**Задача: Разработать архитектуру и ключевые модули для интеграции Odoo 18 POS с армянскими фискальными регистраторами (HDM), поддерживающую работу с несколькими кассами.**

**1. Анализ требований и архитектура:**

* **Несколько касс (POS):** Каждая касса в Odoo (pos.config) должна быть привязана к своему физическому фискальному регистратору (HDM). Это означает, что в настройках каждой кассы должны быть указаны свои HDM IP Address, Port, HDM Password.
* **Драйвер фискального регистратора:** Необходимо создать отдельный Python-модуль (например, l10n\_am\_edm или pos\_hdm), который будет инкапсулировать всю логику обмена по протоколу из документации.
* **Менеджер соединений:** Драйвер должен управлять пулом TCP-соединений к разным HDM-устройствам, так как кассы могут работать параллельно.
* **Интеграция с POS Flow:** Драйвер должен предоставлять сервис, который будет вызываться из JavaScript-кода фронтенда Odoo POS при различных событиях (закрытие заказа, возврат, внесение/изъятие денег и т.д.).

**2. Модели данных (Odoo Models):**

* pos.config**(Настройки кассы):** Добавить новые поля для хранения параметров HDM.
  + hdm\_enabled (Boolean): Включена ли интеграция с HDM для этой кассы.
  + hdm\_ip (Char): IP-адрес HDM устройства.
  + hdm\_port (Integer): Порт HDM устройства.
  + hdm\_password (Char): Пароль для подключения к HDM.
  + hdm\_print\_receipt (Boolean): "Print receipt with HDM printer". Если False, Odoo должен распечатать фискальный чек на своем принтере, используя данные из ответа HDM.
* pos.order**(Заказ):** Добавить поля для хранения фискальных данных, полученных от HDM.
  + hdm\_fiscal\_number (Char): Фискальный номер чека.
  + hdm\_rseq (Integer): Последовательный номер чека (Receipt sequence number).
  + hdm\_verification\_number (Char): Номер для проверки чека.
  + hdm\_qr\_code (Char): QR-код чека.
  + hdm\_crn (Char): Регистрационный номер (RN) HDM, который напечатал чек. Важно для возвратов.

**3. Бэкенд (Python Сервис):**

Создать класс HDMService, который реализует:

* Установку и поддержку TCP-соединения с HDM (для каждой кассы свое).
* Формирование бинарного заголовка запроса согласно п. 4.4.1.
* Кодирование/декодирование тела запроса/ответа с помощью 3DES (два ключа, как описано в п. 4.4.3).
* Обработку последовательных номеров (sequence number) для защиты от replay-атак (п. 4.4.5).
* Реализацию всех JSON-запросов и парсинг ответов, описанных в п. 4.5.
* Обработку ошибок (п. 4.10) и их трансляцию в понятный для Odoo и пользователя вид.

**Примерная логика ключевого метода**print\_receipt**в сервисе:**

python

def hdm\_print\_receipt(self, pos\_config\_id, order\_data, sequence\_number):

# 1. Получить настройки кассы (pos\_config\_id) для IP, порта, пароля

# 2. Получить сессионный ключ для этой кассы (если нет -> выполнить Login)

# 3. Сформировать JSON-запрос согласно п. 4.5.4 на основе order\_data

# 4. Закодировать тело запроса сессионным ключом

# 5. Сформировать бинарный заголовок

# 6. Отправить данные по TCP

# 7. Получить и раскодировать ответ

# 8. Распарсить JSON-ответ

# 9. Если успешно (код 200) -> сохранить фискальные данные в order\_data

# 10. Если ошибка -> сбросить соединение, выбросить исключение с описанием ошибки

# 11. Вернуть результат (фискальные данные) вызывающему коду

**4. Фронтенд (Odoo POS JavaScript):**

Модифицировать стандартное поведение POS.

* **При закрытии заказа:** После того как пользователь выбирает способ оплаты, но до фиксации заказа в базе, необходимо вызвать метод бэкенда print\_receipt. Если вызов успешен, сохранить полученные фискальные данные вместе с заказом. Если произошла ошибка, показать ее пользователю и отменить закрытие заказа.
* **Печать чека:** Если в настройках кассы hdm\_print\_receipt = False, необходимо сформировать фискальный чек на принтере Odoo, используя данные (hdm\_fiscal\_number, hdm\_qr\_code и т.д.), полученные от HDM.
* **Возвраты:** Реализовать вызов функции print\_return\_receipt (п. 4.5.6). Важно передать crn и receiptID оригинального чека.
* **Внесение/изъятие денег:** Вызывать cash\_in\_out (п. 4.5.8).

