Специфікація використання

API COM Cashälot

v. 4.5

Зміст

Поняття і визначення

КЕП – кваліфікований електронний підпис.
Локальний номер чека – номер, призначений чеку системою виписки електронних касових чеків, що використовується продавцем.
Офлайн сесія – сукупність документів, створених в режимі офлайн, між припиненням і відновленням зв'язку ПРРО з ФСКО.
ПРРО – програмний реєстратор розрахункових операцій. Система виписки електронних касових чеків, що використовується продавцем.
Фіскальний номер чека(ФН) – номер, призначений чеку сервером ФСКО.
ФСКО – фіскальний сервер контролюючого органу.

Загальна інформація

API COM Cashälot використовується для інтеграції ПРРО з розрахунково-обліковими системами. В даному документі приведено специфікацію методів бібліотек Cashälot для здійснення підключення з обліковими системами.

Інтеграція надає користувачам можливість, безпосередньо з розрахунково-облікової системи, скористатися наступним функціоналом програмного забезпечення Cashälot:

- Відкриття та закриття касової зміни;
- Створення та реєстрація чеків продажу/повернення;
- Створення та реєстрація чеків з типами оплати передоплата, післяплата, інтернет-продажа;
- Створення та реєстрація чеків ломбардних операцій;
- Створення видаткового чека на основі чека продажу та його реєстрація;
- Формування та друк Z-звіту за даними фіскального сервера контролюючого органу (скорочено ФСКО)/X-звіту за даними ПРРО;
- Створення та реєстрація службових чеків;
- Службове внесення;
- Службова видача;
- Робота в режимі офлайн;
- Можливість відображення на екрані зареєстрованого чека на пристрої продавця з QR-кодом після успішної реєстрації;
- Друк зареєстрованого чека;
- Формування періодичних звітів.

Апаратні та програмні вимоги

Для оптимальної роботи ПРРО Cashälot рекомендовані наступні системні та програмні вимоги:

Операційна система

Програмний комплекс коректно функціонує на комп'ютерах з сучасними операційними системами Microsoft та компонентами, що входять до її складу. Рекомендовано використовувати останню версією ОС, наприклад, Windows 10 або Windows Server 2019 відповідно до потреб, з встановленими актуальними оновленнями.

Апаратне забезпечення:

- процесор з мінімальною тактовою частотою від 2 ГГц;
- оперативна пам'ять від 2 GB;
- вільне місце на жорсткому диску від 1,5 GB;
- кольоровий графічний дисплей;
- маніпулятор типу миша та клавіатура;
- встановлений український або російський мовний стандарт регіональних налаштувань операційної системи;
- підтримка кирилиці в операційній системі;
- доступ до поштового сервера;
- наявний зв'язок з фіскальним сервером контролюючого органу.

Мережа

Для роботи користувача необхідний доступ до мережі інтернет. Також, необхідно мати доступ до кабінету Cashälot, фіскального сервера контролюючого органу та до ресурсу центру сертифікації ключів. Наприклад, якщо Ви використовуєте ключі від "АЦСК Україна", то необхідно надати доступ до ресурсу: https://uakey.com.ua/ - 212.90.186.150 - 80/443. Якщо ж ви використовуєте КЕП (ЕЦП) інших кваліфікованих надавачів (АЦСК), то необхідно надати доступ до серверів TSP, OCSP, CMP відповідних центрів.

Загальний список ресурсів:

- https://load.cashalot.org.ua/update/ - 94.131.247.22:443 - Сервіс завантаження інсталяцій програми (дистрибутиви, оновлення);
- http://fs.tax.gov.ua:8609/fs/cmd - Сервіс API ДПС;
- https://my.cashalot.org.ua/ - 94.131.247.21:443 - Особистий кабінет Cashälot.

API COM ПРРО Cashälot

API COM ПРРО Cashalot призначений для підключення до сторонніх розрахунково - облікових систем з використанням COM-методів **(COM CashalotApi).**

Використовуються бібліотеки **CashalotApi32.dll** та **CashalotApi64.dll** , що розміщені в кореневому каталозі ПРРО Cashalot. В даній специфікації приклади написані під VBS.

Перед початком роботи з **CashalotApi** потрібно:

- зареєструватись в особистому кабінеті https://my.cashalot.org.ua/;
- виконати всі пункти з розділу “Початок роботи” в особистому кабінеті;
- встановити ПРРО **Cashalot**, попередньо завантаживши дистрибутив з офіційного сайту. Виконати запуск програми Cashalot(В момент першого запуску завантажуються дані з бек-офісу). Для паралельної роботи в декількох касах одночасно, необхідно встановлювати ПРРО **Cashalot** для кожної каси окремо;
- зареєструвати бібліотеки в ОС за допомогою команди **regsvr32**

regsvr32 “<Повний шлях до ПРРО Cashalot>\CashalotApi32.dll”

(для 32-бітної платформи)

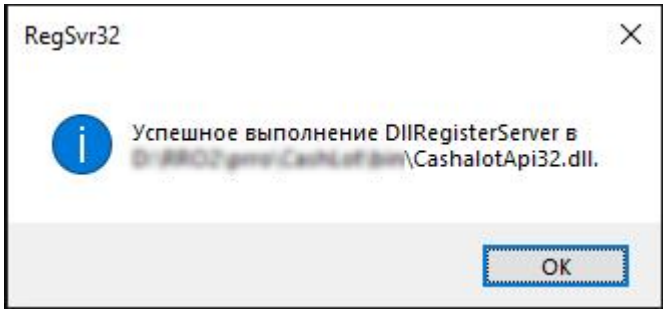
regsvr32 “<Повний шлях до ПРРО Cashalot>\CashalotApi64.dll”

(для 64-бітної платформи)

Наприклад:

regsvr32 “C:\Cashalot\CashalotApi32.dll”

Після успішної реєстрації бібліотеки з’явиться повідомлення зразку:



В бібліотеках CashalotApi реалізована обробка наступних методів:

- ***SetParameter***
- ***SetParameterEx***
- ***GetVersion***
- ***GetInterfaceRevision***
- ***ServiceInput***
- ***ServiceInputEx***
- ***ServiceOutput***
- ***ServiceOutputEx***
- ***OpenShift***
- ***FiscalizeCheck***
- ***FiscalizeReturnCheck***
- ***GetReceiptLocalNumberByFiscalNumber***
- ***GetReceiptFiscalNumberByLocalNumber***
- ***GetCurrentStatus***
- ***SyncBalanceOnGoods***
- ***SyncGoodsWithBalance***
- ***GetReceiptXML***
- ***GetFirstReceiptByPeriod***
- ***GetFirstReceiptByShift***
- ***GetNextReceipt***
- ***GetFirstZReportByPeriod***
- ***GetNextZReport***
- ***GetZReportByShift***
- ***ShowReceipt***
- ***GetLastError***
- ***PrintXReport***
- ***GetXReport***
- ***GetLastZReport***
- ***PrintZReport***
- ***CloseShift***
- ***PrintPReportDate***
- ***PrintTextDocument***

Методи повертають відповідь у вигляді структури **CashalotApiRetVal**, в залежності від виконаного методу, можуть міститись такі дані:

Властивість	Тип	Значення
Return	bool	Ознака успішності виконання.
Description	string	Опис результату виконання або помилки. Заповнюється у випадку коли: Return = False.
JsonVal	string	JSON-структура .
ReceiptFiscalNum	string	Фіскальний номер чека.
ReceiptLocalNum	string	Локальний номер чека.
ShiftID	string	Ідентифікатор касової зміни.
XML	string	XML-структура документу.
Type	int	Тип документа.

---

1. Встановлення налаштувань SetParameter (Name, Value)

Функція призначена для встановлення значень параметрів налаштувань API бібліотеки. Рекомендовано встановити всі параметри на початку роботи.
Кожен параметр повинен містити в собі:

Ім'я	Тип	Опис
Name	string	Ім'я параметра
Value	string	Значення параметра

Приклад:

```
App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
```

SetParameter складається з обов'язкових параметрів (виділені червоним) та не обов'язкових параметрів. Параметри повинні мати послідовність, як в таблиці опису параметрів. Тип всіх параметрів string, для використання параметрів з різними мовами програмування. Опис параметрів вказано в даній таблиці:

Ім'я параметра(Name)	Значення параметра(Value)
PathToCashalotDir	Шлях до каталогу з встановленим ПРРО Cashälot.
DeviceIDFnRRO	Фіскальний номер ПРРО.
NOAUTOUPDATE	Перевірка на наявність нової версії на сервері https://load.cashalot.org.ua/update/ та запит на оновлення програми Cashälot в момент початку роботи з API: "True" – заборонити оновлення; "False" – дозволити оновлення(за замовчуванням).
NOINTERFACEMODE	Ознака використання інтерфейсу РМК для авторизації та службових операцій видачі та внесення готівки в касу. "False" – при роботі з API викликається діалогове вікно. "True" – авторизація відбувається шляхом передачі параметрів авторизації PathToCertificate та PwdToCertificate. Вікна службових операцій видачі та внесення готівки не викликаються (проте, результат операції службового внесення/видачі готівки відображається).

PathToCertificate	Шлях до каталогу з файловим ключем користувача. Важливо: в папці повинні міститися ключ і сертифікат тільки одного користувача. Якщо касою користуються декілька касирів, необхідно помістити їх сертифікати і ключі в окремі каталоги.
PwdToCertificate	Пароль від ключа.
USETOKEN	Чи використовується токен для авторизації. "True" – використовується. "False" – не використовується (використовується файловий ключ). Важливо: параметр USETOKEN має бути заданий після параметрів PathToCashalotDir, DeviceIDFnRRO та NOINTERFACEMODE. Для пришвидшення завантаження програми рекомендовано не використовувати токени.
USECLOUDKEY	Чи використовується CloudKey для авторизації. "True" – використовується. "False" – не використовується. Параметр USECLOUDKEY має бути заданий після параметрів PathToCashalotDir, DeviceIDFnRRO та NOINTERFACEMODE.
NOAUTOOPENSHIFT	Перевірка стану зміни (відкрита або закрита). Якщо зміна закрита, при виклику будь-якої COM-функції зміна автоматично відкривається. "False" – автоматично відкривати зміну, якщо вона закрита. "True"– не відкривати автоматично зміну. В цьому випадку робота COM-функції буде перервана і результат виконання буде повернуто у вигляді помилки.
DEFAULTPRINTERNAME	Визначає принтер для друку при натисканні кнопки "Друк" в діалогових вікнах: 1) "" – пустий рядок, виводиться діалог вибору принтера; 2) "DEFAULTWINDOWS" – використовувати принтер Windows за замовчуванням; 3) "ім'я принтера чеків" - в параметр передається ім'я принтера чеків, який буде використовуватись для друку чеків в ПРРО Cashälot.
AUTOPRINTMODE	Параметр визначає автоматичну відправку на друк чека або службового звіту без відображення інформаційного вікна: "True" – відправляти на друк; "False" – не відправляти на друк. Якщо задано принтер за замовчуванням (DEFAULTPRINTERNAME), чек друкується одразу, в іншому випадку відображається вікно вибору принтера для друку.
NOPRINTMODE	Дозвіл на друк: "True" – заборонити друк; "False" – дозволити друк.

SHOWREPORTADDITIONALINFO	Виводити додаткову інформацію в нижній частині Z-звіті. В разі встановленого значення "True", відображаються такі поля: - Залишок готівки в касі; - Сума знижки; - Сума націнки. "True" - відображає додаткову інформацію; "False" – не відображає додаткову інформацію.
SPECIFICAPIMODE	Режим передбачений для інтеграції з обліковими системами, які розроблені з використанням мов програмування Pascal, Delphi та Xbase++. При використанні SPECIFICAPIMODE, в залежності від значення параметру, можливо використовувати режим без інтерфейсу та з інтерфейсом. **Зверніть увагу!** Встановлення параметра SPECIFICAPIMODE та його значень **обов'язково** повинно бути виконано перед встановленням інших параметрів. **Встановлення SPECIFICAPIMODE необхідно здійснювати один раз, перед запуском роботи з ПРРО. Зміна значень у процесі роботи не рекомендована.** Параметр SPECIFICAPIMODE може приймати наступні значення: "0" - програма працює в стандартному режимі; "1" - переводить API COM Cashälot автоматично в режим роботи без інтерфейсу, який підтримує одночасну роботу тільки з одним екземпляром ПРРО. Встановлення параметрів підтримується тільки за допомогою **SetParameter**, "2" - переводить API COM Cashälot в оптимізований режим роботи з інтеграціями, який підтримує можливість одночасної роботи з декількома ПРРО та може працювати як в режимі без інтерфейсу, так і з увімкненим інтерфейсом. Встановлення параметрів для роботи з двома та більше ПРРО здійснюється за допомогою SetParameterEx. При необхідності використання тільки одного екземпляру ПРРО, параметри можливо встановити як за допомогою SetParameterEx так і SetParameter. Також, робота даного режиму забезпечується процесом CashalotReporter.exe, який буде автоматично запущено при виконанні операцій з ПРРО.
USEGOODSSTORAGEMODE	Параметр USEGOODSSTORAGEMODE визначає використання ведення складського обліку по товарам та автоматичну синхронізацію залишків по ним при використанні API COM Cashälot за умови активованого налаштування в кабінеті користувача в розділі «Склад - Налаштування - Облік товарів у ПРРО». Параметр може приймати наступні значення:

	«0» - програма працює в стандартному режимі без ведення обліку товарів в API та синхронізації залишків по товарам з кабінетом користувача. «1» - режим передбачає використання автоматичної синхронізації залишків по товарам та інформування користувача окремим повідомленням про перевищення їх ліміту під час спроби реєстрації фіскального чека. Після появи повідомлення користувачу доступна можливість відмовитись від подальшої обробки чека або погодитись та продовжити реєстрацію, але таким чином залишки по товарам будуть мати від'ємне значення кількості. «2» - режим передбачає використання автоматичної синхронізації залишків по товарам та інформування користувача окремим повідомленням про перевищення їх ліміту без можливості продовжити обробку та реєстрацію чека. Візуалізація повідомлення про перевищення ліміту залишків з можливістю продовжити/перервати процедуру фіскалізації чека, використовуючи перший режим параметра USEGOODSSTORAGEMODE, буде виникати також при ознаці використання режиму роботи без інтерфейса (NOINTERFACEMODE=true). *Кроки по практичному застосуванню параметра USEGOODSSTORAGEMODE 1. Імпортувати номенклатуру, яка використовується в API COM Cashälot, до кабінету користувача. Після імпорту необхідно перенести номенклатуру до еталонної групи. 2. В кабінеті користувача в модулі «Склад» сформувати та провести документ «Надходження» по необхідним товарним позиціям. 3. Налаштувати та встановити використання параметра USEGOODSSTORAGEMODE у необхідному режимі. 4. Виконати синхронізацію даних по товарам (номенклатурі) та залишкам товарів методом SyncGoodsWithBalance. Подальше оновлення інформації по залишкам відбувається в автоматичному режимі при відкриті зміни та реєстрації фіскального чека, а також в ручному режимі, з використанням методів синхронізації товарів та залишків по ним або окремо, виконавши синхронізацію залишків. 5. Сформувати та заповнити фіскальний чек даними по номенклатурі, для якої доступні залишки по товарам. **Зверніть увагу!** Коректна робота обраного параметра відбувається лише за умови, що в API COM Cashälot використовується одна й та сама номенклатура, що заведена в кабінеті користувача.

USEPRGPOSTERMINAL	Параметр відповідає за можливість проведення оплати платіжною карткою через POS-термінал, який підключений в Cashalot в розділі «Налаштування - Додаткове обладнання - POS-термінал». Оплату на терміналі можливо здійснити після передачі чека на реєстрацію. В разі використання обраного параметра фіскальний чек буде містити всі необхідні обов'язкові реквізити банківського терміналу (ЕПЗ, RRN, еквайр та ін.). "True" - використовується POS-термінал, підключений в Cashalot для оплати під час передачі чека на фіскалізацію з обраною формою оплати Картка (SumPayByCard) . "False" - під час фіскалізації чека з формою оплати Картка підключений POS-термінал в Cashalot не використовується. В разі неуспішної транзакції на банківському терміналі операція реєстрації чека не відбувається.

