify**

PIN verification is a method that “unlocks” a card for invoice signing and other operations protected by PIN code. PIN shall be in a decimal format, example PIN:2017 is represented as 0x02, 0x00, 0x01, 0x07

IsoCase: Case3Short
Class: 0x88
Instruction: 0x11
Example: 0x88 0x11 0x04 0x00 0x04 0x02 0x00 0x01 0x07 (for Pin 2017)

**Sign Invoice**

Signs invoice and returns fiscalization data for a submitted invoice.

IsoCase: Case4Extended
Class: 0x88
Instruction: 0x13

**Request Data**

<table>
<thead>
<tr>
<th>Start (byte)</th>
<th>Length (bytes)</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>Date/time</td>
<td>E-SDC timestamp UTC time in Unix Timestamp.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: 1495018011910 is 2017-05-17T10:46:51.910Z</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>Taxpayer ID</td>
<td>Hex encoded byte array, leading bytes filled with 0x00; MSB are sent first</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: Taxpayer ID = 928615467,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Byte array = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x00, 0x00, 0x00, 0x00, 0x00, 0x39, 0x32, 0x38, 0x36, 0x31,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x35, 0x34, 0x36, 0x37}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(byte 0x37 is sent last to SE)</td>
</tr>
<tr>
<td>28</td>
<td>20</td>
<td>Buyer ID</td>
<td>If unknown, leave zeroes, formatting is the same as for Taxpayer ID</td>
</tr>
<tr>
<td>48</td>
<td>1</td>
<td>Invoice type</td>
<td>Values 0, 1, 2, 3, as explained in section Model</td>
</tr>
<tr>
<td>49</td>
<td>1</td>
<td>Transaction Type</td>
<td>Sale=0, Refund=1</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
<td>Invoice amount</td>
<td>Sale or refund total amount (including taxes) - depends on applied tax types</td>
</tr>
<tr>
<td>Number of tax categories</td>
<td>Defines the number of tax categories which appears on the invoice (value between 0 and 26). The following data structure <strong>Tax categories</strong> shall be repeated exactly this number of times.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tax categories</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59 [1] [Tax category ID]</td>
<td>The first tax category’s OrderID, as explained in Tax Rates section (mandatory if <strong>Number of tax categories</strong> &gt; 0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60 [8] [Tax category amount]</td>
<td>The first total tax amount for the category specified in preceding field <strong>Tax category ID</strong> (mandatory if <strong>Number of tax categories</strong> &gt; 0)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>68 [1] [Tax category ID]</td>
<td>The next tax category’s OrderID (mandatory if <strong>Number of tax categories</strong> &gt; 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>69 [8] [Tax category amount]</td>
<td>The next total tax amount for the category specified in preceding field <strong>Tax category ID</strong> (mandatory if <strong>Number of tax categories</strong> &gt; 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77 ...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Response Data**

<table>
<thead>
<tr>
<th>Start (byte)</th>
<th>Length (bytes)</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>Date/time</td>
<td>Same as data sent from E-SDC to SE</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>Taxpayer ID</td>
<td>Same as data sent from E-SDC to SE</td>
</tr>
<tr>
<td>28</td>
<td>20</td>
<td>Buyer ID</td>
<td>Same as data sent from E-SDC to SE</td>
</tr>
<tr>
<td>48</td>
<td>1</td>
<td>Invoice type</td>
<td>Same as data sent from E-SDC to SE</td>
</tr>
<tr>
<td>49</td>
<td>1</td>
<td>Transaction type</td>
<td>Same as data sent from E-SDC to SE</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
<td>Invoice amount</td>
<td>Same as data sent from E-SDC to SE</td>
</tr>
<tr>
<td>58</td>
<td>8</td>
<td>Sale or refund counter</td>
<td>Depends of request’s <strong>Tax type</strong> field</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>8</td>
<td>Total counter value</td>
<td>unsigned int 64bit big endian, (sale+refund)</td>
</tr>
<tr>
<td>74</td>
<td>256</td>
<td>512</td>
<td>Encrypted Internal Data</td>
</tr>
<tr>
<td>330</td>
<td>586</td>
<td>256</td>
<td>Digital signature</td>
</tr>
</tbody>
</table>
Amount Status

Returns 16 bytes long data structure (8 bytes for sum SALE and REFUND, and 8 bytes for Limit Amount)

IsoCase: Case2Short
Class: 0x88
Instruction: 0x14
Example: 0x88 0x14 0x04 0x00 0x00

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 for Audits.