**5. Дополнительные функции:**

* **Настройка заголовков и логотипа:** Реализовать UI в настройках кассы для управления заголовками и загрузки логотипа, которые будут отправляться на HDM (пп. 4.6.3, 4.6.4).
* **Отчеты (X/Z):** Добавить кнопки в интерфейс управления кассой для запуска отчетов на HDM (п. 4.6.2).
* **Синхронизация:** Реализовать фоновую задачу или ручную кнопку для синхронизации времени между сервером Odoo и HDM (п. 4.7).
* **Валидация eMark:** Реализовать вызов функции проверки кода check\_emark (п. 4.9), если такая функциональность требуется.

**Ключевые технические вызовы:**

1. **Работа с TCP-сокетами:** Необходимо использовать библиотеку socket в Python. Важно обеспечить устойчивость соединения, таймауты и реконнекты.
2. **3DES Шифрование:** Использовать библиотеку pycryptodome или аналогичную для реализации шифрования ECB mode with PKCS7 padding.
3. **Управление состоянием:** Нужно хранить сессионные ключи и последние sequence number для каждой активной кассовой сессии. Это можно делать в кеше (например, redis) или в транзиентной модели Odoo.
4. **Блокирующие вызовы:** Вызовы к HDM являются блокирующими и медленными (сетевая коммуникация + время печати). Необходимо использовать асинхронные задачи (через queue.job или rpc) или tread pooling, чтобы не блокировать интерфейс пользователя и работу других касс.

**Заключение:**

Задача комплексная, требует глубокого понимания как протокола HDM, так и архитектуры Odoo, особенно модуля POS. Начать следует с создания изолированного Python-драйвера для HDM, протестировать его на всех функциях, а затем уже интегрировать его в Odoo через отдельный модуль, обеспечив корректную работу в многопользовательском режиме с несколькими кассами.

**Technical Task: Integration of Odoo 18 POS with Armenian HDM Fiscal Registers (Multi-POS Support)**

**Version:** 1.0  
**Date:** 2024-03-19  
**Status:** Draft

1. Overview

This document outlines the requirements for developing an Odoo 18 module to integrate multiple Point of Sale (POS) terminals with Armenian Fiscal Register Machines (HDM - ՀԴՄ) according to the official protocol specification (v0.7.3). The primary goal is to ensure all sales, returns, and cash operations are fiscalized correctly on respective HDM devices, with support for concurrent operation across different POS terminals.

2. Objectives

* Enable each Odoo POS terminal to communicate with a dedicated HDM device via TCP/IP.
* Automatically fiscalize all sales orders, returns, cash-in, and cash-out operations.
* Store fiscal data (fiscal number, verification number, QR code, etc.) within Odoo for record-keeping and reprinting.
* Provide robust error handling and user notifications for HDM communication failures.
* Allow configuration of HDM parameters per POS terminal.

3. System Architecture & Design

3.1. Data Models (Odoo Models)

* **Extension of**pos.config**(POS Terminal Configuration):**
  + Add new fields to store HDM connection parameters for each POS.
    - hdm\_enabled (Boolean): Enable/disable HDM integration for this terminal.
    - hdm\_ip (Char): IP Address of the HDM device.
    - hdm\_port (Integer): Port number of the HDM service.
    - hdm\_password (Char): Password for HDM authentication.
    - hdm\_print\_locally (Boolean): If True, the HDM prints the receipt. If False, Odoo must print the fiscal data on its receipt.
* **Extension of**pos.order**(Sales Orders):**
  + Add fields to persist fiscal data received from HDM responses.
    - hdm\_fiscal\_number (Char)
    - hdm\_rseq (Integer): Receipt Sequence Number.
    - hdm\_verification\_number (Char)
    - hdm\_qr\_code (Char)
    - hdm\_crn (Char): The Registration Number (RN) of the HDM that fiscalized the order. Critical for returns.

3.2. Backend Service (HDMFiscalDriver - Python)

A core Python service class will be developed to handle all protocol-specific communication.

* **Responsibilities:**
  + Manage TCP socket connections (establish, maintain, close) to HDM devices.
  + Generate the 12-byte request header as per section 4.4.1.
  + Implement 3DES encryption/decryption (ECB mode, PKCS7 padding) using the two-key system (first key for login, session key for subsequent calls) described in 4.4.3.
  + Manage and validate sequence numbers (seq) for each session to prevent replay attacks (4.4.5).
  + Construct JSON request bodies and parse JSON responses for all HDM functions.
  + Handle errors (section 4.10) and translate them into meaningful messages for the frontend.
* **Key Methods (mapping to protocol functions 4.5):**
  + \_send\_request(self, pos\_config, function\_code, payload\_dict): Generic method to handle header creation, encryption, transmission, and response parsing.
  + get\_operators\_and\_deps(pos\_config): (Func. 1)
  + operator\_login(pos\_config, cashier\_id, pin): (Func. 2) Returns and stores the session key for the POS.
  + operator\_logout(pos\_config): (Func. 3)
  + print\_receipt(pos\_config, order\_data, sequence\_number): (Func. 4) The main method for fiscalizing sales.
  + print\_return\_receipt(pos\_config, original\_order, sequence\_number): (Func. 6)
  + cash\_in\_out(pos\_config, amount, is\_cashin, description, sequence\_number): (Func. 11)
  + get\_receipt\_info(pos\_config, ...): (Func. 10)
  + get\_datetime(pos\_config): (Func. 12)
  + print\_report(pos\_config, report\_type, ...): (Func. 9)
  + sync\_device(pos\_config): (Func. 14)