*Приклад встановлення параметрів **SetParameter**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")

   App.SetParameter "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", "4000000000"
   App.SetParameter "NOINTERFACEMODE", "False"
   App.SetParameter "PathToCertificate", "C:\keys"
   App.SetParameter "PwdToCertificate", "111"
   App.SetParameter "USETOKEN", "False"
   App.SetParameter "NOAUTOOPENSHIFT", "False"
   App.SetParameter "DEFAULTPRINTERNAME", ""
   App.SetParameter "AUTOPRINTMODE", "False"
```

Дані параметри можна змінювати в процесі роботи з API. Наприклад, для того, щоб виконувати автоматичний друк тільки Z-звіту в процесі роботи програми.

---

2. Встановлення налаштувань API SetParameterEx (FiscalNumberRRO, Name, Value)

Якщо використовується робота одночасно з декількома ПРРО, то встановлення параметрів здійснюється виключно через **SetParameterEx**.

У такому випадку **SetParameter не використовується**.

При встановленні параметрів за допомогою **SetParameterEx** обов'язково потрібно зазначити шлях до кореневого каталогу кожного екземпляру ПРРО Cashälot та фіскальний номер каси окремо. Кожен параметр повинен містити в собі:

Ім'я	Тип	Опис
FiscalNumberRRO	string	Фіскальний номер ПРРО.
Name	string	Ім'я параметра.
Value	string	Значення параметра.

Приклад:

```
App.SetParameterEx FiscalNumberRRO1,"PathToCashalotDir", "C:\Cashalot1"
```

Опис параметрів вказано в даній таблиці:

Ім'я параметра(Name)	Значення параметра(Value)
PathToCashalotDir	Шлях до каталогу з встановленим ПРРО Cashälot.
DeviceIDFnRRO	Фіскальний номер ПРРО.
NOAUTOUPDATE	Перевірка на наявність нової версії на сервері https://load.cashalot.org.ua/update/ та запит на оновлення програми Cashälot в момент початку роботи з API: "True" – заборонити оновлення; "False" – дозволити оновлення(за замовчуванням).
NOINTERFACEMODE	Ознака використання інтерфейсу РМК для авторизації та службових операцій видачі та внесення готівки в касу. "False" – при роботі з API викликається діалогове вікно. "True" – авторизація відбувається шляхом передачі параметрів авторизації PathToCertificate та PwdToCertificate. Вікна службових операцій видачі та внесення готівки не викликаються (проте, результат операції службового внесення/видачі готівки відображається).
PathToCertificate	Шлях до каталогу з файловим ключем користувача. Важливо: в папці повинні міститися ключ і сертифікат тільки одного користувача. Якщо касою користуються декілька касирів, необхідно помістити їх сертифікати і ключі в окремі каталоги.
PwdToCertificate	Пароль від ключа.
USETOKEN	Чи використовується токен для авторизації: "True" – використовується; "False" – не використовується (використовується файловий ключ). Важливо: параметр USETOKEN має бути заданий після параметрів PathToCashalotDir, DeviceIDFnRRO та NOINTERFACEMODE. Для пришвидшення завантаження програми рекомендовано не використовувати токени.
NOAUTOOPENSHIFT	Перевірка стану зміни (відкрита або закрита). Якщо зміна закрита, при виклику будь-якої COM-функції зміна автоматично відкривається. "False" – автоматично відкривати зміну, якщо вона закрита. "True"– не відкривати автоматично зміну. В цьому випадку робота COM-функції буде перервана і результат виконання буде повернуто у вигляді помилки.
DEFAULTPRINTERNAME	Визначає принтер для друку при натисканні кнопки "Друк" в діалогових вікнах: 1) "" – пустий рядок, виводиться діалог вибору принтера; 2) "DEFAULTWINDOWS" – використовувати принтер Windows за замовчуванням; 3) "ім'я принтера чеків" - в параметр передається ім'я принтера чеків, який буде використовуватись для друку чеків в ПРРО Cashälot.

AUTOPRINTMODE	Параметр визначає автоматичну відправку на друк чеку або службового звіту без відображення інформаційного вікна: "True" – відправляти на друк; "False" – не відправляти на друк. Якщо задано принтер за замовчуванням (DEFAULTPRINTERNAME), чек друкується одразу, в іншому випадку відображається вікно вибору принтера для друку.
NOPRINTMODE	Дозвіл на друк: "True" – заборонити друк; "False" – дозволити друк.
SHOWREPORTADDITIONALINFO	Виводити додаткову інформацію в нижній частині Z-звіту. В разі встановленого значення "True", відображаються такі поля: - Залишок готівки в касі; - Сума знижки; - Сума націнки. "True" - відображає додаткову інформацію; "False" – не відображає додаткову інформацію.
SPECIFICAPIMODE	Режим передбачений для інтеграції з обліковими системами, які розроблені з використанням мов програмування Pascal, Delphi та Xbase++. **Зверніть увагу!** Встановлення параметра SPECIFICAPIMODE та його значень **обов'язково** повинно бути виконано перед встановленням інших параметрів. **Встановлення SPECIFICAPIMODE необхідно здійснювати один раз, перед запуском роботи з ПРРО. Зміна значень у процесі роботи не рекомендована.** Параметр SPECIFICAPIMODE може приймати наступні значення: "0" - програма працює в стандартному режимі; "2" - переводить API COM Cashälot в оптимізований режим роботи з інтеграціями, який підтримує можливість одночасної роботи з декількома ПРРО та може працювати як в режимі без інтерфейсу, так і з увімкненим інтерфейсом. Встановлення параметрів для роботи з двома та більше ПРРО здійснюється за допомогою SetParameterEx. При необхідності використання тільки одного екземпляру ПРРО, параметри можливо встановити як за допомогою SetParameterEx так і SetParameter. Також, робота даного режиму забезпечується процесом CashalotReporter.exe, який буде автоматично запущено при виконанні операцій з ПРРО.
USEGOODSSTORAGEMODE	Параметр USEGOODSSTORAGEMODE визначає використання ведення складського обліку по товарам та автоматичну синхронізацію залишків по ним при використанні API COM Cashälot за умови активованого налаштування в кабінеті користувача в розділі «Склад - Налаштування - Облік товарів у ПРРО». Параметр може приймати наступні значення:

	«0» - програма працює в стандартному режимі без ведення обліку товарів в API та синхронізації залишків по товарам з кабінетом користувача. «1» - режим передбачає використання автоматичної синхронізації залишків по товарам та інформування користувача окремим повідомленням про перевищення їх ліміту під час спроби реєстрації фіскального чека. Після появи повідомлення користувачу доступна можливість відмовитись від подальшої обробки чека або погодитись та продовжити реєстрацію, але таким чином залишки по товарам будуть мати від'ємне значення кількості. «2» - режим передбачає використання автоматичної синхронізації залишків по товарам та інформування користувача окремим повідомленням про перевищення їх ліміту без можливості продовжити обробку та реєстрацію чека. Візуалізація повідомлення про перевищення ліміту залишків з можливістю продовжити/перервати процедуру фіскалізації чека, використовуючи перший режим параметра USEGOODSSTORAGEMODE, буде виникати також при ознаці використання режиму роботи без інтерфейса (NOINTERFACEMODE=true). *Кроки по практичному застосуванню параметра USEGOODSSTORAGEMODE 1. Імпортувати номенклатуру, яка використовується в API COM Cashälot, до кабінету користувача. Після імпорту необхідно перенести номенклатуру до еталонної групи. 2. В кабінеті користувача в модулі «Склад» сформувати та провести документ «Надходження» по необхідним товарним позиціям. 3. Налаштувати та встановити використання параметра USEGOODSSTORAGEMODE у необхідному режимі. 4. Виконати синхронізацію даних по товарам (номенклатурі) та залишкам товарів методом SyncGoodsWithBalance. Подальше оновлення інформації по залишкам відбувається в автоматичному режимі при відкриті зміни та реєстрації фіскального чека, а також в ручному режимі, з використанням методів синхронізації товарів та залишків по ним або окремо, виконавши синхронізацію залишків. 5. Сформувати та заповнити фіскальний чек даними по номенклатурі, для якої доступні залишки по товарам. **Зверніть увагу!** Коректна робота обраного параметра відбувається лише за умови, що в API COM Cashälot використовується одна й та сама номенклатура, що заведена в кабінеті користувача.

USEPRGPOSTERMINAL	Параметр відповідає за можливість проведення оплати платіжною карткою через POS-термінал, який підключений в Cashalot в розділі «Налаштування - Додаткове обладнання - POS-термінал». Оплату на терміналі можливо здійснити після передачі чека на реєстрацію. В разі використання обраного параметра фіскальний чек буде містити всі необхідні обов'язкові реквізити банківського терміналу (ЕПЗ, RRN, еквайр та ін.). "True" - використовується POS-термінал, підключений в Cashalot для оплати під час передачі чека на фіскалізацію з обраною формою оплати Картка (SumPayByCard) . "False" - під час фіскалізації чека з формою оплати Картка підключений POS-термінал в Cashalot не використовується. В разі неуспішної транзакції на банківському терміналі операція реєстрації чека не відбувається.

Приклад використання SetParameterEx:

```
Set App = CreateObject("AddIn.CashaLotApi")
     FiscalNumberRRO1 = XXXXXXXXXX
     FiscalNumberRRO2 = YYYYYYYYYY

     App.SetParameterEx FiscalNumberRRO1,"PathToCashalotDir",  "C:\Cashalot1"
     App.SetParameterEx FiscalNumberRRO1, "DeviceIDFnRRO", FiscalNumberRRO1

     App.SetParameterEx FiscalNumberRRO2,"PathToCashalotDir", "C:\Cashalot2"
     App.SetParameterEx FiscalNumberRRO2, "DeviceIDFnRRO", FiscalNumberRRO2
```

3. Отримання номеру версії API бібліотеки GetVersion

Функція немає вхідних параметрів.

У відповіді отримаємо номер версії API бібліотеки з типом String.

Приклад:

```
Set App = CreateObject("AddIn.CashaLotApi")
     libVersion = App.GetVersion()
     MsgBox "Версія бібліотеки = " & libVersion
```

Приклад відповіді:

Версія бібліотеки = 1.2

---

4. Відкриття зміни OpenShift (FiscalNumberRRO)

OpenShift повинен містити параметр з фіскальним номером каси, який має тип string.

При виклику методу (наприклад: фіскалізації чека), якщо зміна не відкрита, відбувається автоматичне відкриття зміни з викликом функції **OpenShift**. Для запобігання автоматичного відкриття зміни, за допомогою функції **SetParameter,** встановіть параметр:

NOAUTOOPENSHIFT = "True"

В цьому випадку, поки не буде явно відкрито зміну викликом **OpenShift**, виклик інших функцій, які вимагають відкритої зміни, буде перериватися з поверненням помилки: "Операція неможлива, необхідно відкрити зміну"

При відкритті зміни, у відповіді повертається структура **CashalotApiRetVal**, що містить значення:

Ім'я	Тип	Опис
Return	bool	Ознака успішності виконання.
Description	string	Опис помилки. Заповнюється у випадку коли: Return = False.
JsonVal	string	JSON-структура, що містить значення успішності операції, помилки, ID зміни .
ShiftID	string	Ідентифікатор поточної зміни.

Приклад:

```
Set resultOpenShift = App.OpenShift (FiscalNumberRRO)
   MsgBox "Результат = " & resultOpenShift.JsonVal ' також можна додати
resultOpenShift.Description, resultOpenShift.Return, resultOpenShift.ShiftID
```

Відповідь:

```
resultOpenShift.Description = ""
resultOpenShift.JsonVal =
{"Ret":true,"ErrorString":"","Values":{"ShiftID":"1a9eb031-b1ce-4ee7-8a11-329824
d895a7"}}
resultOpenShift.Return = True
resultOpenShift.ShiftID = 1a9eb031-b1ce-4ee7-8a11-329824d895a7
```

або:

```
resultOpenShift.Description = "Авторизація відмінена користувачем"
resultOpenShift.JsonVal = {"Ret":false,"ErrorString":"Авторизація відмінена
користувачем", "Values":{}}
resultOpenShift.Return = False
resultOpenShift.ShiftID =
```

5. Фіскалізація чека продажу товарів, послуг FiscalizeCheck (FiscalNumberRRO, JSONGoodsLstData, JSONPayData)

Для створення та фіскалізації чека продажу товарів та/або послуг необхідно сформувати JSON структуру JSONGoodsLstData, що міститиме дані номенклатури, та структуру JSON структуру JSONPayData з формами оплати.

Параметри, які повинен містити метод **FiscalizeCheck**:

Ім'я	Тип	Опис
FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
JSONGoodsLstData	string	JSON структура з описом проданих товарів або послуг.
JSONPayData	string	JSON структура з підсумками оплати по чеку.

Структура **JSONGoodsLstData** містить массив **ReceiptLst**, кожен елемент якого містить параметри номенклатури та об'єкти **Comment**, **DocType**, **SaleOrderNumber**.

Опис ReceiptLst:

Поле	Значення
VendorCode*	Артикул.
Name	Найменування товару.
DocNumber**	Номер договору (при використанні ломбардних послуг).
CdUKDZED	Код УКТЗЕД.
CdDKPP	Код ДКПП.
GoodsType	Тип номенклатури. За замовчуванням товар, якщо параметр не задано. 1 – товар; 2 – послуга; 3 – ломбардна послуга **.
Barcode	Штрих-код.
UnitType*	Одиниця вимірювання.
Quantity*	Кількість товару.
Price*	Ціна одиниці товару (в форматі <Гривні, Копійки>).
Amount*	Кінцева сума по позиції чека (з урахуванням всіх націнок та знижок, у форматі <Гривні, Копійки>).
DiscountPrc	Відсоток знижки. Увага! При заповненні DiscountPrc, необхідно також обов'язково заповнювати та передавати DiscountSum
DiscountSum	Сума знижки на позицію.
VATRate	ПДВ у відсотках (від 0 до 99,99).
VATLetter	Літера ставки ПДВ
IsPriceIncludeVAT	Для випадків, коли сума по позиції не включає суму ПДВ(true/false)
SumVAT	Сума ПДВ за одиницю товару (ігнорується, якщо вказати ознаку, що враховує ПДВ “true”).
IsExcise	Ознака підакцизного товару(true/false).
ExciseLetter	Літера ставки акцизного податку
ExciseStampBarcode	Штрих-код марки акцизного податку. У випадку коли товарів більше чим 1, по одній позиції номенклатури, то всі штрих-коди заповнюються через кому. Поле ігнорується, якщо ознака підакцизного товару IsExcise = false.
OtherParametrs	Додаткові параметри у вигляді XML таблиці.

* обов'язкові параметри при передачі даних товару.
** для Ломбардних операцій.

Зауваження! Для передачі кількості товару з дробовою частиною (в т. ч. нульовою, наприклад, "2,00") допускається використовувати десятковий розділювач "." (крапка) або "," (кома). Якщо параметр не задано, або задано некоректно, його значення приймається рівним одиниці (1).

Параметр OtherParametrs передається в форматі XML з кодуванням UTF-8. Атрибути XML-рядка вказані в таблиці нижче.

Зазначимо, що параметри XML-рядка OtherParametrs мають пріоритет над параметрами, які передані в **ReceiptLst**.

Атрибути XML-рядка OtherParametrs:

Назва атрибута	Опис атрибута
Nomenclature *	Найменування товару.
UKTZED	Код УКТЗЕД.
VendorCode *	Артикул.
Barcode	Штрих-код.
Dimension *	Одиниця вимірювання.
Discount	Відсоток знижки. Увага! При заповненні Discount, необхідно також обов'язково заповнювати та передавати DiscountAmount
DiscountAmount	Сума знижки.
VAT *	Ставка ПДВ в відсотках.
VATLetter	Літера ставки ПДВ
PriceIncludesVAT або PriceIncludeVAT	Для випадків, коли сума по позиції не включає суму ПДВ(true/false)
AmountVAT	Сума ПДВ за одиницю товару, у форматі <Гривні, Копійки>.
IsExcisable	Ознака підакцизного товару, True/False.
ExciseLetter	Літера ставки акцизного податку
ExciseStampBarcode	Штрих-код марки акцизного податку. У випадку коли товарів більше чим 1, по одній позиції номенклатури, то всі штрих-коди заповнюються через кому. Поле ігнорується, якщо ознака підакцизного товару IsExcise = false.

* обов'язкові параметри в рядку OtherParametrs, якщо не вказані відповідні явні параметри.

Приклад OtherParametrs:

```
<?xml version=\"1.0\" encoding=\"UTF-8\"?> <Table> <Record VAT=\"5\" IsExcisable=\"True\" Nomenclature=\"Товар1\" UKTZED=\"0202\" VendorCode=\"1\" Barcode=\"1452\" Dimension=\"шт\" Discount=\"0.54\" DiscountAmount=\"0.25\" PriceIncludeVAT=\"1\" AmountVAT=\"0\" ExciseStampBarcode=\"11111, 22222, 33333\"/> </Table>
```

Об'єкт **Comment** використовується для додавання нефіскальної інформації:

Comment	Опціональний рядок нефіскальної інформації, коментар. Допускається використання одного коментаря на чек.

Об'єкт **DocType** є обов'язковим при проведенні ломбардних операцій, призначений для визначення типу документа (чека). Ломбардні операції.

DocType	Тип документа: 0 – чек продажу; 4 – ломбардний чек; За замовчуванням DocType = 0, якщо об'єкт не задано.

Об'єкт **SaleOrderNumber** (Номер замовлення/договору) потрібен для заповнення номера договору при проведенні інтернет продажу, передоплати та післяплати. Для відображення номера замовлення в післяплаті обов'язково потрібно заповнювати в **JSONPayData** фіскальний номер чека передоплати ParentReceiptFiscalNumber.

Заокруглення суми по чеку реалізовано наступним чином:

В **JSONPayData** передається значення готівки, що надана клієнтом на касі, значення оплати картою, сертифікатом та кредитом, а також заокруглена сума, розрахована за певним алгоритмом користувача SumPayCheck (відповідно до Постанови НБУ п.4, 1-4 коп заокруглюється до 0 коп, 5-9 коп заокруглюється до 10 коп). В чеку буде відображено заокруглення, надану готівку та інші форми оплати, заокруглену суму до сплати, решту.

Решта по чеку розраховується так:

Сума решти = SumCash + SumPayByCard + SumPayByCredit + SumPayByCertificate – SumPayCheck

Решта розраховується програмою Cashälot автоматично, згідно вказаних користувачем даних.

Структура **JSONPayData** містить підсумки оплати по чеку, і має наступні поля:

Поле	Значення
SumCash	Сума наданої готівки.
SumPayByCard	Сума оплати чека банківською картою.
SumPayByCredit	Сума оплати чека в кредит.
SumPayByCertificate	Сума оплати чека сертифікатом.
SumPayCheck	Сума чека до оплати з урахуванням всіх надбавок, знижок та округлень.
PaymentOrderType	Встановлення порядку сплати: 0 - звичайна; 1 - передоплата; 2 - післяплата; 3 - інтернет продаж.
SumPreparePayed	Сума попередньої оплати.
ParentReceiptFiscalNumber	Фіскальний номер чека, по якому здійснюється доплата/повернення.
ParentRROFiscalNumber	Фіскальний номер РРО, по якому здійснюється доплата/повернення.
CustomerEmail	Опціонально. Електронна адреса покупця, на яку можна відправити фіскалізований чек, передається в поле електронної адреси при натисканні кнопки "Надіслати чек на Email" екрану відображення чека. Рекомендовано використовувати при встановленому параметрі відображення вікон інтерфейсу: SetParameter("NOINTERFACEMODE","False").
OtherParametrs	Додаткові параметри у вигляді XML таблиці.
Необов'язкові параметри інформації по транзакції термінала при безготівкових розрахунках:	
TerminalID	Код термінала.
ApprovalCode	Код підтвердження.
RRN	РРН транзакції.
IssuerName	Платіжна система.
PAN	Номер картки.
TransactionDate	Дата транзакції (в форматі dd.mm.yyyy hh:mm:ss).
SignVerif	Електронний підпис.
AcquireName	Термінал.
InvoiceNumber	Номер чека.
ParentRRN	РРН транзакції чека, по якому здійснено повернення.

Увага! Різниця між сумою оплати чека та SumPayCheck (сумою по всіх видах оплати) вважається значенням заокруглення. Тому розробник має самостійно реалізувати алгоритм округлення значення виданої готівки або суми оплати по чеку.

Відповідь

У відповіді на **FiscalizeCheck** повертається структура **CashalotApiRetVal**, що містить:

Return	bool	Ознака успішності виконання.
JsonVal	string	JSON-структура, що містить значення успішності операції, тип документу, помилки, локальний та фіскальний номери чека, ID зміни та Base64 рядок XML чека.
ReceiptFiscalNum	string	Фіскальний номер чека.
ReceiptLocalNum	string	Локальний номер чека.
ShiftID	string	Ідентифікаційний номер поточної зміни.
Type	int	Тип документа.
XML	string	XML-структура фіскалізованого чека.

Тип документа *"Type"* визначається числом, яке формується наступним чином:

1<Клас документа><Тип документа><Додатковий тип документа>

Опис *"Type"* :

Число	Кількість знаків	Значення
<Клас документа>	1	0 - Чек 1 - Z-звіт 2 - X-звіт
<Тип документа>	3	000 - Чек реалізації товарів/послуг 001 - Чек переказу коштів 002 - Чек операції обміну валюти 003 - Чек видачі готівки 100 - Відкриття зміни 101 - Закриття зміни 102 - Початок офлайн сесії 103 - Завершення офлайн сесії
<Додатковий тип документа>	3	000 - Касовий чек (реалізація) 001 - Видатковий чек (повернення) 002 - Чек операції «службове внесення»/«отримання авансу» 003 - Чек операції «отримання підкріплення» 004 - Чек операції «службова видача»/«інкасація» 005 - Чек сторнування попереднього чека

*Приклад виконання методу **FiscalizeCheck**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
        FiscalNumberRRO = "4000000000"
        App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
        App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

        JSONGoodsLstData = "{""ReceiptLst"":[{""VendorCode"":""АртикулТовару1"",
""Name"": ""НазваТовару1"", ""CdUKDZED"": """", ""CdDKPP"": """",
""GoodsType"":"""", ""Barcode"":""123456789"", ""UnitType"":""шт"",
""Quantity"": ""1"", ""Price"": ""183"", ""Amount"":""181,17"",
""DiscountPrc"":"""", ""DiscountSum"":""1.83"",
""VATRate"":""20"",""IsPriceIncludeVAT"":true, ""SumVAT"":""30.2"",
""IsExcise"":false, ""OtherParametrs"":null}, {""VendorCode"":
""АртикулТовару2"", ""Name"":""НазваТовару2"", ""CdUKDZED"":"""",
""CdDKPP"":"""", ""GoodsType"":"""", ""Barcode"":""9999999999"",
""UnitType"":""шт"", ""Quantity"":""1"", ""Price"":""183"",
""Amount"":""181,17"", ""DiscountPrc"":"""", ""DiscountSum"":""1.83"",
""VATRate"":""20"", ""IsPriceIncludeVAT"":true, ""SumVAT"":""30.2"",
""IsExcise"": true, ""ExciseStampBarcode"":""1111"",
""OtherParametrs"":null}],""Comment"":""##################################\n\r #
Тестовий коментар \n\r##################################""}"
        JSONPayData =
"{""SumCash"":""52"",""SumPayByCard"":""310,34"",""SumPayByCredit"":null,""SumPa
yByCertificate"":null,""SumPayCheck"":""362,34"",""TerminalID "":
""S1K700J8"",""ApprovalCode"": ""68014B"",""RRN"":
""048867173620"",""IssuerName"":""MASTERCARD"",""PAN"":""XXXXXXXXXXX6883"",""Tr
```

```
ansactionDate"":""07.01.2022 17:33:00"",""AcquireName"": ""Кашалот
банк"",""InvoiceNumber"": ""224"",""CustomerEmail"":""test@gmai.com""}"

 Set Receipt = App.FiscalizeCheck(FiscalNumberRRO,JSONGoodsLstData,JSONPayData)

        Wscript.Echo "JsonVal = " & Receipt.JsonVal
        MsgBox "Фіскальний номер чека = " & Receipt.ReceiptFiscalNum
        MsgBox "Локальний номер чека = " & Receipt.ReceiptLocalNum
        MsgBox "Успіх реєстрації = " & Receipt.Return
        MsgBox "Тип чека = " & Receipt.Type
        MsgBox "ID зміни = " & Receipt.ShiftID
        Wscript.Echo "XML зареєстрованого чека = " & Receipt.XML
```