IsoCase: Case2Extended
Class: 0x88
Instruction: 0x07
Example: 0x88 0x07 0x04 0x00 0x00 0x00 0x00

Export Internal Data

Exports encrypted Internal Data structure only (256 or 512 bytes).

Class: 0x88
Instruction: 0x12
Example: 0x88 0x12 0x04 0x00 0x00

Start Audit

Notifies Secure element that audit process has been initialized by E-SDC.

Secure element returns encrypted message that shall be submitted to TaxCore as Audit Request Payload.

IsoCase: Case2Extended
Class: 0x88
Instruction: 0x21
Example: 0x88 0x21 0x04 0x00 0x00 0x00 0x00

End Audit

Notifies Secure element that audit process has been finalized by TaxCore. If APDU Command status is OK (0x90 0x00) consider audit operation is successfully completed.
IsoCase: Case3Extended
Class: 0x88
Instruction: 0x20
Example: 0x88 0x20 0x04 0x00 0x01 0x00 0x53 0x8B 0x46 0xC8 0x86 0x48 0x74 0xE4 0x33 0x46 0xA7 0x13 0x81 0x58 0x5E 0xF4 0xD6 0x88 0xB9 0x92 0x42 0x23 0x1B 0xCA 0x60 0xAD 0xF4 0x98 0xF6 0x99 0xC2 0x4E 0x77 0x2E 0xF9 0x6F 0xF8 0x72 0x99 0xBB 0x20 0x16 0x2F 0xAD 0xC6 0x97 0xCD 0x42 0x04 0x0A 0xF1 0x96 0xF8 0x22 0x00 0x7C 0xD4 0xD1 0xE9 0xE9 0x41 0x19 0x33 0x24 0xF4 0xB0 0x01 0xE1 0x6D 0x40 0xEB 0x9D 0xE1 0xC3 0xBE 0x22 0x67 0x4B 0xAC 0xA6 0x23 0x99 0x3F 0xF5 0xA5 0xA2 0x7F 0x67 0x7A 0xA0 0x81 0x8B 0xC8 0x3E 0x45 0x08 0x7E 0x34 0xCD 0xEA 0x2F 0x0B 0xCF 0x59 0x5F 0xCE 0x9D 0x6B 0xFE 0x36 0x80 0x85 0x86 0x40 0xD3 0xB4 0x3F 0xD7 0x06 0x90 0x79 0x35 0xCE 0x07 0x4B 0x9F 0xAA 0xBB 0x70 0x95 0x5F 0xAC 0x15 0x40 0xE2 0x8A 0xD0 0x5C 0x81 0x27 0x72 0x14 0x00 0xBD 0x68 0x52 0x9B 0x23 0xE5 0xD2 0x23 0x63 0x62 0x87 0x32 0x98 0xA2 0x7A 0x2E 0xDD 0x0A 0x10 0x0E 0x2B 0xSE 0xA0 0x66 0x89 0xEF 0xDD 0x7E 0x61 0xF9 0x6A 0xA7 0x34 0xE 0xFE 0xCF 0x6F 0xA6 0xFC 0x67 0xFA 0x88 0xC2 0xA4 0xD5 0x13 0x31 0x12 0x5F 0xC1 0xE8 0x28 0x98 0x87 0x2C 0x43 0xF9 0x11 0x1E 0xC9 0x76 0x16 0xD6 0x9D 0x9D 0x68 0x89 0x7D 0x85 0x0D 0x61 0xB4 0x12 0xB3 0xB5 0x95 0x84 0xCD 0xCA 0x44 0x92 0x9E 0x10 0x22 0x4A 0x10 0x8F 0xB1 0xEE 0xC1 0x1D 0xD4 0xAF

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.