3.3. Frontend Integration (POS JavaScript)

The Odoo POS interface will be modified to intercept key actions and call the backend service.

* **Order Finalization:**
  1. After payment is selected and before the order is finally submitted, the frontend will call the backend print\_receipt method via RPC.
  2. If successful, the fiscal data from the response will be added to the order payload and saved with the order.
  3. If hdm\_print\_locally is False, the standard receipt printing process will be modified to include the HDM fiscal data (Fiscal No:, Verification No:, QR Code).
  4. If the HDM call fails, an error message will be shown, and the order finalization will be aborted.
* **Returns:**
  1. The process will look up the original order's hdm\_crn and hdm\_rseq to build the correct request for print\_return\_receipt.
* **Cash Operations:**
  1. The "Put Money In" / "Take Money Out" functions will trigger the cash\_in\_out method.

4. Key Features & Workflows

4.1. Initial Setup & Configuration

1. Administrator configures each POS terminal (Point of Sale -> Configuration -> Settings) with its respective HDM IP, Port, and Password.
2. The "HDM Print" option is set based on whether the physical HDM printer or an Odoo-controlled printer should produce the fiscal receipt.

4.2. Daily Operation (Cashier)

1. A cashier logs into the Odoo POS using their PIN.
2. *(Optional but Recommended)* The system automatically calls operator\_login in the background upon POS startup to establish a session key for the terminal.
3. The cashier makes sales as usual. All fiscalization happens automatically and transparently in the background when closing an order.
4. For returns, the cashier selects the original order. The system uses the stored fiscal data to execute a correct fiscal return on the HDM.
5. At end of day, a Z Report can be triggered from the Odoo POS interface, which calls the corresponding HDM report function (print\_report).

5. Protocol Implementation Details

* **Encoding:** All string data must use UTF-8 encoding. The system must be designed to handle Armenian, Russian, and English characters for product names.
* **Numeric Precision:** Monetary values must be rounded to 2 decimal places. Quantities must support up to 3 decimal places. Rounding must be performed as specified: 0.005=0.01; 0.004=0.00.
* **Encryption:** The pycryptodome library will be used for 3DES encryption. The first key is derived from the SHA-256 hash of the HDM password. The second key (session key) is received from the operator\_login response.
* **Sequence Numbers:** A persistent sequence counter must be maintained for each active POS session and incremented with every request.

6. Error Handling & Resilience

* The system must handle common errors like network timeouts, printer paper jams (Error 145), and invalid operator logins (Error 111-113).
* Errors must be clearly displayed to the cashier in the POS UI, advising on corrective action (e.g., "Check HDM Printer," "Incorrect PIN").
* Under no circumstances should an order be saved in Odoo as paid if the HDM fiscalization fails. The transaction must be rolled back, and the cashier must be prompted to try again.

7. Non-Functional Requirements

* **Performance:** Communication with the HDM is a blocking operation. To avoid freezing the POS interface, all calls to the HDMFiscalDriver must be handled asynchronously (e.g., using Odoo's queue.job framework or similar threading approach).
* **Security:** HDM passwords and session keys must be stored and handled securely.
* **Logging:** Detailed logs of all requests and responses to/from the HDM must be maintained for debugging and audit purposes.

8. Dependencies

* Odoo 18.0
* Python library: pycryptodome (for 3DES encryption)

9. Acceptance Criteria

* A sale made on POS Terminal A is fiscalized on HDM Device A and stored in Odoo with the correct fiscal data.
* A sale made simultaneously on POS Terminal B does not interfere with Terminal A and is fiscalized on HDM Device B.
* A return for an order created on Terminal A is correctly processed as a fiscal return on HDM Device A.
* A cash-in operation is recorded on the respective HDM.
* If the HDM is offline, the sale is not completed, and the user receives an error message.
* Receipts printed by Odoo (when hdm\_print\_locally is False) contain all mandatory HDM fiscal information.
* All monetary calculations adhere to the rounding rules specified in the protocol.