Приклад відповіді:

```
      JsonVal =
{"Ret":true,"ErrorString":"","Values":{"ShiftID":"92b59cb8-f34b-43dd-9cec-d8b187
448cbf","ReceiptFiscalNumber":"630168","ReceiptLocalNumber":"188","OfflineMode":
"False","Type":"10000000","Base64Str1251ReceiptXML":"'тут буде закодована
структура XML чека в стандарті Base64"}}
      Фіскальний номер чека = 630168
      Локальний номер чека = 188
      Успіх реєстрації = true
      Тип чека = 10000000
      ID зміни = 92b59cb8-f34b-43dd-9cec-d8b187448cbf
      XML зареєстрованого чека =
<?xml version="1.0" encoding="windows-1251"?>
<CHECK xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="check01.xsd">
  <CHECKHEAD>
    <DOCTYPE>0</DOCTYPE>
    <DOCSUBTYPE>0</DOCSUBTYPE>
    <UID>43DB4C60-50CD-4C5F-A0A2-5F72853B2A02</UID>
    <TIN>30300303</TIN>
    <ORGNM>test</ORGNM>
    <POINTNM>Магазин "ТЕСТ-ФСКО"</POINTNM>
    <POINTADDR>УКРАЇНА, м. Київ, вул. Басейна, буд. 5б</POINTADDR>
    <ORDERDATE>08012022</ORDERDATE>
    <ORDERTIME>200843</ORDERTIME>
    <ORDERNUM>188</ORDERNUM>
    <CASHDESKNUM>1007</CASHDESKNUM>
    <CASHREGISTERNUM>4000000000</CASHREGISTERNUM>
    <CASHIER>ТОВ "ТЕСТ-ФСКО"(Печатка)</CASHIER>
    <COMMENT>##################################

 # Тестовий коментар

##################################
    </COMMENT>
    <VER>1</VER>
    <ORDERTAXNUM>630168</ORDERTAXNUM>
    <!--<CASHALOTCOMMENT>
  <PAYMENTSTATUS>3</PAYMENTSTATUS>
  <PAYMENTORDER>0</PAYMENTORDER>
```

```
</CASHALOTCOMMENT>-->
  </CHECKHEAD>
  <CHECKTOTAL>
    <SUM>362.34</SUM>
    <DISCOUNTSUM>3.66</DISCOUNTSUM>
  </CHECKTOTAL>
  <CHECKPAY>
    <ROW ROWNUM="1">
      <PAYFORMCD>0</PAYFORMCD>
      <PAYFORMNM>ГОТІВКА</PAYFORMNM>
      <SUM>52.00</SUM>
    </ROW>
    <ROW ROWNUM="2">
      <PAYFORMCD>1</PAYFORMCD>
      <PAYFORMNM>КАРТКА</PAYFORMNM>
      <SUM>310.34</SUM>
      <PAYSYS>
        <ROW ROWNUM="1">
          <NAME>MASTERCARD</NAME>
          <ACQUIRENM>Кашалот банк</ACQUIRENM>
          <ACQUIRETRANSID>048867173620</ACQUIRETRANSID>
          <POSTRANSDATE>07012022173300</POSTRANSDATE>
          <POSTRANSNUM>224</POSTRANSNUM>
          <EPZDETAILS>XXXXXXXXXXXX6883</EPZDETAILS>
          <AUTHCD>68014B</AUTHCD>
          <SUM>310.34</SUM>
        </ROW>
      </PAYSYS>
    </ROW>
  </CHECKPAY>
  <CHECKTAX>
    <ROW ROWNUM="1">
      <TYPE>0</TYPE>
      <NAME>ПДВ</NAME>
      <LETTER>А</LETTER>
      <PRC>20.00</PRC>
      <TURNOVER>366.00</TURNOVER>
      <SOURCESUM>353.71</SOURCESUM>
      <SUM>58.95</SUM>
    </ROW>
    <ROW ROWNUM="2">
      <TYPE>1</TYPE>
      <NAME>Акциз</NAME>
      <LETTER>Щ</LETTER>
      <PRC>5.00</PRC>
      <TURNOVER>183.00</TURNOVER>
      <SOURCESUM>181.17</SOURCESUM>
      <SUM>8.63</SUM>
    </ROW>
  </CHECKTAX>
  <CHECKBODY>
    <ROW ROWNUM="1">
      <CODE>АртикулТовару1</CODE>
```

```
      <BARCODE>123456789</BARCODE>
      <NAME>НазваТовару1</NAME>
      <UNITCD>2009</UNITCD>
      <UNITNM>шт</UNITNM>
      <AMOUNT>1.000</AMOUNT>
      <PRICE>183.00</PRICE>
      <LETTERS>А</LETTERS>
      <COST>183.00</COST>
      <DISCOUNTTYPE>0</DISCOUNTTYPE>
      <DISCOUNTSUM>1.83</DISCOUNTSUM>
    </ROW>
    <ROW ROWNUM="2">
      <CODE>АртикулТовару2</CODE>
      <BARCODE>9999999999</BARCODE>
      <NAME>НазваТовару2</NAME>
      <UNITCD>2009</UNITCD>
      <UNITNM>шт</UNITNM>
      <AMOUNT>1.000</AMOUNT>
      <PRICE>183.00</PRICE>
      <LETTERS>АЩ</LETTERS>
      <COST>183.00</COST>
      <DISCOUNTTYPE>0</DISCOUNTTYPE>
      <DISCOUNTSUM>1.83</DISCOUNTSUM>
      <EXCISELABELS>
        <ROW ROWNUM="1">
          <EXCISELABEL>1111</EXCISELABEL>
        </ROW>
      </EXCISELABELS>
    </ROW>
  </CHECKBODY>
</CHECK>
```

---

6. Фіскалізація чека повернення товарів, послуг FiscalizeReturnCheck (FiscalNumberRRO, JSONGoodsLstData, JSONPayData, FiscalNumReturnReceipt)

Для створення та фіскалізації чека повернення товарів та/або послуг необхідно сформувати JSON структуру JSONGoodsLstData, що міститиме дані номенклатури, яка повертається, структуру JSON, структуру JSONPayData з формами повернення грошей та номер чека, за яким здійснюється повернення.

Параметри, які вказуються при використанні методу FiscalizeReturnCheck:

Ім'я	Тип	Опис
FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
JSONGoodsLstData	string	JSON структура з описом проданих товарів або послуг.
JSONPayData	string	JSON структура з підсумками оплати по чека.
FiscalNumReturnReceipt	string	Фіскальний номер чека, за яким здійснюється повернення.

Структури JSONGoodsLstData та JSONPayData визначені так само, як і в специфікації функції формування чека продажу **FiscalizeCheck**.

У відповіді повертається структура **CashalotApiRetVal**, що містить значення:

Return	bool	Ознака успішності виконання.
JsonVal	string	JSON-структура, що містить значення успішності операції, тип документу, помилки, локальний та фіскальний номери чека, ID зміни та Base64 рядок XML чека.
ReceiptFiscalNum	string	Фіскальний номер чека.
ReceiptLocalNum	string	Локальний номер чека.
ShiftID	string	Ідентифікаційний номер поточної зміни.
Type	int	Тип документа.
XML	string	XML-структура фіскалізованого чека.

Приклад **FiscalizeReturnCheck**:

```
Set App = CreateObject ("AddIn.CashaLotApi")
    FiscalNumberRRO = "4000000000"
    FiscalNumReturnReceipt  = "630168"
    App.SetParameter  "PathToCashalotDir", "C:\ProgramData\Cashalot\Cashalot"
    App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO
    JSONGoodsLstData = "{""ReceiptLst"":[{""VendorCode"":""АртикулТовару1"",
""Name"": ""НазваТовару1"", ""CdUKDZED"": """", ""CdDKPP"": """",
""GoodsType"":"""", ""Barcode"":""123456789"", ""UnitType"":""шт"",
""Quantity"": ""1"", ""Price"": ""183"", ""Amount"":""181,17"",
""DiscountPrc"":"""", ""DiscountSum"":""1.83"",
""VATRate"":""20"",""IsPriceIncludeVAT"":true, ""SumVAT"":""30.2"",
""IsExcise"":false, ""OtherParametrs"":null}, {""VendorCode"":
""АртикулТовару2"", ""Name"":""НазваТовару2"", ""CdUKDZED"":"""",
""CdDKPP"":"""", ""GoodsType"":"""", ""Barcode"":""9999999999"",
""UnitType"":""шт"", ""Quantity"":""1"", ""Price"":""183"",
""Amount"":""181,17"", ""DiscountPrc"":"""", ""DiscountSum"":""1.83"",
""VATRate"":""20"", ""IsPriceIncludeVAT"":true, ""SumVAT"":""30.2"",
""IsExcise"": true, ""ExciseStampBarcode"":""1111"",
""OtherParametrs"":null}],""Comment"":""################################\n\r #
Тестовий коментар \n\r################################""}"
    JSONPayData =
"{""SumCash"":""52"",""SumPayByCard"":""310,34"",""SumPayByCredit"":null,""SumPa
yByCertificate"":null,""SumPayCheck"":""362,34"",""TerminalID "":
""S1K700J8"",""ApprovalCode"": ""68014B"",""RRN"":
""048867173620"",""IssuerName"":""MASTERCARD"",""PAN"":""XXXXXXXXXXXX6883"",""Tr
ansactionDate"":""07.01.2022 17:33:00"",""AcquireName"": ""Кашалот
банк"",""InvoiceNumber"": ""224"",""CustomerEmail"":""test@gmai.com""}"

Set ResultReturnReceipt = App.FiscalizeReturnCheck(FiscalNumberRRO
,JSONGoodsLstData, JSONPayData, FiscalNumReturnReceipt)
    MsgBox "Результат виконання = " & ResultReturnReceipt.JsonVal
```

Приклад відповіді:

```
Результат виконання =
{"Ret":true,"ErrorString":"","Values":{"ShiftID":"92b59cb8-f34b-43dd-9cec-d8b187
448cbf","ReceiptFiscalNumber":"630169","ReceiptLocalNumber":"189","OfflineMode":
"False","Type":"10000001","Base64Str1251ReceiptXML":"PD94bWwgdmVyc2lvbj0iMS4wIiB
lbmNvZGluZz0id2luZG93cy0xMjUxIj8+DQo8Q0hFQ0sgeG1sbnM6eHNpPSJodHRwOi8vd3d3LnczLm9
yZy8yMDAxL1hNTFNjaGVtYS1pbnN0YW5jZSIgeHNpOm5vTmFtZXNwYWNlU2NoZW1hTG9jYX....."}}
```

7. Отримання локального номера чека за його фіскальним номером GetReceiptLocalNumberByFiscalNumber (FiscalNumberRRO, ReceiptFiscalNumber)

Метод дозволяє визначити локальний номер чека за відомим фіскальним номером чека. Пошук чека відбувається в локальній базі чеків, якщо в локальній базі чек не знайдено, виконується пошук чека на сервері ФСКО.

Параметри **GetReceiptLocalNumberByFiscalNumber**:

Ім'я	Тип	Опис
FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою
ReceiptFiscalNumber	number	Фіскальний номер чеку

У відповіді надходить локальний номер чека. Якщо номер чека не знайдено, то повертається пустий рядок. У випадку коли задано параметр NOAUTOOPENSHIFT = True, виклик функції при закритій зміні переривається і відображається повідомлення "Операція неможлива, необхідно відкрити зміну"

*Приклад виконання **GetReceiptLocalNumberByFiscalNumber**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
    FiscalNumberRRO = "4000000000"
    ReceiptFiscalNumber = "630169"
    App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
    App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

    result = App.GetReceiptLocalNumberByFiscalNumber(FiscalNumberRRO,
ReceiptFiscalNumber)
    MsgBox "Локальний номер = " & result
```

Приклад відповіді:

```
Локальний номер = 189
```

---

8. Отримання фіскального номера чека за його локальним номером GetReceiptFiscalNumberByLocalNumber (FiscalNumberRRO, ReceiptLocalNumber)

Метод дозволяє визначити локальний номер чека за відомим фіскальним номером чека. Пошук чека відбувається в локальній базі чеків, якщо в локальній базі чек не знайдено, виконується пошук чека на сервері ФСКО.

У відповіді надходить локальний номер чека. Якщо номер чека не знайдено, то повертається пустий рядок.

Параметри **GetReceiptFiscalNumberByLocalNumber**:

Ім'я	Тип	Опис
FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
ReceiptFiscalNumber	number	Фіскальний номер чека.

*Приклад виконання **GetReceiptLocalNumberByFiscalNumber**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
    FiscalNumberRRO = "4000000000"
    ReceiptFiscalNumber = "630169"
    App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
    App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

    result = App.GetReceiptFiscalNumberByLocalNumber(FiscalNumberRRO,
ReceiptLocalNumber)
    MsgBox "Локальний номер = " & result
```

Приклад відповіді:

```
Локальний номер = 636465
```

---

9. Отримання xml структури фіскального чека GetReceiptXML (FiscalNumberRRO, ReceiptFiscalNumber)

Метод дозволяє отримати XML-структуру зареєстрованого чека за його відомим фіскальним номером.

Параметри **GetReceiptXML:**

Ім'я	Тип	Опис
FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
ReceiptFiscalNumber	number	Фіскальний номер чека.

*Приклад виконання **GetReceiptXML**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
    FiscalNumberRRO = "4000000000"
    ReceiptFiscalNumber = "630169"

    App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
    App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

    receiptXML = App.GetReceiptXML(FiscalNumberRRO, ReceiptFiscalNumber)
    Wscript.Echo "XML receipt = " & receiptXML
```

У відповіді надходить XML-структура чека.

---

10. Отримати перший чек з періоду GetFirstReceiptByPeriod (FiscalNumberRRO, dateBeg, dateEnd)

Метод **GetFirstReceiptByPeriod** рекомендовано використовувати в ланцюжку запитів для отримання чеків за певний період часу. Дана функція викликається першою для підготовки списку чеків, і для отримання першого чека з вказаного періоду часу.

Параметри методу **GetFirstReceiptByPeriod**:

Ім'я	Тип	Опис
FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
dateBeg	string	Початок періоду в форматі dd.MM.yyyy.
dateEnd	string	Кінець періоду в форматі dd.MM.yyyy.

*Приклад виконання **GetFirstReceiptByPeriod**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
    FiscalNumberRRO = "4000000000"
    dateBeg = "01.01.2022"
    dateEnd = "09.01.2022"
    App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
    App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

 Set firstReceipt = App.GetFirstReceiptByPeriod(FiscalNumberRRO, dateBeg,
dateEnd)
    MsgBox "Перший чек в періоді = " & firstReceipt.JsonVal
```

У відповіді на **GetFirstReceiptByPeriod** повертається структура **CashalotApiRetVal**, що містить значення першого чека з вказаного періоду:

Return	bool	Успішність операції.
JsonVal	string	JSON-структура, що містить значення успішності операції, тип чека, помилки, локальний та фіскальний номери чека, ID зміни та Base64 рядок XML чека.
ReceiptFiscalNum	string	Фіскальний номер.
ReceiptLocalNum	string	Локальний номер.
ShiftID	string	Ідентифікаційний номер поточної зміни.
Type	int	Тип документа.
XML	string	XML-структура.

11. Отримати перший чек по зміні GetFirstReceiptByShift (FiscalNumberRRO, shiftID)

Метод **GetFirstReceiptByShift** рекомендовано використовувати в ланцюжку запитів для отримання чеків певної касової зміни. Дана функція викликається першою для підготовки списку чеків, і для отримання першого чека з вказаної касової зміни.

Параметри **GetFirstReceiptByShift**:

FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
shiftID	string	ID зміни.

Приклад:

```
Set App = CreateObject ("AddIn.CashaLotApi")
    FiscalNumberRRO = "4000000000"
    shiftID = "92b59cb8-f34b-43dd-9cec-d8b187448cbf"
    App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
    App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

 Set firstReceiptByShift = App.GetFirstReceiptByShift(FiscalNumberRRO, shiftID)
    MsgBox "Перший чек в зміні = " & firstReceiptByShift.JsonVal
```

У відповіді повертається структура **CashalotApiRetVal**, що містить значення першого чека з заданого періоду:

Return	bool	Успішність операції.
JsonVal	string	JSON-структура, що містить значення успішності операції, тип першого чека, помилки, локальний та фіскальний номери першого чека, ID зміни та Base64 рядок XML структури чека.
ReceiptFiscalNum	string	Фіскальний номер першого чека.
ReceiptLocalNum	string	Локальний номер першого чека.
ShiftID	string	Ідентифікаційний номер поточної зміни.
Type	int	Тип документа.
XML	string	XML-структура чека.