<table>
<thead>
<tr>
<th>Error code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x6301</td>
<td>PIN verification required before executing a command</td>
</tr>
<tr>
<td>0x6302</td>
<td>PIN verification failed – wrong PIN code</td>
</tr>
<tr>
<td>0x6303</td>
<td>Wrong PIN size</td>
</tr>
<tr>
<td>0x6304</td>
<td>Maximum number of tax categories exceeded</td>
</tr>
<tr>
<td>0x6305</td>
<td>Maximum amount of tax exceeded (Sign Invoice) or Audit has not been started yet (End Audit)</td>
</tr>
<tr>
<td>0x63FF</td>
<td>8 byte arithmetic operation overflow</td>
</tr>
<tr>
<td>0x6700</td>
<td>Data shall be 256 bytes long</td>
</tr>
<tr>
<td>0x6A80</td>
<td>Audit Identification is not valid</td>
</tr>
</tbody>
</table>

PKI Applet

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

A role of the PKI applet is to support the secure communication and client certificate authentication of the E-SDC with Backend.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.
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 certificate from PKI Applet using PKCS#11 API. Pin for PKI Applet is the same as PIN for SE.

**Required Drivers**

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 OpenSC library is required. OpenSC libraries and tools are freely available on [https://github.com/OpenSC](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
   i.     # apt-get install libudev-dev
   ii.    # wget [https://alioth.debian.org/frs/download.php/file/4126/pcsc-lite-x.y.z.tar.bz2](https://alioth.debian.org/frs/download.php/file/4126/pcsc-lite-x.y.z.tar.bz2)
   iii.   # tar -xf pcsc-lite-x.y.z.tar.bz2
   iv.    # cd pcsc-lite-x.y.z
   v.     # ./configure
   vi.    # make
   vii.   # make install
   viii.  # aptitude install libusb-1.0-0-dev
   ix.    # wget [https://alioth.debian.org/frs/download.php/file/4111/ccid-x.y.z.tar.bz2](https://alioth.debian.org/frs/download.php/file/4111/ccid-x.y.z.tar.bz2)
   x.     # tar -xf ccid-x.y.z.tar.bz2
   xi.    # cd ccid-x.y.z
   xii.   # ./configure
   xiii.  # make
   xiv.   # make install
   xv.    copy 92_pcsclinux_rules file from src directory to /etc/udev/rules.d/
   xvi.   # aptitude install libltdl-dev
   xviii. # tar -xf openct_x.y.z.orig.tar.gz
   xix.   # cd openct_x.y.z
   xx.    # ./configure
   xxi.   # make
   xxii.  # make all

2. Install OpenSSL development library
   i.     # apt-get install libssl-dev

3. Install OpenSC package
1. # wget http://cznic.dl.sourceforge.net/project/opensc/OpenSC/opensc-x.y.z/opensc-x.y.z.tar.gz
2. # tar -xf opensc-x.y.z.tar.gz
3. # cd opensc-x.y.z
4. # ./configure
5. # make
6. # make install
7. Run opensc-tool command from terminal
8. If message that libopensc.so.3 cannot be loaded find it with find / -name “libopensc.so”
9. Copy found library to /usr/lib

4. Install libp11 library
   i. # apt-get install libp11-2

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

After above steps are executed, the certificate shall be accessible from the appropriate slot / token using a PKCS11 family of functions from the libp11 library. ENGINE family of functions can be used to load pkcs11 engine in the OpenSSL.

Other Platforms and Operating Systems

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

E-SDC to Backend.Api

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

Once valid Test certificate(s) are obtained you can access API description on the following URL: https://api.staging.vms.frcs.org.fj/Swagger (for Staging) or https://api.vms.frcs.org.fj/Swagger (for Production).