12. Отримати наступний чек з періоду GetNextReceipt (FiscalNumberRRO)

Метод **GetNextReceipt** використовується в ланцюжку запитів для отримання чеків за певний період часу. Цей метод викликається після виконання **GetFirstReceiptByPeriod** або **GetFirstReceiptByShift**, які готують список чеків. Функцію можна викликати декілька разів, поки не буде отримано весь список чеків.

Коли в списку більше немає чеків повертається false.

В параметрах **GetNextReceipt** має бути тільки FiscalNumberRRO (Фіскальний номер ПРРО. Ідентифікатор пристрою) з типом string

*Приклад виконання **GetNextReceipt**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   dateBeg = "01.01.2022"
   dateEnd = "09.01.2022"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

Set nextReceipt = App.GetFirstReceiptByPeriod(FiscalNumberRRO, dateBeg,
dateEnd)
Do while nextReceipt.Return
'Обробити отримані чеки
Set nextReceipt = App.GetNextReceipt(FiscalNumberRRO)
   if nextReceipt.Return <> False Then
   MsgBox "Наступний чек в періоді = " & nextReceipt.JsonVal
   Else
   MsgBox "Чеків немає"
   End if
Loop
```

У відповіді на **GetNextReceipt** повертається структура **CashalotApiRetVal**, що містить значення наступного чека з вказаного періоду:

Return	bool	Успішність операції. Коли в списку більше немає чеків, повертається false.
JsonVal	string	JSON-структура, що містить значення успішності операції, тип чека, помилки, локальний та фіскальний номери чека, ID зміни та Base64 рядок XML чека.
ReceiptFiscalNum	string	Фіскальний номер чека.
ReceiptLocalNum	string	Локальний номер чека.
ShiftID	string	Ідентифікаційний номер поточної зміни.
Type	int	Тип документа.
XML	string	XML-структура поточного чека.

---

13. Отримати інформацію по поточному стану каси GetCurrentStatus (FiscalNumberRRO)

Функція **GetCurrentStatus** використовується для отримання інформації по поточному стану каси. В параметрах **GetCurrentStatus** має бути тільки FiscalNumberRRO (Фіскальний номер ПРРО. Ідентифікатор пристрою) з типом string.

У відповіді повертається структура **CashalotApiRetVal**, що містить значення наступного чека з заданого періоду:

Return	bool	Успішність операції.
JsonVal	string	JSON-структура, що містить значення "Ret" – успішність операції, "ErrorString" – рядок з описом помилки (в разі невдачі), "Values" – масив параметрів каси.
ShiftID	string	Ідентифікаційний номер зміни. Якщо зміна відкрита, то відповідає ідентифікатору поточної зміни, якщо зміна закрита, то відповідає ідентифікатору останньої зміни.

Опис параметрів каси, які передаються в масиві Values структури JsonVal:

ShiftState	Стан зміни: 1 – закрита, 2 – відкрита.
ShiftStateStr	Стан зміни: "Closed" – закрита, "Opened" – відкрита.
ShiftLocalNumber	Локальний номер зміни.
ShiftFiscalNumber	Фіскальний номер зміни.
ShiftID	Ідентифікатор зміни.
LastCheckLocalNumber	Локальний номер останнього чека.
LastCheckFiscalNumber	Фіскальний номер останнього чека.
LastCheckDateTime	Мітка часу останнього чека.
IsOfflineMode	Стан офлайн-режиму каси: 0 – каса в онлайн-режимі, 1 – каса в офлайн-режимі.
NextLocalNumber	Наступний локальний номер чека.
CashBalance	Значення залишків готівки в касі.
Base64Str1251CurrentStatusXML	XML-рядок вказаних вище параметрів в форматі Base64.

*Приклад виконання **GetCurrentStatus**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

Set statusRRO = App.GetCurrentStatus(FiscalNumberRRO)
   MsgBox "Інформація про ПРРО = " & statusRRO.JsonVal
```

Приклад відповіді:

```
Інформація про ПРРО =
{"Ret":true,"ErrorString":"","Values":{"ShiftState":"2","ShiftStateStr":"Opened"
,"ShiftLocalNumber":"178","ShiftFiscalNumber":"630158","ShiftID":"92b59cb8-f34b-
43dd-9cec-d8b187448cbf","LastCheckLocalNumber":"190","LastCheckFiscalNumber":"63
0170","LastCheckDateTime":"2022-01-09T16:16:44","IsOfflineMode":"0","NextLocalNu
mber":"191","CashBalance":"3055","Base64Str1251CurrentStatusXML":"PD94bWwgdmVyc2
lvbj0iMS4wIiBlbmNvZGluZz0iVVRGLTgiPz48RG9jdW1lbnRPdXRwdXRQYXJhbWV0ZXJzPjxQYXJhbW
V0ZXJzIENoZWNrTnVtYmVyPSIxOTAiIENhc2hCYWxhbmNlPSIzMDU1IiBJc09mZmxpbmVNb2RlPSIwIi
BOZXh0TG9jYWxOdW1iZXI9IjE5MSIgTGFzdENoZWNrRmlzY2FsTnVtYmVyPSI2MzAxNzAiIFNoaWZ0Tn
VtYmVyPSIxNzgiIFNoaWZ0SUQ9IjkyYjU5Y2I4LWYzNGItNDNkZC05Y2VjLWQ4YjE4NzQ0OGNiZiIgU2
hpZnRGaXNjYWxOdW1iZXI9IjYzMDE1OCIgU2hpZnRTdGF0ZT0iMiIgRGF0ZVRpbWU9IjIwMjItMDEtMD
lUMTY6MTY6NDQiLz48L0RvY3VtZW50cHV0UGFyYW1ldGVycz4="}}
```

14. Отримати перший Z-звіт за період GetFirstZReportByPeriod (FiscalNumberRRO, dateBeg, dateEnd)

Метод **GetFirstZReportByPeriod** рекомендовано використовувати в ланцюжку запитів для отримання Z-звітів за певний період часу. Дана функція викликається першою для підготовки списку Z-звітів і для отримання першого Z-звіту з вказаного періоду часу.

Параметри **GetFirstZReportByPeriod:**

FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
dateBeg	string	Початок періоду в форматі dd.MM.yyyy.
dateEnd	string	Кінець періоду в форматі dd.MM.yyyy.

Приклад виконання **GetFirstZReportByPeriod**:

```
Set App = CreateObject ("AddIn.CashaLotApi")
    FiscalNumberRRO = "4000000000"
    dateBeg = "01.12.2021"
    dateEnd = "10.01.2022"
    App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
    App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

 Set firstZReport = App.GetFirstZReportByPeriod(FiscalNumberRRO, dateBeg,
dateEnd)
    MsgBox "Z-звіт = " & firstZReport.JsonVal
```

У відповіді повертається структура **CashalotApiRetVal**, що містить значення першого Z-звіту з заданого періоду:

Return	bool	Успішність операції.
JsonVal	string	JSON-структура, що містить значення успішності операції, помилки, тип документу Z-звіту, локальний та фіскальний номери Z-звіту, ID зміни та Base64 рядок XML структури Z-звіту .
ReceiptFiscalNum	string	Фіскальний номер першого Z-звіту.
ReceiptLocalNum	string	Локальний номер першого Z-звіту.
ShiftID	string	Ідентифікаційний номер поточної зміни.
Type	int	.Тип документа Z-звіту - 11000000.
XML	string	XML-структура Z-звіту.

15. Отримати наступний Z-звіт з періоду GetNextZReport (FiscalNumberRRO)

Метод **GetNextZReport** використовується в ланцюжку запитів для отримання Z-звітів за певний період часу. Цей метод викликається після виклику **GetFirstZReportByPeriod**, яка готує список Z-звітів. Можна викликати декілька разів, поки не буде отримано весь список Z-звітів.

Коли в списку більше немає чеків, повертається false.

В параметрах **GetNextZReport** має бути тільки FiscalNumberRRO (Фіскальний номер ПРРО. Ідентифікатор пристрою) з типом string

*Приклад виконання **GetNextZReport**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   dateBeg = "01.12.2021"
   dateEnd = "10.01.2022"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

Set nextZReport = App.GetFirstZReportByPeriod(FiscalNumberRRO, dateBeg, dateEnd)

Do while nextZReport.Return
Set nextZReport = App.GetNextZReport(FiscalNumberRRO)
   if nextZReport.Return <> False Then
   MsgBox "Наступний Z-звіт в періоді = " & nextZReport.ReceiptFiscalNum
   Else
   MsgBox "Z-звітів немає"
   End if
Loop
```

У відповіді повертається структура **CashalotApiRetVal**, що містить значення наступного Z-звіту з заданого періоду:

Return	bool	Успішність операції. Коли в списку більше нема Z-звіту повертається false.
JsonVal	string	JSON-структура, що містить значення успішності операції, тип документу, помилки, локальний та фіскальний номери Z-звіту, ID зміни та Base64 рядок XML структури Z-звіту.
ReceiptFiscalNum	string	Фіскальний номер Z-звіту.
ReceiptLocalNum	string	Локальний номер Z-звіту.
ShiftID	string	Ідентифікаційний номер поточної зміни.
Type	int	Тип документа.
XML	string	XML-структура Z-звіту.

---

16. Отримати Z-звіт касової зміни GetZReportByShift (FiscalNumberRRO, ShiftID)

Метод **GetZReportByShift** призначений для отримання Z-звіту вказаної касової зміни.

Параметри:

FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
ShiftID	string	Ідентифікатор касової зміни.

*Приклад виконання **GetZReportByShift**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   shiftID = "92b59cb8-f34b-43dd-9cec-d8b187448cbf"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

Set getZReport = App.GetZReportByShift(FiscalNumberRRO, shiftID)
   MsgBox "Z-звіт обраної зміни = " & getZReport.JsonVal
```

У відповіді на **GetZReportByShift** повертається структура **CashalotApiRetVal**, що містить значення Z-звіту вказаної касової зміни:

Return	bool	Успішність операції.
JsonVal	string	JSON-структура, що містить значення успішності операції, помилки, тип документу Z-звіту (11000000), локальний та фіскальний номери Z-звіту, ID зміни та Base64 рядок XML структури Z-звіту.
ReceiptFiscalNum	string	Фіскальний номер чека Z-звіту.
ReceiptLocalNum	string	Локальний номер чека Z-звіту.
ShiftID	string	Ідентифікаційний номер поточної зміни.
Type	int	Тип документа.
XML	string	XML-структура Z-звіту.

---

17. Вивести візуальне відображення чека на екран ShowReceipt (FiscalNumberRRO, ReceiptFiscalNumber)

Метод **ShowReceipt** використовується для виклику вікна з візуальним відображенням зареєстрованого чека. Вікно з відображенням чека також дозволяє використати додаткові функції друку або відправки чека на електронну пошту, кнопки яких присутні у вікні чека.

Параметри **ShowReceipt:**

FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
ReceiptFiscalNumber	number	Фіскальний номер чека.

*Приклад виконання **ShowReceipt**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   receiptFiscalNum = "626726"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

   openReceipt = App.ShowReceipt(FiscalNumberRRO, receiptFiscalNum)
   MsgBox openReceipt
```

У відповіді відкриється чек та повертається інформація про успішність виконання True або False.

---

18. Друкувати періодичний звіт по датам PrintPReportDate (FiscalNumberRRO, dateBeg, dateEnd, isShort)

Метод **PrintPReportDate** використовується для можливість друку періодичного звіту, використовуючи проміжки часу, що встановлюється між dateBeg та dateEnd у форматі "dd.mm.yyyy".

*Параметри **PrintPReportDate:***

FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
dateBeg	string	Початок періоду в форматі dd.MM.yyyy.
dateEnd	string	Кінець періоду в форматі dd.MM.yyyy.
isShort	string	Ознака скороченого звіту True/False.

*Приклад виконання **PrintPReportDate**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
    FiscalNumberRRO = "4000000000"
    dateBeg = "01.12.2021"
    dateEnd = "31.12.2021"
    isShort = "False"
    App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
    App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO
    pReport = App.PrintPReportDate (FiscalNumberRRO, dateBeg, dateEnd, isShort)
    MsgBox pReport
```

У відповіді на **PrintPReportDate** відкривається періодичний звіт (повний або скорочений) та повертається інформація про успішність виконання True або False.

---

19. Надрукувати довільний нефіскальний текст PrintTextDocument (FiscalNumberRRO, DocumentPackage)

Метод **PrintTextDocument** використовується для друку на чекову стрічку довільного тексту. Після виклику відображається вікно з зображенням тексту і можливістю друку. У разі потреби, можна викликати без відкриття касової зміни.

У випадку, коли задано автоматичну відправку чеків на друк і принтер для друку за замовчуванням, друк заданого тексту відбувається без попереднього перегляду:

```
App.SetParameter "AUTOPRINTMODE", "True"
App.SetParameter "DEFAULTPRINTERNAME", "xprint80-name")
```

*Параметри **PrintTextDocument**:*

FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
DocumentPackage	string	Текст для друку.

*Приклад виконання **PrintTextDocument**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
    FiscalNumberRRO = "4000000000"
    DocumentPackage = "З'їж цих м'яких київських перепічок і випий філіжанку
львівської кави."
    App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
    App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

    printText = App.PrintTextDocument(FiscalNumberRRO, DocumentPackage)
    MsgBox printText
```

У відповіді на **PrintTextDocument** відкривається вікно з вказаним текстом та повертається інформація про успішність виконання True або False.

---

20. Службове внесення грошей в касу ServiceInput (FiscalNumberRRO, Amount)

Для виконання службового внесення коштів в касу використовують метод **ServiceInput**.

Якщо задано параметр NOINTERFACEMODE = "False" (за допомогою функції SetParameter), то при цьому також виводиться службове вікно для підтвердження операції. Якщо задано параметр NOAUTOOPENSHIFT = “True”, виклик методу при закритій зміні переривається, відображається повідомлення "Операція неможлива, необхідно відкрити

зміну" і повертається false. При успішному виконанні відкривається чек службового внесення та повертається true.

*Параметри **ServiceInput:***

FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
Amount	string	Додатне значення суми внесення у форматі <Грн, Коп>.

*Приклад виконання **ServiceInput**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   Amount = "15.65"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

   deposit = App.ServiceInput(FiscalNumberRRO, Amount)
   MsgBox deposit
```

---

21. Службове внесення грошей в касу ServiceInputEx (FiscalNumberRRO, Amount)

Параметри використовуються такі ж, як і при виконанні **ServiceInput.** Різниця між **ServiceInput** та **ServiceInputEx** у відповіді, яку отримуємо на виконання методу.
У відповіді на **ServiceInputEx** повертається структура **CashalotApiRetVal**, що містить значення:

Return	bool	Успішність операції.
JsonVal	string	JSON-структура, що містить значення успішності операції, помилки, локальний та фіскальний номери документа службового внесення.
ShiftID	string	Ідентифікатор зміни.
ReceiptFiscalNum	string	Фіскальний номер чека.
ReceiptLocalNum	string	Локальний номер чека.
Type	int	Тип документа.

*Приклад виконання **ServiceInput**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   Amount = "15.65"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

Set depositRet  = App.ServiceInputEx(FiscalNumberRRO, Amount)
   MsgBox "JSON = " & depositRet.JsonVal
```

---

22. Службова видача грошей з каси ServiceOutput (FiscalNumberRRO, Amount)

Для виконання службової видачі з каси використовують функцію **ServiceOutput**. Якщо задано параметр NOINTERFACEMODE = "False" (за допомогою функції SetParameter), то при цьому також виводиться службове вікно для підтвердження операції. Якщо задано параметр NOAUTOOPENSHIFT= “True”, виклик функції при закритій зміні переривається, та з’являється повідомлення "Операція неможлива, необхідно відкрити зміну". При успішному виконанні відкривається чек службової видачі та повертається true.

*Параметри **ServiceOutput:***

FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
Amount	string	Додатне значення суми внесення у форматі <Грн, Коп>.

*Приклад виконання **ServiceOutput**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   Amount = "15.65"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

   withdrawRet = App.ServiceOutput(FiscalNumberRRO, Amount)
   MsgBox deposit
```

---

23. Службова видача грошей з каси ServiceOutputEx (FiscalNumberRRO, Amount)

Параметри використовуються такі ж, як і при виконанні **ServiceOutput.** Різниця між **ServiceOutput** та **ServiceOutputEx** у відповіді, яку отримуємо на виконання методу. У відповіді на **ServiceOutputEx** повертається структура **CashalotApiRetVal**, що містить значення:

Return	bool	Успішність операції.
JsonVal	string	JSON-структура, що містить значення успішності операції, помилки, локальний та фіскальний номери документа.
ShiftID	string	Ідентифікатор зміни.
ReceiptFiscalNum	string	Фіскальний номер чека.
ReceiptLocalNum	string	Локальний номер чека.
Type	int	Тип документа.

*Приклад виконання **ServiceOutputEx**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   Amount = "15.65"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

Set withdrawRet  = App.ServiceInputEx(FiscalNumberRRO, Amount)
   MsgBox "JSON = " & withdrawRet.JsonVal
```

---

24. Отримання опису останньої помилки GetLastError()

У разі неуспішної операції виклик цього методу дозволяє отримати опис останньої помилки. **GetLastError** немає вхідних параметрів. Відповідь має тип string

*Приклад виконання **GetLastError**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

   lastError = App.GetLastError()
   MsgBox lastError
```

25. Формування X-звіту PrintXReport (FiscalNumberRRO)

Метод **PrintXReport** дозволяє викликати на екран X-звіт для перегляду та відправки на друк. В параметрах **PrintXReport** має бути тільки FiscalNumberRRO (Фіскальний номер ПРРО. Ідентифікатор пристрою) з типом string. При успішному виконанні відкривається X-звіт та повертається true.

Приклад:

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

   XReport = App.PrintXReport(FiscalNumberRRO)
   MsgBox XReport
```

---

26. Формування X-звіту GetXReport (FiscalNumberRRO, ShowPrintReport)

GetXReport дозволяє сформувати і програмно отримати X-звіт у вигляді JSON-структури.

Параметри:

FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
ShowPrintReport	bool	Ознака необхідності вивести звіт на друк. Якщо "true", то на екран виводиться візуальне відображення X-звіту для перегляду та можливістю вручну роздрукувати звіт (аналогічно функції PrintXReport).

*Приклад **GetXReport**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   ShowPrintReport = "True"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

Set XReport = App.GetXReport(FiscalNumberRRO, ShowPrintReport)
   MsgBox XReport.JsonVal
```

У відповіді повертається структура **CashalotApiRetVal**, що містить значення:

Return	bool	Успішність операції.
JsonVal	string	JSON-структура, яка містить інформацію X-звіту.
ShiftId	string	Ідентифікатор зміни, в якій сформовано X-звіт.
Type	int	Тип документа.
XML	string	XML-структура X-звіту.

В **JsonVal**, а саме в “Values”, надходять наступні об'єкти:

ServiceInput	Загальна сума службових внесень по зміні
ServiceOutput	Загальна сума службових видач по зміні
OrderCount	Кількість чеків продажу
OrderExpCount	Кількість чеків повернення
SumCash	Поточний залишок готівки в касі
TotalSUM	Загальна сума продажу
TotalSUMCash	Загальна сума продажу "Готівка"
TotalSUMCard	Загальна сума продажу "Картка"

TotalTAXSUM	Загальна сума ПДВ по продажу
ReturnSUM	Загальна сума повернення
ReturnSUMCash	Загальна сума повернення "Готівка"
ReturnSUMCard	Загальна сума повернення "Картка"
ReturnTAXSUM	Загальна сума ПДВ по продажу
TotalTAX_<1>_<2>%_<3>	Деталізація податків по фіскальним чекам
ReturnTAX_<1>_<2>%_<3>	Деталізація податків по чекам повернення
Base64Str1251ReportXML	Дані Z-звіту у вигляді XML-структури в кодуванні Windows-1251 конвертованої в Base64-рядок

Назви об'єктів податків чеків оплати TotalTAX та повернення ReturnTAX побудовані з використанням складових, що визначаються на основі карток проданих товарів:

Складова	Зміст складової	Можливі значення	Приклад
<1>	Літера податку зазначена в картках товарів	А, Б, В, ...	TotalTAX_А_20%_SUM ReturnTAX_Б_5%_TURNOVER TotalTAX_Є_10%_REM1OVEDTAXES
<2>	Відсоток податку	0, 5, 20, ...	
<3>	Сума	SUM	
	Обіг	TURNOVER	
	Знижка	DISCOUNTSUM	
	Виключені податки	REMOVEDTAXES	

---

27. Отримати останній Z-звіт GetLastZReport (FiscalNumberRRO, ShowPrintReport)

Виклик функції **GetLastZReport** дозволяє отримати інформацію по останньому зареєстрованому Z-звіту у вигляді структури **CashalotApiRetVal**.

Параметри:

FiscalNumberRRO	string	Фіскальний номер ПРРО. Ідентифікатор пристрою.
ShowPrintReport	bool	Ознака необхідності вивести звіт на друк. Якщо "true", то на екран виводиться візуальне відображення X-звіту для перегляду та можливістю вручну роздрукувати звіт (аналогічно функції PrintXReport).

Приклад:

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   ShowPrintReport = "True"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

Set ZReport = App.GetLastZReport(FiscalNumberRRO, ShowPrintReport)
   MsgBox ZReport.JsonVal
```

У відповіді повертається структура **CashalotApiRetVal**, що містить значення:

Return	bool	Успішність операції.
JsonVal	string	JSON-структура, яка містить інформацію останнього зареєстрованого Z-звіту. Опис JsonVal.
ShiftId	string	Ідентифікатор зміни, в якій сформовано Z-звіт.
Type	int	Тип документа.
XML	string	XML-структура Z-звіту.
ReceiptFiscalNum	string	Фіскальний номер документа.
ReceiptLocalNum	string	Локальний номер документа.

---

28. Формування Z-звіту PrintZReport (FiscalNumberRRO)

Метод **PrintZReport** призводить до виконання послідовності операцій аналогічних до виклику функції **CloseShift** в такій послідовності: службова видача з каси наявної готівки, формування Z-звіту та закриття зміни. Проте, виконання методу **PrintZReport** повертає ознаку успішності операції (true/false), в той час як виконання **CloseShift** повертає структуру **CashalotApiRetVal**.

В параметрах **PrintZReport** має бути тільки FiscalNumberRRO (Фіскальний номер ПРРО. Ідентифікатор пристрою) з типом string. При успішному виконанні відкривається Z-звіт та повертається true.

Увага! Якщо встановлений параметр NOINTERFACEMODE = True, то службову видачу, потрібно передбачити перед виконанням методу **PrintZReport.**

*Приклад **PrintZReport**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000015625"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

   createZReport = App.PrintZReport (FiscalNumberRRO)
   MsgBox createZReport
```

---

29. Закриття поточної зміни CloseShift (FiscalNumberRRO)

Метод **CloseShift** використовується для закриття поточної зміни. В параметрах **CloseShift** має бути тільки FiscalNumberRRO (Фіскальний номер ПРРО. Ідентифікатор пристрою) з типом string.

Увага! Якщо встановлений параметр NOINTERFACEMODE = True, то службову видачу, потрібно передбачити перед виконанням методу **CloseShift.**

*Приклад виконання **CloseShift**:*

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

Set finishShift = App.CloseShift(FiscalNumberRRO)
   MsgBox "JSON = " & finishShift.JsonVal
```

У відповіді на **CloseShift** повертається структура **CashalotApiRetVal**, що містить значення Z-звіту закритої касової зміни:

Return	bool	Успішність операції.
JsonVal	string	JSON-структура, яка містить інформацію останнього зареєстрованого Z-звіту. Опис JsonVal.
ShiftId	string	Ідентифікатор зміни, в якій сформовано Z-звіт.
Type	int	Тип документа.
XML	string	XML-структура Z-звіту.
ReceiptFiscalNum	string	Фіскальний номер документа.
ReceiptLocalNum	string	Локальний номер документа.

---

30. Метод ручного переводу ПРРО в режим офлайн SetOfflineMode (FiscalNumberRRO, AutoExitToOnline)

Метод використовується для ручного переводу ПРРО в режим офлайн.
Даний метод містить такі параметри:

FiscalNumberRRO	string	Фіскальний номер ПРРО
AutoExitToOnline	bool	Ознака автоматичного виходу в онлайн. TRUE - автоматично виходити з офлайну при наявності інтернету; FALSE - виходити з офлайну лише при виклику методу **TryGoToOnlineMode** або після закриття зміни.

Виклик **SetOfflineMode** не означає автоматичний перехід в офлайн одразу в момент визову, перехід до офлайну буде здійснено при будь-якій фіскальній операції(відкриття/закриття зміни, службовій видачі/внесенню або реєстрації чека) після визову **SetOfflineMode**.

В якості відповіді повертається стандартна структура параметрів повернення **CashalotApiRetVal** з заповненим параметром Return(Ознака успішності виконання).

Приклад виконання **SetOfflineMode**:

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   AutoExitToOnline = False
   Amount = "15.65"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

Set ret = App.SetOfflineMode (FiscalNumberRRO, AutoExitToOnline)
   MsgBox ret.Return
   App.ServiceInput(FiscalNumberRRO, Amount)
```

---

31. Метод ручного переводу ПРРО в режим онлайн TryGoToOnlineMode (FiscalNumberRRO)

Метод використовується для ручного переводу ПРРО в режим онлайн. В якості параметру метод **TryGoToOnlineMode** містить фіскальний номер ПРРО(FiscalNumberRRO).

Приклад виконання **TryGoToOnlineMode**:

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

Set ret = App.TryGoToOnlineMode(FiscalNumberRRO)
   MsgBox ret.Return
```

В якості відповіді повертається стандартна структура параметрів повернення **CashalotApiRetVal** з заповненим параметром Return(Ознака успішності виконання).

---

32. Синхронізація залишків по товарам SyncBalanceOnGoods (FiscalNumberRRO)

Метод призначений для ручної синхронізації залишків по товарам з кабінету користувача в локальну базу даних Cashalot. Для можливості виконання методу необхідно встановити **параметр ведення обліку товарів в API** в режимі 1 або 2 та ознаку ведення обліку товарів в кабінеті користувача в розділі «Склад - Налаштування - Облік товарів у ПРРО». В якості параметру метод **SyncBalanceOnGoods** містить фіскальний номер ПРРО (FiscalNumberRRO).

Приклад виконання **SyncBalanceOnGoods**:

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

Set ret = App.SyncBalanceOnGoods(FiscalNumberRRO)
   MsgBox ret.Return
```

В якості відповіді повертається стандартна структура параметрів повернення **CashalotApiRetVal** з заповненим параметром Return(Ознака успішності виконання).

---

33. Синхронізація товарів та залишків SyncGoodsWithBalance (FiscalNumberRRO)

Метод призначений для ручної синхронізації товарів та залишків по ним з кабінету користувача в локальну базу даних Cashalot. Для можливості виконання методу необхідно встановити **параметр ведення обліку товарів в API** в режимі 1 або 2 та ознаку ведення обліку товарів в кабінеті користувача в розділі «Склад - Налаштування - Облік товарів у ПРРО». В якості параметру метод **SyncGoodsWithBalance** містить фіскальний номер ПРРО (FiscalNumberRRO).

Приклад виконання **SyncGoodsWithBalance**:

```
Set App = CreateObject ("AddIn.CashaLotApi")
   FiscalNumberRRO = "4000000000"
   App.SetParameter  "PathToCashalotDir", "C:\Cashalot"
   App.SetParameter "DeviceIDFnRRO", FiscalNumberRRO

Set ret = App.SyncGoodsWithBalance(FiscalNumberRRO)
   MsgBox ret.Return
```

В якості відповіді повертається стандартна структура параметрів повернення **CashalotApiRetVal** з заповненим параметром Return(Ознака успішності виконання).

ЛОМБАРДНІ ОПЕРАЦІЇ

Для фіскалізації ломбардних послуг введено системні номенклатури, які необхідно використовувати при виклику методу фіскалізації чека **FiskalizeCheck**. Повернення товару при ломбардних операціях не підтримується.

Також необхідно вказати тип документа DocType = 4 в JSON-структурі параметрів чека. У разі потреби можна вказати номер договору DocNumber, за яким надається ломбардна послуга.

Напрям руху необхідно враховувати при обчисленні кінцевого підсумку в чеку: від позицій типу 2 віднімаються позиції типу 3.

Системні номенклатури для ломбардних операцій:

Артикул	Назва	Тип послуги	Напрям руху*
"ЛО-00001"	"Повернення страхування"	"3"	-
"ЛО-00002"	"Надання кредиту"	"3"	-
"ЛО-00003"	"Добір кредиту"	"3"	-
"ЛО-00004"	"Надання додаткового кредиту"	"3"	-
"ЛО-00005"	"Зберігання"	"2"	+
"ЛО-00006"	"Страховий платіж"	"2"	+
"ЛО-00007"	"Часткове повернення кредиту (Частковий викуп)"	"2"	+
"ЛО-00008"	"Повне повернення кредиту (Повний викуп)"	"2"	+
"ЛО-00009"	"Повернення кредиту"	"2"	+
"ЛО-00010"	"Користування кредитом"	"2"	+
"ЛО-00011"	"Сплата відсотків по кредиту"	"2"	+
"ЛО-00012"	"Сплата пені"	"2"	+

* Напрям руху "-" видача грошей з каси; "+" повернення грошей в касу.

Рекомендований порядок роботи з CashalotApi

Для коректної роботи розрахунково-облікової системи рекомендується описана послідовність виклику функцій при виконанні касових операцій.

Встановлення параметрів

Перед початком роботи рекомендовано в **SetParameter** або **SetParameterEx** встановити параметри в такій послідовності:

SPECIFICAPIMODE (за необхідності)
PathToCashalotDir
DeviceIDFnRRO
NOAUTOUPDATE
NOINTERFACEMODE
PathToCertificate
PwdToCertificate
USETOKEN
NOAUTOOPENSHIFT
AUTOPRINTMODE
DEFAULTPRINTERNAME
SHOWREPORTADDITIONALINFO
USEGOODSSTORAGEMODE (за необхідністю)

Відкриття зміни

Для відкриття касової зміни необхідно скористатися методом **OpenShift**.

Продаж товарів, послуг

Для формування чека та його фіскалізації створюються JSON-структури з масивом товарів та підсумками оплати по чеку, які передаються як параметри для виконання методу фіскалізації чека **FiscalizeCheck**.

У чеку необхідно сформувати щонайменше 1 рядок товарів та встановити розширений порядок оплати для поточного чека PaymentOrderType.

Спосіб оплати (0-звичайна | 1 - передоплата | 2 - післяплата | 3 - інтернет продаж).

Повернення товарів, рекомпенсація послуг

Для повернення товару або рекомпенсації послуги необхідно скористатися методом **FiscalizeReturnCheck**, в якому потрібно вказати фіскальний номер чека, за яким було продано товар або послугу.

Пошук фіскального та локального номера чека

Для визначення фіскального номера чека за його локальним номером використовується метод **GetReceiptFiscaNumberByLocalNumber**.

Для визначення локального номера чека за його фіскальним номером використовується функція GetReceiptLocalNumberByFiscalNumber.

Отримання XML-структури чека

Метод GetReceiptXML повертає XML-структуру чека за його вказаним фіскальним номером.

Відображення зареєстрованого чека

Метод ShowReceipt відкриває вікно з візуальним відображенням чека. У вікні є кнопки "Друк", "Надіслати на email", “Надіслати чек в SMS” які дозволяють роздрукувати або відправити на електронну пошту чек та відправити чек в SMS.

Друкувати періодичний звіт по датам

Метод PrintPReportDate використовується для друку періодичного звіту за проміжок часу, що фіксується значеннями (дата початку періоду - у форматі "dd.mm.yyyy" та дата закінчення періоду - у форматі "dd.mm.yyyy").

Налаштування електронної пошти

Електронну пошту можна налаштувати в РМК Cashälot, а саме в “Загальні налаштування - Господарська одиниця - Налаштування”. Для налаштування необхідно зазначити електронну адресу, з якої будуть відправлятися повідомлення, а також, налаштування з'єднання з поштовою скринькою: вказати SMTP-сервер, порт, ім'я для входу в поштову скриньку, пароль поштової скриньки.

Відправляти чек на пошту можна після фіскалізації чека або виклику функції відображення чека (ShowReceipt). Для цього у вікні візуального відображення зареєстрованого чека необхідно натиснути кнопку "Надіслати на email", після чого відкриється вікно, де можна заповнити пошту покупця та виконати відправку.

Службове внесення грошей в касу

Для виконання службового внесення грошей в касу використовується методи ServiceInput або ServiceInputEx.

Службова видача грошей з каси

Для виконання службової видачі з каси використовується методи ServiceOutput або ServiceOutputEx.

Формування X-звіту

X-звіт формується в результаті виконання методів PrintXReport або GetXReport.

Отримання чеків

Для отримання чеків з історії операцій реалізовано функції, що передбачають завантаження чеків за певний період часу або за певну касову зміну. Дані завантажуються з бази даних Cashälot.

Для формування списку чеків за заданий період часу скористайтеся:

GetFirstReceiptByPeriod

Для формування списку чеків за задану касову зміну скористайтеся:

GetFirstReceiptByShift

Обидві функції формують список чеків і повертають перший чек з сформованого списку. Для отримання наступних чеків скористайтеся:

GetNextReceipt

необхідну кількість разів, або поки не буде вичерпано список чеків.

Отримання документів Z-звітів закритих касових змін

Для отримання Z-звітів з історії операцій реалізовано методи, що передбачають завантаження документів за певний період часу або для певної касової зміни. Дані завантажуються з бази даних Cashälot.

Для формування списку документів за заданий період часу скористайтеся:

GetFirstZReportByPeriod

Функція формує список чеків і повертає перший документ із сформованого списку.

Для отримання наступних документів скористайтеся:

GetNextZReport

необхідну кількість разів, або поки не буде вичерпано список документів.

Для отримання Z-звіту заданої касової зміни скористайтеся:

GetZReportByShift

Якщо звіт для вказаної касової зміни відсутній (наприклад, поточна касова зміна), метод поверне ознаку неуспішності операції.

Отримання останнього Z-звіту

Для отримання і перегляду останнього зареєстрованого Z-звіту використовується метод **GetLastZReport**.

Закриття зміни, Z-звіт

Формування Z-звіту відбувається при закритті зміни, тому виконання операції закриття зміни та формування Z-звіту рівнозначні за результатом. Для цього виконується один з методів: **PrintZReport** або **CloseShift**.

Виклик зазначених функцій призводить до виконання послідовності операцій: службова видача з каси наявної готівки, формування Z-звіту та закриття зміни. Проте, виконання функції **PrintZReport** повертає ознаку успішності операції, в той час як виконання **CloseShift** повертає структуру **CashalotApiRetVal**. Якщо встановлений параметр NOINTERFACEMODE = True, то службову видачу потрібно передбачити перед виконанням методу **CloseShift**.

Приклад реалізації підключення COM бібліотеки на C++

Створення файлів заголовків *.h

Якщо відсутні файли заголовків, або потрібно їх оновити:

1. Генеруємо через імпорт, вказавши шлях до COM бібліотеки, наприклад:

#import "C:\\Users\User1\\AppData\\Local\\Cashalot\\CashaLotApi64.dll"

2. Компілюємо solution з рядком import (після компіляції цей рядок прибираємо).
3. Далі в папці debug\ вашого solution з'являються файли CashaLotApi64.tlh та CashaLotApi64.tli.
4. Переносимо їх в каталог solution, перейменовуємо в CashaLotApi.tlh, CashaLotApi.tli (за бажанням, можна зібрати два x32 та x64, та підключати директивами).
5. Всередині CashaLotApi.tlh прибираємо повний шлях до CashaLotApi64.tli і також міняємо ім'я на CashaLotApi.tli.
6. Заголовні файли готові до використання, підключаємо їх в solution:

#include "CashaLotApi.tlh" using namespace CashaLotApi.

Зауваження

1. При старті програми додаємо ініціалізацію роботи з COM об'єктами: CoInitialize(NULL).
2. При закритті програми припиняємо роботу з COM об'єктами: CoUninitialize().
3. Перед початком роботи з COM-об'єктом ініціалізуємо його викликом InitCashaLotApiComInterface(), перед виходом з програми звільняємо ресурси викликом функції TermCashaLotApiComInterface().
4. Функції повертають структуру ICashalotApiRetVal або BOOL.