How to Obtain a URL of the Backend.Api Service in Runtime


Backend.Api URL is stored in a Digital certificate as value of OID. OID is dynamically created during Smart Card personalization and depends on a target environment. Test and Production environments will have different OID so that smart cards issued for one environment cannot be used for any purpose on another environment.

In order to use the same E-SDC with the Test and Production environment, the correct OID shall be constructed using the following procedure.

1. Get Certificate using Export Certificate APDU command
2. Read Value of EnhancedKeyUsage (for example, 1.3.6.1.4.1.49952.5.2.3.3)
3. Fourth and Third integer to the right identify the environment
4. Construct OID that contain a URL of the Backend.Api by replacing stars with numbers using the following pattern - 1.3.6.1.4.1.49952.*.*.5
5. For this example, resulting OID will be 1.3.6.1.4.1.49952.5.2.5
6. Read Value of resulting OID containing root URL of Backend.Api REST Service

Authenticate

Communication between an E-SDC and Backend.Api is carried out via HTTPS protocol. The E-SDC is authenticated by Backend.Api using a certificate stored on PKI Applet or an authentication token received from Backend.Api, once a client certificate authentication has been successfully conducted as the first step.

Request Authentication Token

When requesting the authentication token, an E-SDC shall authenticate with a valid Digital Certificate (stored in the PKI applet). If the token is successfully created it shall be returned to the E-SDC as a string. In order to receive an authentication token, an E-SCD shall establish a secure connection to “/api/SDC/RequestAuthenticationToken” operation on Backend.Api.

A request is composed as follows:

1. Create HTTPS POST 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 request to “/api/SDC/RequestAuthenticationToken” operation on Backend.Api web service.

Response from Backend.Api will contain a JSON formatted text with the authentication token string object that shall be used for further communication.

The certificate based authentication is used only to request a token. Request for the authentication token shall be periodically invoked to obtain new token and to verify date and time.
The following image is an example of token request and response from the staging environment.

The Token is valid for 8 hours by default.

An E-SDC shall use the current token when calling all other services exposed by Backend.Api.

When a token expires an E-SDC shall request a new token.

If an E-SDC requests a new token while current token is still valid, TaxCore shall return the existing token.

While creating HTTP request, an E-SDC shall put the token in HTTP request header. Header key is `TaxCoreAuthenticationToken`, and value is a valid token string.

**Get Initialization Commands**

For each new smart card issued by Tax Service, 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 Backend.Api operation Notify Online Status (typically by E-SDC)
- By invoking Backend.Api operation Submit Audit Package (typically by E-SDC)
- By invoking Backend.Api operation Get Initialization Commands (typically by E-SDC)
- By using TaxPayer web portal

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

E-SDC can explicitly require initialization commands, by invoking Backend.Api operation Get Initialization Commands.

- Initialization commands include:
  - Configure Time Server URL Command,
  - Set Tax Rates Command,
  - Update Verification URL Command.

To get Initialization commands compose HTTPS POST request as follows:

1. Add headers "Accept: application/json", "Content-Type: application/json" and header that contains an authentication token
2. Submit POST request to https://<backend_api_url>/Api/SDC/GetInitializationCommands

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

**Notify Online Status**

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

Compose HTTPS POST request as follows:

1. Add headers "Accept: application/json", "Content-Type: application/json" and header that contains authentication token
2. Add a POST field “true” to the request
3. Submit POST request to https://<backend_api_url>/api/SDC/NotifyOnlineStatus

After the request is sent, Backend.Api shall return 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.

**Notify Command Processed**

After an E-SDC process commands received from Backend.Api, it shall report the results of execution to Backend.Api.

Format of request is described in the section Commands Results.