ЛІСТИНГ

```
#include "stdafx.h"
#include "SampleUseCashaLotCOMAPI_CPP.h"
#include "SampleUseCashaLotCOMAPI_CPPDlg.h"
#include "afxdialogex.h"
#include <string>       // std::string
#include <sstream>      // std::ostringstream
#include <algorithm>
#include <list>

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


#include "CashaLotApi.tlh"
using namespace CashaLotApi;

extern LPCTSTR pAppName;

CSampleUseCashaLotCOMAPICPPDlg::CSampleUseCashaLotCOMAPICPPDlg(CWnd* pParent /*=nullptr*/)
    : CDialogEx(IDD_SAMPLEUSECASHALOTCOMAPI_CPP_DIALOG, pParent)
    , m_FiscalNumberRRO(_T(""))
    , m_PathToCashaLot(_T(""))
    , m_IsNoInterfaceMode(FALSE)
    , m_PathCert(_T(""))
    , m_pwdCert(_T(""))
    , m_PrinterName(_T(""))
    , m_useToken(FALSE)
    , m_UseDefCashalotPrinter(FALSE)
    , m_PrintUseDefWindowsPrinter(FALSE)
    , m_PrintUseNamedPrinter(FALSE)
    , m_DateBeg(COleDateTime::GetCurrentTime())
    , m_DateEnd(COleDateTime::GetCurrentTime())
{
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CSampleUseCashaLotCOMAPICPPDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);
    DDX_Control(pDX, IDC_EDPATH, m_edPathToCashaLot);
    DDX_Control(pDX, IDC_EDFISKNUMRCP, m_edFiscNumRcp);
    DDX_Text(pDX, IDC_EDPATH2, m_FiscalNumberRRO);
    DDX_Text(pDX, IDC_EDPATH, m_PathToCashaLot);
    DDX_Control(pDX, IDC_EDPATH2, m_edFiscalNumberRRO);
    DDX_Control(pDX, IDC_BTGETPATH, m_btnPath);
    DDX_Check(pDX, IDC_CHECKNOINTERFACEMODE, m_IsNoInterfaceMode);
    DDX_Text(pDX, IDC_EDPATHTOCERT, m_PathCert);
    DDX_Text(pDX, IDC_EDPWD, m_pwdCert);
    DDX_Control(pDX, IDC_RADIO1, m_rdbtPrintUseNamedPrinter);
    DDX_Control(pDX, IDC_RADIO2, m_rdbtPrintUseDefPrinter);

    DDX_Text(pDX, IDC_EDPATH5, m_PrinterName);
    DDX_Control(pDX, IDC_EDPATHTOCERT, m_edPathCert);
    DDX_Control(pDX, IDC_EDPWD, m_edPwdCert);
    DDX_Control(pDX, IDC_CHECKNOINTERFACEMODE, m_chNoInterfaceMode);
    DDX_Check(pDX, IDC_CHECKUSETOKEN, m_useToken);
    DDX_Control(pDX, IDC_CHECKUSETOKEN, m_chUseToken);
    DDX_Check(pDX, IDC_CHECKUSECASHALOTPRINTERSETTINGS, m_UseDefCashalotPrinter);
```

```
    DDX_Control(pDX, IDC_CHECKUSECASHALOTPRINTERSETTINGS, m_chUseDefCashalotPrinter);
    DDX_Control(pDX, IDC_EDPATH5, m_edPrinterName);
    DDX_Radio(pDX, IDC_RADIO2, m_PrintUseDefWindowsPrinter);
    DDX_Radio(pDX, IDC_RADIO1, m_PrintUseNamedPrinter);
    DDX_DateTimeCtrl(pDX, IDC_DATETIMEPICKER1, m_DateBeg);
    DDX_DateTimeCtrl(pDX, IDC_DATETIMEPICKER2, m_DateEnd);
}

BEGIN_MESSAGE_MAP(CSampleUseCashaLotCOMAPICPPDlg, CDialogEx)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_BTOPENSHIFT, &CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtopenshift)
    ON_BN_CLICKED(IDC_BTCloseShift, &CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtcloseshift)
    ON_BN_CLICKED(IDC_BTFiscalizeReceipt,
&CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtfiscalizereceipt)
    ON_BN_CLICKED(IDC_BTReturnReceipt,
&CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtreturnreceipt)
    ON_BN_CLICKED(IDC_BTServiceInput,
&CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtserviceinput)
    ON_BN_CLICKED(IDC_BTServiceOutput,
&CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtserviceoutput)
    ON_BN_CLICKED(IDC_BTServiceOutput2, &CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedPrintXRep)
    ON_WM_DESTROY()
    ON_BN_CLICKED(IDC_BTGETPATH, &CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtgetpath)
    ON_BN_CLICKED(IDC_BTGETPATHTOCERT,
&CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtgetpathtocert)
    ON_BN_CLICKED(IDC_BTPrintXRepEx,
&CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtprintxrepex)
    ON_BN_CLICKED(IDC_CHECKUSECASHALOTPRINTERSETTINGS,
&CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedCheckusecashalotprintersettings)
    ON_BN_CLICKED(IDC_RADIO2, &CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedRadio2)
    ON_BN_CLICKED(IDC_RADIO1, &CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedRadio1)
    ON_BN_CLICKED(IDC_BTGETPRINTERNAME,
&CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtgetprintername)
    ON_BN_CLICKED(IDC_BTPrintLastZRep,
&CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtprintlastzrep)
    ON_BN_CLICKED(IDC_BTGETRECEIPTANDREPFROMPERIOD,
&CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedGetRcpByPeriod)
END_MESSAGE_MAP()

// CSampleUseCashaLotCOMAPICPPDlg message handlers

BOOL CSampleUseCashaLotCOMAPICPPDlg::OnInitDialog()
{
    CDialogEx::OnInitDialog();

    // Set the icon for this dialog.  The framework does this automatically
    //  when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE);         // Set big icon
    SetIcon(m_hIcon, FALSE);        // Set small icon

    // TODO: Add extra initialization here
    CWinApp* pApp = AfxGetApp();
    m_PathToCashaLot = pApp->GetProfileString(pAppName, _T("PathToCashaLot"));
    m_FiscalNumberRRO = pApp->GetProfileString(pAppName, _T("FiscalNumberRRO"));
    m_PathCert = pApp->GetProfileString(pAppName, _T("PathToCert"));
    m_pwdCert = pApp->GetProfileString(pAppName, _T("PwdToCert"));
    m_IsNoInterfaceMode = pApp->GetProfileString(pAppName, _T("NoInterfaceMode")) == L"1";
    m_useToken = pApp->GetProfileString(pAppName, _T("UseToken")) == L"1";
```

```
    m_PrintUseDefWindowsPrinter = (pApp->GetProfileString(pAppName,
_T("PrintUseDefWindowsPrinter")) == L"0")? 0 : -1;
    m_PrintUseNamedPrinter = (pApp->GetProfileString(pAppName, _T("PrintUseNamedPrinter")) ==
L"0")? 0 : -1;;
    m_PrinterName = pApp->GetProfileString(pAppName, _T("PrinterName"));

    m_edPrinterName.SetWindowText(m_PrinterName);

    CString defPrndPrm = pApp->GetProfileString(pAppName, _T("UseDefCashalotPrinter"));
    if (defPrndPrm == L"") defPrndPrm = L"1";
    m_UseDefCashalotPrinter = defPrndPrm == L"1";

    m_chUseDefCashalotPrinter.SetCheck(m_UseDefCashalotPrinter);

    m_chNoInterfaceMode.SetCheck(m_IsNoInterfaceMode);
    m_chUseToken.SetCheck(m_useToken);

    m_edFiscalNumberRRO.SetWindowText(m_FiscalNumberRRO);
    m_edPathToCashaLot.SetWindowText(m_PathToCashaLot);
    m_edPathCert.SetWindowText(m_PathCert);
    m_edPwdCert.SetWindowText(m_pwdCert);
    UpdateData(0);

    OnBnClickedCheckusecashalotprintersettings();
    OnBnClickedRadio2();

    return TRUE;  // return TRUE  unless you set the focus to a control
}
void CSampleUseCashaLotCOMAPICPPDlg::OnDestroy()
{
    UpdateData();
    CDialogEx::OnDestroy();

    CWinApp* pApp = AfxGetApp();
    pApp->WriteProfileString(pAppName, _T("PathToCashaLot"), m_PathToCashaLot);
    pApp->WriteProfileString(pAppName, _T("FiscalNumberRRO"), m_FiscalNumberRRO);
    pApp->WriteProfileString(pAppName, _T("PathToCert"), m_PathCert);
    pApp->WriteProfileString(pAppName, _T("PwdToCert"), m_pwdCert);
    pApp->WriteProfileString(pAppName, _T("NoInterfaceMode"), ((m_IsNoInterfaceMode == TRUE)?
L"1" : L"0"));
    pApp->WriteProfileString(pAppName, _T("UseToken"), ((m_useToken == TRUE)? L"1" : L"0"));
    pApp->WriteProfileString(pAppName, _T("UseDefCashalotPrinter"), ((m_UseDefCashalotPrinter
== TRUE)? L"1" : L"0"));

    pApp->WriteProfileString(pAppName, _T("PrintUseDefWindowsPrinter"),
(m_PrintUseDefWindowsPrinter == 0 ? L"0" : L"-1"));
    pApp->WriteProfileString(pAppName, _T("PrintUseNamedPrinter"), (m_PrintUseNamedPrinter ==
0 ? L"0" : L"-1"));
    pApp->WriteProfileString(pAppName, _T("PrinterName"), m_PrinterName );

}

void CSampleUseCashaLotCOMAPICPPDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
    CDialogEx::OnSysCommand(nID, lParam);
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CSampleUseCashaLotCOMAPICPPDlg::OnPaint()
```

```
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting

        SendMessage(WM_ICONERASEBKGND, reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialogEx::OnPaint();
    }
}

// The system calls this function to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CSampleUseCashaLotCOMAPICPPDlg::OnQueryDragIcon()
{
    return static_cast<HCURSOR>(m_hIcon);
}

static int CALLBACK BrowseCallbackProc(HWND hwnd, UINT uMsg, LPARAM lParam, LPARAM lpData)
{

    if (uMsg == BFFM_INITIALIZED)
    {
        std::string tmp = (const char *)lpData;

        SendMessage(hwnd, BFFM_SETSELECTION, TRUE, lpData);
    }

    return 0;
}

CString BrowseFolder()
{
    TCHAR path[MAX_PATH] = { 0 };

    const char * path_param = "";

    BROWSEINFO bi = { 0 };
    bi.lpszTitle = _T("Вибір каталогу з програмою CashaLot");
    bi.ulFlags = BIF_RETURNONLYFSDIRS | BIF_NEWDIALOGSTYLE;
    bi.lpfn = BrowseCallbackProc;
    bi.lParam = (LPARAM)path_param;

    LPITEMIDLIST pidl = SHBrowseForFolder(&bi);

    if (pidl != 0)
    {
        //get the name of the folder and put it in path
        SHGetPathFromIDList(pidl, path);
```

```
        //free memory used
        IMalloc * imalloc = 0;
        if (SUCCEEDED(SHGetMalloc(&imalloc)))
        {
            imalloc->Free(pidl);
            imalloc->Release();
        }

        return CString(path);
    }

    return CString("");
}

void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedCheckusecashalotprintersettings()
{
    UpdateData();

    m_rdbtPrintUseDefPrinter.EnableWindow(!m_UseDefCashalotPrinter);
    m_rdbtPrintUseNamedPrinter.EnableWindow(!m_UseDefCashalotPrinter);
    m_edPrinterName.EnableWindow(!m_UseDefCashalotPrinter);
}


void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedRadio2()
{
    UpdateData();
    m_rdbtPrintUseDefPrinter.SetCheck(m_PrintUseDefWindowsPrinter == 0);
    m_rdbtPrintUseNamedPrinter.SetCheck(m_PrintUseDefWindowsPrinter == -1);
}


void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedRadio1()
{
    UpdateData();
    m_rdbtPrintUseDefPrinter.SetCheck(m_PrintUseDefWindowsPrinter == -1);
    m_rdbtPrintUseNamedPrinter.SetCheck(m_PrintUseDefWindowsPrinter == 0);
}


void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtgetprintername()
{
    CPrintDialog dlg(true, PD_NOPAGENUMS | PD_HIDEPRINTTOFILE | PD_NOCURRENTPAGE |
PD_NONETWORKBUTTON);
    if (dlg.DoModal() == IDOK)
    {
        m_PrinterName = dlg.GetDeviceName();
        UpdateData(0);
    }
}

void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtgetpath()
{
    CString ret = BrowseFolder();
    if (ret.GetLength())
    {
        m_PathToCashaLot = ret;
        m_edPathToCashaLot.SetWindowText(m_PathToCashaLot);
    }
}
////////////////ПРИКЛАД ВИКОРИСТАННЯ//////////////////////////////
```

```
struct CashaLotApiRsp
{
    void Clear()
    {
        Ret = false;
        ErrorString = "";
        Values.clear();
    }

    CashaLotApiRsp()
    {
        Clear();
    }
    bool Ret ;
    std::string ErrorString ;
    std::list< std::pair<std::string, std::string> > Values;
};

CashaLotApiRsp rspEx;
CashaLotApiRsp DeserializeResp(std::string response)
{
    rspEx.Clear();
    return rspEx;
}

//Структура інформації по позиції товару в чеці(артикул, назва сумма, кількість, податки і тд.)
struct ReceiptItemInfo
{
    ReceiptItemInfo()
    {
        VendorCode = "";
        Name = "";
        CdUKDZED = "";
        CdDKPP = "";
        Barcode = "";
        GoodsType = 1; //1- товар, 2- услуга
        UnitType = "шт";
        Quantity = 0;
        Price = 0;
        Amount = 0;
        DiscountPrc = 0;
        DiscountSum = 0;
        VATRate = 0;
        IsPriceIncludeVAT = true;
        SumVAT = 0;
        IsExcise = false;
        OtherParametrs = "";
    }

    std::string VendorCode;
    std::string Name;
    std::string CdUKDZED;
    std::string CdDKPP;
    std::string Barcode;
    int GoodsType; //1- товар, 2- услуга
    std::string UnitType;
    double Quantity;
    double Price;
    double Amount;
    double DiscountPrc;
```

```
    double DiscountSum;
    double VATRate;
    bool   IsPriceIncludeVAT;
    double SumVAT;
    bool IsExcise;
    std::string OtherParametrs;

    //Повертає JSON рядок з інформацією по позиції товару
    std::string ToJSON()
    {
        std::ostringstream ret;

        ret << "{" <<
            "\"VendorCode\":\"" << (VendorCode.c_str()) <<
            "\",\"Name\":\"" << (Name.c_str()) <<
            "\",\"CdUKDZED\":\"" << (CdUKDZED.c_str()) <<
            "\",\"CdDKPP\":\"" << (CdDKPP.c_str()) <<
            "\",\"Barcode\":\"" << (Barcode.c_str()) <<
            "\",\"GoodsType\":\"" << GoodsType <<
            "\",\"UnitType\":\"" << (UnitType.c_str()) <<
            "\",\"Quantity\":\"" << Quantity <<
            "\",\"Price\":\"" << Price <<
            "\",\"Amount\":\"" << Amount <<
            "\",\"DiscountPrc\":\"" << DiscountPrc <<
            "\",\"DiscountSum\":\"" << DiscountSum <<
            "\",\"VATRate\":\"" << VATRate <<
            "\",\"IsPriceIncludeVAT\":\"" << ((!IsPriceIncludeVAT) ? "false" : "true") <<
            "\",\"SumVAT\":\"" << SumVAT <<
            "\",\"IsExcise\":\"" << ((!IsExcise)? "false":"true") <<
            "\",\"OtherParametrs\":\"" << (OtherParametrs.c_str()) << "\""
            << "}"
            ;

        return ret.str();
    }
};

struct ReceiptInfoLst
{
private:
    std::list<ReceiptItemInfo*>  lstReceiptItemsInfo;
    std::string Comment;
public:
    ReceiptInfoLst()
    {
        Clear();
    };
    ~ReceiptInfoLst()
    {
        Clear();
    };

    std::string ToJSON()
    {
        std::ostringstream ret;
        std::ostringstream comment;
        long cnt=0;

        if(Comment.length() > 0)
            comment << ",\"Comment\":\"" << Comment.c_str() <<"\"";

        ret << "{\"ReceiptLst\":[";
```

```
        for(std::list<ReceiptItemInfo*>::iterator x= lstReceiptItemsInfo.begin(); x!=
lstReceiptItemsInfo.end(); ++x)
        {
            if(cnt++!=0)
                ret << ",";
            ret << ((*x)->ToJSON());
        }
        ret <<  "]"<< (comment.str().c_str()) <<"}";

        return ret.str();
    }

    _bstr_t ToBSTRJSON()
    {
       return _bstr_t(ToJSON().c_str());;
    }
    void Clear()
    {
        Comment.clear();
        for (std::list<ReceiptItemInfo*>::iterator x = lstReceiptItemsInfo.begin(); x !=
lstReceiptItemsInfo.end(); ++x)
            delete (*x);
        lstReceiptItemsInfo.clear();
    }

    void InsertItem( ReceiptItemInfo* check )
    {
        lstReceiptItemsInfo.push_back(check);
    }

    void SetComment(std::string strComment)
    {
        Comment = strComment;
    }
};

//Структура оплати по чеку
struct PayInfoRsp
{
    //Сума готівки наданої покупцем(може бути більшою за сумму до сплати, з неї вираховується
решта/здача)
    double SumCash;
    //Сума оплати картою
    double SumPayByCard;
    //Сума оплати за рахунок кредитних коштів
    double SumPayByCredit;
    //Сума оплати сертифікатом
    double SumPayByCertificate;
    // Сума фактичної оплати всіма видами оплати, з урахуванням заокруглення
    double SumPayCheck;
    // Email платника
    std::string CustomerEmail;

    PayInfoRsp()
    {
        //Сума готівки наданої покупцем(може бути більшою за сумму до сплати, з неї
вираховується решта/здача)
        SumCash =
        SumPayByCard =
        SumPayByCredit =
        SumPayByCertificate =
```

```
        SumPayCheck = 0;
        CustomerEmail = "";
    }

     PayInfoRsp()
//До структури оплати PayInfoRsp, додані поля:
       //Тип виду оплати (0-звичайна | 1 - передоплата | 2 - післяплата | 3 - інтернет продаж
)
            public int PaymentOrderType;
            //Сума попередньої оплати
            public String SumPreparePayed { get; set; }
            //Фіскальний номер чека по якому здійснюється доплата/повернення
            public String ParentReceiptFiscalNumber { get; set; }
            //Фіскальний номер РРО по якому здійснюється доплата/повернення
            public String ParentRROFiscalNumber { get; set; }
            // OtherParametrs - Додаткові параметри ХМЛ(зарезервовано - не використовується)
            public String OtherParametrs { get; set; }
```

В структуру информации о чеке LstCheckInfoRsp, добавлено поле:

```
            //Номер замовлення/договору
            public String SaleOrderNumber { get; set; } = String.Empty;

    //Отримати JSON рядок з інформацією про оплату по чеку(готівка, карта, знижки та
заокруглення)
    std::string ToJSON()
    {
        std::ostringstream ret;
        long cnt = 0;
        ret << "{\"SumCash\":\""<< SumCash <<
                "\",\"SumPayByCard\":\""<< SumPayByCard <<
                "\",\"SumPayByCredit\":\"" << SumPayByCredit <<
                "\",\"SumPayByCertificate\":\"" << SumPayByCertificate <<
                "\",\"SumPayCheck\":\"" << SumPayCheck <<
                "\",\"CustomerEmail\":\"" << CustomerEmail << "\"}";

        return ret.str();
    }

    _bstr_t ToBSTRJSON()
    {
        return _bstr_t(ToJSON().c_str());
    }
};

std::ostringstream strRet;
ReceiptInfoLst gsLstCheckItems;
ICashalotApiRetValPtr retVal = NULL;
ICashaLotApiAddinPtr gsCashaLotApi = NULL;
BSTR fiscalNumRRO = NULL;
BSTR pathToCashaLot = NULL;
BSTR pathToCert = NULL;
BSTR pwdToCert = NULL;
bool IsNoInterfaceMode = false;
bool useToken = false;
int  setupPrinterMode = 0;
BSTR printerName = NULL;

void SetPrinterParametr()
{
    if (gsCashaLotApi != NULL)
    {
```

```
        gsCashaLotApi->SetParameter( L"DEFAULTPRINTERNAME", (setupPrinterMode == 0)? BSTR(L"")
: ((setupPrinterMode == 1)? BSTR(L"DEFAULTWINDOWS") :  printerName) );
    }
}

//Ініціалізація COM об'єкта CashaLot
bool InitCashaLotApiComInterface()
{
    if ( gsCashaLotApi == NULL )
    {
        // Вказуємо ІД COM об'єкта AddIn.CashaLotApi, який має буди попередньо зареєстрований
в реєстрі(regsvr32 D:\prro\CashLot\CashaLotApi64.dll)
        char* szProgID = "AddIn.CashaLotApi";
        WCHAR  szWideProgID[128];
        CLSID  clsid;
        long lLen = MultiByteToWideChar(CP_ACP,
            0,
            szProgID,
            strlen(szProgID),
            szWideProgID,
            sizeof(szWideProgID));

        szWideProgID[lLen] = '\0';
        HRESULT hr = ::CLSIDFromProgID(szWideProgID, &clsid);
        if (FAILED(hr))
        {
            MessageBox(0, L"Unable to get CLSID from ProgID. AddIn.CashaLotApi", 0, 0);
            return false;
        }

        IClassFactory* pCF;
        // Отримати фабрику класів
        hr = CoGetClassObject(clsid,
            CLSCTX_INPROC,
            NULL,
            IID_IClassFactory,
            (void**)&pCF);
        if (FAILED(hr))
        {
            MessageBox(0, L"Failed to GetClassObject server instance.", 0, 0);
            return false;
        }

        // за допомогою фабрики класів створити екземпляр
        // компонента та отримати інтерфейс IUnknown.
        IUnknown* pUnk;
        hr = pCF->CreateInstance(NULL, IID_IUnknown, (void**)&pUnk);

        // Release the class factory
        pCF->Release();

        if (FAILED(hr))
        {
            MessageBox(0, L"Failed to create server instance.", 0, 0);
            return false;
        }

        hr = pUnk->QueryInterface(__uuidof(ICashaLotApiAddin), (LPVOID*)&gsCashaLotApi);
        pUnk->Release();

        if (FAILED(hr))
        {
```

```
            MessageBox(0, L"Failed to create ICashaLotApiAddin", 0, 0);
            return false;
        }

        //-----Задаємо обов'язкові параметри(шлях до каталогу з програмою Кашалот, та фіскальний номер РРО з якою будемо працювати) ----------------------------

        gsCashaLotApi->SetParameter(L"PathToCashalotDir", pathToCashaLot);
        gsCashaLotApi->SetParameter(L"DeviceIDFnRRO", fiscalNumRRO);
        gsCashaLotApi->SetParameter(L"NOINTERFACEMODE", IsNoInterfaceMode ? L"true" : L"false");
        gsCashaLotApi->SetParameter(L"NOAUTOOPENSHIFT", L"true");
        //Ознака використанні захищеного носія(має бути після PathToCashalotDir та DeviceIDFnRRO та NOINTERFACEMODE, якщо необхідно пришвидшити завантаження, та немає необхідності використовувати токени. Інакше токени будуть ініціалізуватися)
        gsCashaLotApi->SetParameter(L"USETOKEN", useToken ? L"true" : L"false");

        SetPrinterParametr();

        gsCashaLotApi->SetParameter(L"PathToCertificate", pathToCert);
        gsCashaLotApi->SetParameter(L"PwdToCertificate", pwdToCert);

    }

    return true;
}

bool CSampleUseCashaLotCOMAPICPPDlg::InitCashaLotApiComInterfaceEx()
{
    UpdateData();

    if (useToken != m_useToken && gsCashaLotApi != NULL)
        gsCashaLotApi->SetParameter(L"USETOKEN", m_useToken ? L"true" : L"false");

    useToken = m_useToken;

    if ( pathToCashaLot == NULL )
    {
        pathToCashaLot = ::SysAllocString(m_PathToCashaLot);
    }

    if (fiscalNumRRO == NULL )
    {
        fiscalNumRRO = ::SysAllocString(m_FiscalNumberRRO);
    }

    if (pathToCert != NULL && pathToCert != m_PathCert)
    {
        ::SysFreeString(pathToCert);
        pathToCert = ::SysAllocString(m_PathCert);
        if (gsCashaLotApi != NULL)
            gsCashaLotApi->SetParameter(L"PathToCertificate", pathToCert);
    }
    else
    if (pathToCert == NULL )
    {
        pathToCert = ::SysAllocString(m_PathCert);
    }

    if (pwdToCert != NULL && pwdToCert != m_pwdCert)
    {
        ::SysFreeString(pwdToCert);
```

```
        pwdToCert = ::SysAllocString(m_pwdCert);
        if (gsCashaLotApi != NULL)
            gsCashaLotApi->SetParameter(L"PwdToCertificate", pwdToCert);
    }
    else
    if (pwdToCert == NULL)
    {
        pwdToCert = ::SysAllocString(m_pwdCert);
    }
    int oldSetupPrinterMode = setupPrinterMode;
    setupPrinterMode = (m_UseDefCashalotPrinter)? 0 : m_PrintUseDefWindowsPrinter!=-1? 1 : 2 ;

    if (printerName != NULL && (printerName != m_PrinterName || oldSetupPrinterMode !=
setupPrinterMode) )
    {
        ::SysFreeString(printerName);
        printerName = ::SysAllocString(m_PrinterName);
        SetPrinterParametr();
    }
    else
    if(printerName == NULL)
        printerName = ::SysAllocString(m_PrinterName);

    if(IsNoInterfaceMode != m_IsNoInterfaceMode && gsCashaLotApi != NULL )
        gsCashaLotApi->SetParameter(L"NOINTERFACEMODE", m_IsNoInterfaceMode ? L"true" :
L"false");

    IsNoInterfaceMode = m_IsNoInterfaceMode;



    return InitCashaLotApiComInterface();
}

//Очистка всіх ресурсів по роботі з COM об'єктом CashaLot
bool TermCashaLotApiComInterface()
{
    try
    {
        if (gsCashaLotApi != NULL)
        {
            gsCashaLotApi->Release();
            gsCashaLotApi = NULL;

            if (pathToCashaLot != NULL)
            {
                ::SysFreeString(pathToCashaLot);
                pathToCashaLot = NULL;
            }
            if (fiscalNumRRO != NULL)
            {
                ::SysFreeString(fiscalNumRRO);
                fiscalNumRRO = NULL;
            }
            if (pathToCert != NULL)
            {
                ::SysFreeString(pathToCert);
                pathToCert = NULL;
            }
            if (pwdToCert != NULL)
            {
                ::SysFreeString(pwdToCert);
```

```
                pwdToCert = NULL;
            }
            return true;
        }
    }
    catch (const char* msg)
    {
        MessageBox(0, BSTR(msg), L"Exception", MB_OK);
    }
    catch (...)
    {
        MessageBox(0, L"", L"Exception", MB_OK);
    }

    return false;
}

//Приклад відкриття зміни
void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtopenshift()
{
    if(!InitCashaLotApiComInterfaceEx())
    return;

    gsCashaLotApi->SetParameter(L"NOINTERFACEMODE", IsNoInterfaceMode ? L"true" : L"false");

    if (pathToCert != NULL)
    {
        ::SysFreeString(pathToCert);
        pathToCert = NULL;
    }
    if (pwdToCert != NULL)
    {
        ::SysFreeString(pwdToCert);
        pwdToCert = NULL;
    }

    pathToCert = ::SysAllocString(m_PathCert);
    pwdToCert = ::SysAllocString(m_pwdCert);

    gsCashaLotApi->SetParameter(L"PathToCertificate", pathToCert);
    gsCashaLotApi->SetParameter(L"PwdToCertificate", pwdToCert);

    _variant_t ret = gsCashaLotApi->OpenShift(fiscalNumRRO);

    if (V_VT(&ret) == VT_DISPATCH)
    {
        retVal = (ICashalotApiRetVal*)ret.pdispVal;
        if (retVal != NULL && retVal->Return != FALSE)
        {
            m_edPathToCashaLot.EnableWindow(false);
            m_btnPath.EnableWindow(false);
            m_edFiscalNumberRRO.EnableWindow(false);

            strRet.str("");
            strRet << "Номер відкритої зміни: " << retVal->ShiftID;
            MessageBox(CString(strRet.str().c_str()));
        }
    }
}

//Приклад закриття зміни
void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtcloseshift()
```

```
{
    if (!InitCashaLotApiComInterfaceEx())
        return;

    _variant_t ret = gsCashaLotApi->CloseShift(fiscalNumRRO);

    if (V_VT(&ret) == VT_DISPATCH)
    {
        retVal = (ICashalotApiRetVal*)ret.pdispVal;
        if (retVal != NULL && retVal->Return != FALSE)
        {
            strRet.str("");
            strRet << "Номер закритої зміни: " << retVal->ShiftID<<"\n"
                   << "Результат у вигляді JSON\n" << retVal->JsonVal;
            MessageBox(CString(strRet.str().c_str()));
        }
    }
}

//Приклад фіскалізації чека на дві позиції
void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtfiscalizereceipt()
{
    if (!InitCashaLotApiComInterfaceEx())
        return;

    gsLstCheckItems.Clear();

    ReceiptItemInfo* inf = new ReceiptItemInfo();
    inf->VendorCode = "АртикулТовару1";
    inf->Name = "НазваТовару1";
    inf->Barcode = "123456789";
    inf->UnitType = "шт";
    inf->Quantity = 1;
    inf->GoodsType = 1;/*товар*/
    inf->Price = 183;
    inf->Amount = 181.17;
    inf->DiscountSum = 1.83;
    inf->VATRate = 20;
    inf->SumVAT = 30.2;
    inf->IsExcise = false;
    gsLstCheckItems.InsertItem(inf);

    inf = new ReceiptItemInfo();
    inf->VendorCode = "АртикулПослуги1";
    inf->Name = "НазваПослуги1";
    inf->Barcode = "923456780";
    inf->UnitType = "шт";
    inf->Quantity = 1;
    inf->GoodsType = 2;/*послуга*/
    inf->Price = 183;
    inf->Amount = 181.17;
    inf->DiscountSum = 1.83;
    inf->VATRate = 20;
    inf->SumVAT = 30.2;
    inf->IsExcise = false;
    gsLstCheckItems.InsertItem( inf );
    gsLstCheckItems.SetComment("Тестовий рядок1\r\nТестовий рядок2\r\nТестовий рядок3");

    _bstr_t jsonChkData = gsLstCheckItems.ToBSTRJSON();

    PayInfoRsp payInfo;
    payInfo.SumCash = 366;
```

```
    payInfo.SumPayCheck = 362.34;
    payInfo.SumPayByCredit = 100;
    payInfo.SumPayByCertificate = 50;
    _bstr_t jsonPayData = payInfo.ToBSTRJSON();

    _variant_t ret = gsCashaLotApi->FiscalizeCheck(fiscalNumRRO, jsonChkData, jsonPayData);

    if (V_VT(&ret) == VT_DISPATCH)
    {
        retVal = (ICashalotApiRetVal*)ret.pdispVal;
        if (retVal != NULL && retVal->Return != FALSE)
        {
            m_edFiscNumRcp.SetWindowTextW(retVal->ReceiptFiscalNum);

            strRet.str("");
            strRet << "Фіскальний номер створеного чека: " << retVal->ReceiptFiscalNum;
            strRet << "\nXML створеного чека: \n" << retVal->XML;
            MessageBox(CString(strRet.str().c_str()));
        }
    }

    gsLstCheckItems.Clear();
}

//Приклад фіскалізації чека повернення на дві позиції, з вказанням чека по якому здійснюється повернення
void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtreturnreceipt()
{
    CString strfiscNumRcpToRet;
    m_edFiscNumRcp.GetWindowText(strfiscNumRcpToRet);

    if (strfiscNumRcpToRet.GetLength() == 0)
    {
        MessageBox(L"Не заданий фіскальний номер чека по якому здійснюється повернення!");
        return;
    }

    if (!InitCashaLotApiComInterfaceEx())
        return;

    gsLstCheckItems.Clear();
    ReceiptItemInfo* inf = new ReceiptItemInfo();
    inf->VendorCode = "АртикулТовару1";
    inf->Name = "НазваТовару1";
    inf->Barcode = "123456789";
    inf->UnitType = "шт";
    inf->Quantity = 1;
    inf->GoodsType = 1;/*товар*/
    inf->Price = 183;
    inf->Amount = 181.17;
    inf->DiscountSum = 1.83;
    inf->VATRate = 20;
    inf->SumVAT = 30.2;
    inf->IsExcise = false;
    gsLstCheckItems.InsertItem(inf);

    inf = new ReceiptItemInfo();
    inf->VendorCode = "АртикулПослуги1";
    inf->Name = "НазваПослуги1";
    inf->Barcode = "923456780";
    inf->UnitType = "шт";
    inf->Quantity = 1;
```

```
    inf->GoodsType = 2;/*послуга*/
    inf->Price = 183;
    inf->Amount = 181.17;
    inf->DiscountSum = 1.83;
    inf->VATRate = 20;
    inf->SumVAT = 30.2;
    inf->IsExcise = false;
    gsLstCheckItems.InsertItem(inf);
    _bstr_t jsonChkData = gsLstCheckItems.ToBSTRJSON();

    PayInfoRsp payInfo;
    payInfo.SumCash = 366;
    payInfo.SumPayCheck = 362.34;
    _bstr_t jsonPayData = payInfo.ToBSTRJSON();

    _bstr_t retFiscNum = ::SysAllocString(strfiscNumRcpToRet);

    _variant_t ret = gsCashaLotApi->FiscalizeReturnCheck(fiscalNumRRO, jsonChkData,
jsonPayData, retFiscNum );

    if ( V_VT(&ret) == VT_DISPATCH )
    {
        retVal = (ICashalotApiRetVal*)ret.pdispVal;
        if (retVal != NULL && retVal->Return != FALSE)
        {
            strRet.str("");
            strRet << "Фіскальний номер створеного видаткового чека: " <<
retVal->ReceiptFiscalNum;
            //strRet << "\nXML створеного видаткового чека: \n" << retVal->XML;
            strRet << "\nТип чека: \n" << retVal->Type;
            MessageBox(CString(strRet.str().c_str()));
        }
    }

    ::SysFreeString(retFiscNum);
}

       //Приклад проведення чека ПЕРЕДОПЛАТИ
        private void buttonPrepayment_Click( object sender, EventArgs e )
        {

            LstCheckInfoRsp lstGoodsInCheck = new LstCheckInfoRsp();
            lstGoodsInCheck.ReceiptLst = new List<CheckInfo>();

            //Заповнюємо дві позиції товарів в чек
            CheckInfo chInfo = new CheckInfo();// структура з параметрами першої позиції
товару
            chInfo.Name = "ТоварПредоплатыПослеплаты1";
            chInfo.UnitType = "шт";
            chInfo.Quantity = "2";
            chInfo.Price = "1000";
            chInfo.Amount = "2000";
            //додати рядок чека з товаром №1
            lstGoodsInCheck.ReceiptLst.Add( chInfo );

            CheckInfo chInfo2 = new CheckInfo( chInfo );//Структура з параметрами другої
позиції товару
            chInfo2.Name = "ТоварПредоплатыПослеплаты2";
            chInfo.Quantity = "1";
            chInfo.Price = "1000";
            chInfo.Amount = "1000";
            //додати рядок чека з товаром №2
```

```
            lstGoodsInCheck.ReceiptLst.Add( chInfo2 );

            //присвоїти номер договору/замовлення передоплати(друкується в чеці)
            lstGoodsInCheck.SaleOrderNumber = "555444333";

            //Заповнюємо інформацію про оплату
            PayInfoRsp payInfo = new PayInfoRsp();
            payInfo.SumCash = "1500"; //Сума готівки наданої покупцем( може бути більшою за
суму до сплати, з неї вираховується решта/здача)

            //встановлюємо порядок оплати "ПЕРЕДОПЛАТА"
            payInfo.PaymentOrderType = 1; //Тип виду сплати (0-звичайна | 1 - передоплата | 2
- післяплата | 3 - інтернет продаж )
            //вказуємо суму передоплати
            payInfo.SumPreparePayed = "1500";

            //вказуємо загальну суму по чеку
            payInfo.SumPayCheck = "1500";//Сума оплати загальна з урахуванням заокруглення

            //Конвертація структур до формату джейсон строки
            string strJSONData = JsonConvert.SerializeObject( lstGoodsInCheck );
            string payDataJSON = JsonConvert.SerializeObject( payInfo );

            //Параметри функції
            Object[] parameters =
            {
                edFiscalNumberRRO.Text.Trim(), // фіскальний номер РРО
                strJSONData, // структура інформації зі строками чека
                payDataJSON // структура інформації про оплату по чеку
            };

            var ret = CallCashaLotMethod( "FiscalizeCheck", parameters );
            if ( ret?.Ret != true )
            {
                MessageBox.Show( String.IsNullOrEmpty( ret?.ErrorString ) ? "Не вдалося
фіскалізувати чек" : ret?.ErrorString );
            }
            else if ( ret?.Values != null )
            {
                //отримання параметра в результаті виконання функції
                var vals = ret.Values.ToArray();
                //перебираемо параметри
                for ( int i = 0; i < vals.Length; i++ )
                {
                    // отримуємо значення по ключу
                    if ( vals[i].Key.ToLower() == "receiptlocalnumber" )
                    {
                        edLocalCheckNum.Text = vals[i].Value;
                    }
                    else
                    if ( vals[i].Key.ToLower() == "receiptfiscalnumber" )
                    {
                        //Отримуємо фіскальний номер створенного чека ПЕРЕДОПЛАТИ, та
зберігаємо його для можливості здійснити "післяплату"
                        edFNRcpPrepayment.Text = vals[i].Value;
                    }
                }
                MessageBox.Show( $"Чек було успішно фіскалізовано.\n" +
                                 $"Чеку присвоєно локальний номер:
{edLocalCheckNum.Text},\n" +
                                 $"та фіскальний номер: {edFiscalCheckNum.Text} "
                                 );
```

```
            }
        }

//Приклад проведення чека "ПІСЛЯПЛАТИ"
        private void buttonAfterpayment_Click( object sender, EventArgs e )
        {
            LstCheckInfoRsp lstGoodsInCheck = new LstCheckInfoRsp();
            lstGoodsInCheck.ReceiptLst = new List<CheckInfo>();

            //присвоїти номер договору/замовлення передоплати(друкується в чеці)
            lstGoodsInCheck.SaleOrderNumber = "555444333";

            PayInfoRsp payInfo = new PayInfoRsp();
            payInfo.SumCash = "1600"; //Сума готівки наданої покупцем( може бути більшою за
суму до сплати, з неї вираховується решта/здача)

            //встановлюємо порядок оплати "ПІСЛЯПЛАТА"
            payInfo.PaymentOrderType = 2; //Тип виду сплати (0-звичайна | 1 - передоплата | 2