1. Add headers "Accept: application/json", "Content-Type: application/json" and header that contains authentication token
2. Submit POST request to https://<backend_api_url>/api/SDC/NotifyCommandProcessed

**Submit Audit Package**

After the invoice audit package is created (explained in section Creating an Audit Package), it shall be transferred to Backend.Api the next time 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://<backend_api_url>/Api/SDC/SubmitAuditData

After the request is sent, Backend.Api shall response with a JSON formatted text containing a status of operation and a list of commands an E-SDC shall execute.

**Model**

AuditDataStatus {
    Status (integer, optional) = ['0', '1', '2', '3', '4', '5', '6']
    Commands (Array[Command], optional),
    OperationSuccess (boolean, optional)
}

Command {
    CommandId (string, optional),
    Type (integer, optional) = ['0', '1', '2', '3', '4', '5']
    Payload (string, optional),
    UID (string, optional),
    Recipient (string, optional)
}

**Data Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>Audit Package unpacked and verified by Backend API. Status may help E-SDC developers rectify problems with audit packages</td>
</tr>
<tr>
<td></td>
<td>If all verifications are successfully Status should have value 4 - Invoice is verified</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Invoice is not yet verified</td>
</tr>
<tr>
<td>1</td>
<td>Signature is valid</td>
</tr>
<tr>
<td>2</td>
<td>Signature is invalid</td>
</tr>
<tr>
<td>3</td>
<td>Internal data is invalid</td>
</tr>
<tr>
<td>4</td>
<td>Invoice is verified</td>
</tr>
<tr>
<td>5</td>
<td>Verification data is not complete</td>
</tr>
<tr>
<td>6</td>
<td>Signed with revoked certificate</td>
</tr>
</tbody>
</table>

**Commands**

Contains list of commands that should be executed by E-SDC, as described in Commands section.

**OperationSuccess**

If the result field OperationSuccess is true you can mark that audit package file for deletion, from the E-SDC, once PoA is successfully completed.

**Submit Audit Request Payload (ARP)**

E-SDC invokes BeginAudit command and receives 256bytes of data that represents Audit Request Payload (ARP). ARP has to be converted to string using Base64 encoding and submitted to endpoint https://<backend_api_url>/api/SDC/StartProofOfAudit as body of 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. Add a JSON message to the body of the HTTP POST request
3. Submit POST Request to https://<backend_api_url>/Api/SDC/StartProofOfAudit

**Data Format**

```json
{
   "AuditRequestPayload": "string"
}
```

**File-Based Communication**

**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

**Tax Inspector Configures a new E-SDC using an SD Card**

JSON file (*.commands) with commands shall be stored in a subfolder named after UID of the card inserted in the E-SDC. e.g D:\YJ37C9Z9\YJ37C9Z9.commands

**JSON File Format**

```json
{
   "Commands": [
   {
       "CommandId": "GUID",
       "Type": 0,
       "Payload": "Command Specific Json as string",
       "UID": "string"
   }
   ]
}
```

**E-SDC Executes Commands Received via SD Card/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. Command types and structure are explained in the section Commands.

JSON file with commands shall be stored in the subfolder named after UID value.

E-SDC shall execute only those command with the same UID as UID assigned to the digital certificate of the Secure Element (stored in the SerialNumber field of the certificate subject).

**E-SDC Stores a command execution result to the SD Card/USB drive**

After commands have been executed, E-SDC shall store JSON file (<UID>.results) with the result to the UID folder on the SD Card/USB drive, e.g. G:\BJ3PN1S9\ where G is the root of the SD card/USB drive and <UID> is UID of the smart card in use. Existing file on the SD Card/USB drive shall be overwritten.

The format of the data is explained in section Commands
E-SDC Stores Audit Files on SD Card/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.

All files shall be stored in the subfolder titled Audit of the UID folder.

Example: G:\BJ3PN1S9\Audit\ where G is the root of the SD card/USB drive.

If the folder does not exist, an E-SDC shall create a new one.

An Audit consists of two file types:

1. One ARP.bin file containing the result of the invocation of Begin Audit APDU command (256Bytes)
2. One or many audit package files named using the following convention: {UID}-. {OrdinalNumber}.json