- післяплата | 3 - інтернет продаж )

            //Вказуємо фіскальний номер чека "ПЕРЕДОПЛАТИ", на основі якого буде
післяплата(поля чека будуть автоматично зачитані з нього, передані користувачем будуть
проігноровані)
            payInfo.ParentReceiptFiscalNumber = edFNRcpPrepayment.Text;
            //вказуємо суму передоплати
            payInfo.SumPreparePayed = "1500";
            //вказуємо загальну суму по чеку
            payInfo.SumPayCheck = "1500";//Сума оплати загальна з урахуванням заокруглення

            //Конвертація структур до формату джейсон строки
            string strJSONData = JsonConvert.SerializeObject( lstGoodsInCheck );
            string payDataJSON = JsonConvert.SerializeObject( payInfo );

            //Параметри функції
            Object[] parameters =
            {
                edFiscalNumberRRO.Text.Trim(), // фіскальний номер РРО
                strJSONData, // структура інформації зі строками чека
                payDataJSON // структура інформації про оплату по чеку
            };

            var ret = CallCashaLotMethod( "FiscalizeCheck", parameters );
            if ( ret?.Ret != true )
            {
                MessageBox.Show( String.IsNullOrEmpty( ret?.ErrorString ) ? "Не вдалося
фіскалізувати чек" : ret?.ErrorString );
            }
            else if ( ret?.Values != null )
            {
                //отримання параметра в результаті виконання функції
                var vals = ret.Values.ToArray();
                //перебираемо параметри
                for ( int i = 0; i < vals.Length; i++ )
                {
                    // отримуємо значення по ключу
                    if ( vals[i].Key.ToLower() == "receiptlocalnumber" )
                    {
                        edLocalCheckNum.Text = vals[i].Value;
                    }
                    else
                    if ( vals[i].Key.ToLower() == "receiptfiscalnumber" )
                    {
```

```
                    edFiscalCheckNum.Text = vals[i].Value;
                }
            }
            MessageBox.Show( $"Чек було успішно фіскалізовано.\n" +
                                $"Чеку присвоєно локальний номер:
{edLocalCheckNum.Text},\n" +
                                $"та фіскальний номер: {edFiscalCheckNum.Text} "
                                );
        }
    }


    //Приклад проведення чека ПЕРЕДОПЛАТИ
    private void buttonPrepayment_Click( object sender, EventArgs e )
    {

        LstCheckInfoRsp lstGoodsInCheck = new LstCheckInfoRsp();
        lstGoodsInCheck.ReceiptLst = new List<CheckInfo>();

        //Заповнюємо дві позиції товарів в чек
        CheckInfo chInfo = new CheckInfo();// структура з параметрами першої позиції
товару
        chInfo.Name = "ТоварПредоплатыПослеплаты1";
        chInfo.UnitType = "шт";
        chInfo.Quantity = "2";
        chInfo.Price = "1000";
        chInfo.Amount = "2000";
        //додати рядок чека з товаром №1
        lstGoodsInCheck.ReceiptLst.Add( chInfo );

        CheckInfo chInfo2 = new CheckInfo( chInfo );//Структура з параметрами другої
позиції товару
        chInfo2.Name = "ТоварПредоплатыПослеплаты2";
        chInfo.Quantity = "1";
        chInfo.Price = "1000";
        chInfo.Amount = "1000";
        //додати рядок чека з товаром №2
        lstGoodsInCheck.ReceiptLst.Add( chInfo2 );

        //присвоїти номер договору/замовлення передоплати(друкується в чеці)
        lstGoodsInCheck.SaleOrderNumber = "555444333";

        //Заповнюємо інформацію про оплату
        PayInfoRsp payInfo = new PayInfoRsp();
        payInfo.SumCash = "1500"; //Сума готівки наданої покупцем( може бути більшою за
сумму до сплати, з неї вираховується решта/здача)

        //встановлюємо порядок оплати "ПЕРЕДОПЛАТА"
        payInfo.PaymentOrderType = 1; //Тип виду оплати (0-звичайна | 1 - передоплата | 2
- післяплата | 3 - інтернет продаж )
        //вказуємо суму передоплати
        payInfo.SumPreparePayed = "1500";

        //вказуємо загальну суму по чеку
        payInfo.SumPayCheck = "1500";//Сума оплати загальна з урахуванням заокруглення

        //Конвертація структур до формату джейсон строки
        string strJSONData = JsonConvert.SerializeObject( lstGoodsInCheck );
        string payDataJSON = JsonConvert.SerializeObject( payInfo );

        //Параметри функції
        Object[] parameters =
```

```
            {
                edFiscalNumberRRO.Text.Trim(), // фіскальний номер РРО
                strJSONData, // структура інформації зі строками чека
                payDataJSON // структура інформації про оплату по чеку
            };

            var ret = CallCashaLotMethod( "FiscalizeCheck", parameters );
            if ( ret?.Ret != true )
            {
                MessageBox.Show( String.IsNullOrEmpty( ret?.ErrorString ) ? "Не вдалося
фіскалізувати чек" : ret?.ErrorString );
            }
            else if ( ret?.Values != null )
            {
                //отримання параметра в результаті виконання функції
                var vals = ret.Values.ToArray();
                //перебираемо параметри
                for ( int i = 0; i < vals.Length; i++ )
                {
                    // отримуємо значення по ключу
                    if ( vals[i].Key.ToLower() == "receiptlocalnumber" )
                    {
                        edLocalCheckNum.Text = vals[i].Value;
                    }
                    else
                    if ( vals[i].Key.ToLower() == "receiptfiscalnumber" )
                    {
                        //Отримуємо фіскальний номер створенного чека ПЕРЕДОПЛАТИ, та
зберігаємо його для можливості здійснити "післяплату"
                        edFNRcpPrepayment.Text = vals[i].Value;
                    }
                }
                MessageBox.Show( $"Чек було успішно фіскалізовано.\n" +
                                 $"Чеку присвоєно локальний номер:
{edLocalCheckNum.Text},\n" +
                                 $"та фіскальний номер: {edFiscalCheckNum.Text} "
                                 );
            }
        }

        //Приклад проведення чека "ПІСЛЯПЛАТИ"
        private void buttonAfterpayment_Click( object sender, EventArgs e )
        {
            LstCheckInfoRsp lstGoodsInCheck = new LstCheckInfoRsp();
            lstGoodsInCheck.ReceiptLst = new List<CheckInfo>();

            //присвоїти номер договору/замовлення передоплати(друкується в чеці)
            lstGoodsInCheck.SaleOrderNumber = "555444333";

            PayInfoRsp payInfo = new PayInfoRsp();
            payInfo.SumCash = "1600"; //Сума готівки наданої покупцем( може бути більшою за
сумму до сплати, з неї вираховується решта/здача)

            //встановлюємо порядок оплати "ПІСЛЯПЛАТА"
            payInfo.PaymentOrderType = 2; //Тип виду оплати (0-звичайна | 1 - передоплата | 2
- післяплата | 3 - інтернет продаж )

            //Вказуємо фіскальний номер чека "ПЕРЕДОПЛАТИ", на основі якого буде
післяплата(поля чека будуть автоматично зачитані з нього, передані користувачем будуть
проігноровані)
            payInfo.ParentReceiptFiscalNumber = edFNRcpPrepayment.Text;
            //вказуємо сумму передоплати
```

```
            payInfo.SumPreparePayed = "1500";
            //вказуємо загальну суму по чеку
            payInfo.SumPayCheck = "1500";//Сума оплати загальна з урахуванням заокруглення

            //Конвертація структур до формату джейсон строки
            string strJSONData = JsonConvert.SerializeObject( lstGoodsInCheck );
            string payDataJSON = JsonConvert.SerializeObject( payInfo );

            //Параметри функції
            Object[] parameters =
            {
                edFiscalNumberRRO.Text.Trim(), // фіскальний номер РРО
                strJSONData, // структура інформації зі строками чека
                payDataJSON // структура інформації про оплату по чеку
            };

            var ret = CallCashaLotMethod( "FiscalizeCheck", parameters );
            if ( ret?.Ret != true )
            {
                MessageBox.Show( String.IsNullOrEmpty( ret?.ErrorString ) ? "Не вдалося
фіскалізувати чек" : ret?.ErrorString );
            }
            else if ( ret?.Values != null )
            {
                //отримання параметра в результаті виконання функції
                var vals = ret.Values.ToArray();
                //перебираемо параметри
                for ( int i = 0; i < vals.Length; i++ )
                {
                    // отримуємо значення по ключу
                    if ( vals[i].Key.ToLower() == "receiptlocalnumber" )
                    {
                        edLocalCheckNum.Text = vals[i].Value;
                    }
                    else
                    if ( vals[i].Key.ToLower() == "receiptfiscalnumber" )
                    {
                        edFiscalCheckNum.Text = vals[i].Value;
                    }
                }
                MessageBox.Show( $"Чек було успішно фіскалізовано.\n" +
                                 $"Чеку присвоєно локальний номер:
{edLocalCheckNum.Text},\n" +
                                 $"та фіскальний номер: {edFiscalCheckNum.Text} "
                                 );
            }
        }

        //Приклад чека "Інтернет-продаж"
        private void btnInternetPayment_Click(object sender, EventArgs e)
        {
            LstCheckInfoRsp lstGoodsInCheck = new LstCheckInfoRsp();
            lstGoodsInCheck.ReceiptLst = new List<CheckInfo>();

            //Заповнюємо дві позиції товарів в чек
            CheckInfo chInfo = new CheckInfo();// структура з параметрами першої позиції
товару

            chInfo.Name = "ТоварІнтернетПродажу1";
            chInfo.UnitType = "шт";
            chInfo.Quantity = "2";
            chInfo.Price = "1000";
            chInfo.VATRate = "20";
```

```
            chInfo.Amount = "2000";

            //додати рядок чека з товаром №1
            lstGoodsInCheck.ReceiptLst.Add(chInfo);

            CheckInfo chInfo2 = new CheckInfo(chInfo);// структура з параметрами другої позиції товару
            chInfo2.Name = "ТоварІнтернетПродажу2";
            chInfo.Quantity = "1";
            chInfo.Price = "1000";
            chInfo.Amount = "1000";

            //додати рядок чека з товаром №2
            lstGoodsInCheck.ReceiptLst.Add(chInfo2);

            //присвоїти номер договору/замовлення (друкується в чеці)
            lstGoodsInCheck.SaleOrderNumber = "555444333555";

            PayInfoRsp payInfo = new PayInfoRsp();
            //вказуємо порядок оплати "Інтернет-продаж"
            payInfo.PaymentOrderType = 3; //Тип виду сплати (0-звичайна | 1 - передоплата | 2 - післяплата | 3 - інтернет-продаж )

            payInfo.SumPayByCard = "1500"; //Сума оплати картою або на рахунок
            //сума передоплати, якщо вона менша за суму всіх позицій чека то решта додається в рядок ПІСЛЯПЛАТА
            payInfo.SumPreparePayed = "1500";

            payInfo.SumPayCheck = "3000";//Сума оплати по чеку загальна

            //Конвертація структур до формату джейсон строки
            string strJSONData = JsonConvert.SerializeObject(lstGoodsInCheck);
            string payDataJSON = JsonConvert.SerializeObject(payInfo);

            //Параметри функції
            Object[] parameters =
            {
                edFiscalNumberRRO.Text.Trim(), // фіскальний номер РРО
                strJSONData, // структура інформації зі строками чека
                payDataJSON // структура інформації про оплату по чеку
            };

            var ret = CallCashaLotMethod("FiscalizeCheck", parameters);
            if (ret?.Ret != true)
            {
                MessageBox.Show(String.IsNullOrEmpty(ret?.ErrorString) ? "Не вдалося фіскалізувати чек" : ret?.ErrorString);
            }
            else if (ret?.Values != null)
            {
                //отримання параметра в результаті виконання функції
                var vals = ret.Values.ToArray();
                //перебираемо параметри
                for (int i = 0; i < vals.Length; i++)
                {
                    // отримуємо значення по ключу
                    if (vals[i].Key.ToLower() == "receiptlocalnumber")
                    {
                        edLocalCheckNum.Text = vals[i].Value;
                    }
                    else
                    if (vals[i].Key.ToLower() == "receiptfiscalnumber")
```

```
                    {
                        edFNRcpPrepayment.Text = vals[i].Value;
                    }
                }
                MessageBox.Show($"Чек було успішно фіскалізовано.\n" +
                                $"Чеку присвоєно локальний номер:
{edLocalCheckNum.Text},\n" +
                                $"та фіскальний номер: {edFiscalCheckNum.Text} "
                                );
            }
        }


//Приклад чека службового внесення
void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtserviceinput()
{
    if (!InitCashaLotApiComInterfaceEx())
        return;
    BSTR strSm = ::SysAllocString(L"100");

    VARIANT vSm;
    ::VariantInit(&vSm);
    V_VT(&vSm) = VT_BSTR;
    V_BSTR(&vSm) = strSm;

    _variant_t ret = gsCashaLotApi->ServiceInput(fiscalNumRRO, vSm);

    if (V_VT(&ret) == VT_BOOL && V_BOOL(&ret)!=FALSE )
        MessageBox(L"Службове внесення виконано успішно");
    else
        MessageBox(L"Не вдалося виконати службове внесення");

    ::SysFreeString(strSm);
}

//Приклад чека службової видачі
void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtserviceoutput()
{
    if (!InitCashaLotApiComInterfaceEx())
        return;
    BSTR strSm = ::SysAllocString(L"100");

    VARIANT vSm;
    ::VariantInit(&vSm);
    V_VT(&vSm) = VT_BSTR;
    V_BSTR(&vSm) = strSm;
    _variant_t ret = gsCashaLotApi->ServiceOutput(fiscalNumRRO, vSm);

    if (V_VT(&ret) == VT_BOOL && V_BOOL(&ret) != FALSE)
        MessageBox(L"Службова видача виконана успішно");
    else
        MessageBox(L"Не вдалося виконати службову видачу");

    ::SysFreeString(strSm);
}

//Приклад чека друку X-Звіту(без гасіння)
void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedPrintXRep()
{
    if (!InitCashaLotApiComInterfaceEx())
        return;
```

```
    _variant_t ret = gsCashaLotApi->PrintXReport(fiscalNumRRO);

    if (V_VT(&ret) == VT_BOOL && V_BOOL(&ret) != FALSE)
        MessageBox(L"X-звіт сформований успішно");
    else
        MessageBox(L"Не вдалося сформувати X-звіт");
}


void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtgetpathtocert()
{
    CString ret = BrowseFolder();
    if (ret.GetLength())
    {
        m_PathCert = ret;
        m_edPathCert.SetWindowText(m_PathCert);
    }
}


void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtprintxrepex()
{
    if (!InitCashaLotApiComInterfaceEx())
        return;

    _variant_t ret = gsCashaLotApi->GetXReport(fiscalNumRRO, VARIANT_FALSE );

    if (V_VT(&ret) == VT_DISPATCH)
    {
        retVal = (ICashalotApiRetVal*)ret.pdispVal;
        if (retVal != NULL && retVal->Return != FALSE)
        {
            strRet.str("");
            strRet << "Результат у вигляді JSON:\n" << retVal->JsonVal;
            MessageBox(CString(strRet.str().c_str()));
        }
    }
}

void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedBtprintlastzrep()
{
    if (!InitCashaLotApiComInterfaceEx())
        return;

    _variant_t ret = gsCashaLotApi->GetLastZReport(fiscalNumRRO, VARIANT_TRUE);

    if (V_VT(&ret) == VT_DISPATCH)
    {
        retVal = (ICashalotApiRetVal*)ret.pdispVal;
        if (retVal != NULL && retVal->Return != FALSE)
        {
            strRet.str("");
            strRet << "Результат у вигляді JSON:\n" << retVal->JsonVal;
            MessageBox(CString(strRet.str().c_str()));
        }
    }
}

void CSampleUseCashaLotCOMAPICPPDlg::OnBnClickedGetRcpByPeriod()
{
    if (!InitCashaLotApiComInterfaceEx())
        return;
```

```
strRet.str("");
BSTR dtBeg = ::SysAllocString(m_DateBeg.Format(CString("%d.%m.%Y")));
BSTR dtEnd = ::SysAllocString(m_DateEnd.Format(CString("%d.%m.%Y")));
//зачитуємо перший чек за період
_variant_t ret = gsCashaLotApi->GetFirstReceiptByPeriod(fiscalNumRRO, dtBeg, dtEnd);

if (V_VT(&ret) == VT_DISPATCH)
{
    retVal = (ICashalotApiRetVal*)ret.pdispVal;
    if (retVal != NULL && retVal->Return != FALSE)
    {
        strRet << "Фіскальні номери чеків за період:\n";
        strRet << retVal->ReceiptFiscalNum;

        bool retN = false;
        do
        {   retN = false;
            //зачитуємо наступний чек за період
            ret = gsCashaLotApi->GetNextReceipt(fiscalNumRRO);
            if (V_VT(&ret) == VT_DISPATCH)
            {
                retVal = (ICashalotApiRetVal*)ret.pdispVal;
                if (retVal != NULL && retVal->Return != FALSE)
                {
                    retN = true;
                    strRet<< ",";
                    strRet << retVal->ReceiptFiscalNum;
                }
            }
            //читаємо поки будуть наступні
        } while (retN);
    }
}

ret = gsCashaLotApi->GetFirstZReportByPeriod(fiscalNumRRO, dtBeg, dtEnd);

if (V_VT(&ret) == VT_DISPATCH)
{
    retVal = (ICashalotApiRetVal*)ret.pdispVal;
    if (retVal != NULL && retVal->Return != FALSE)
    {
        strRet << "\nФіскальні номери Z-звітів за період:\n";
        strRet << retVal->ReceiptFiscalNum;

        bool retN = false;
        do
        {
            retN = false;
            //зачитуємо наступний чек за період
            ret = gsCashaLotApi->GetNextZReport(fiscalNumRRO);
            if (V_VT(&ret) == VT_DISPATCH)
            {
                retVal = (ICashalotApiRetVal*)ret.pdispVal;
                if (retVal != NULL && retVal->Return != FALSE)
                {
                    retN = true;
                    strRet << ",";
                    strRet << retVal->ReceiptFiscalNum;
                }
            }
            //читаємо поки будуть наступні
```

```
            } while (retN);
        }
    }

    MessageBox(CString(strRet.str().c_str()));

    ::SysFreeString(dtBeg);
    ::SysFreeString(dtEnd);

}
```