



ООО “АСТРОН ЛТД”

**Пакет прикладных программ
PowerPOS Версия 5.0**

**Создание драйверов
сервиса интеграции**

Минск – 2015

ООО "Астрон ЛТД"

220113, г. Минск, ул. Мележа 5, корп. 2 оф. 1201

тел. +375 (17) 392-56-00, 01, 02, 03, 04, 05

факс +375 (17) 392-56-06

<http://www.astron.by>

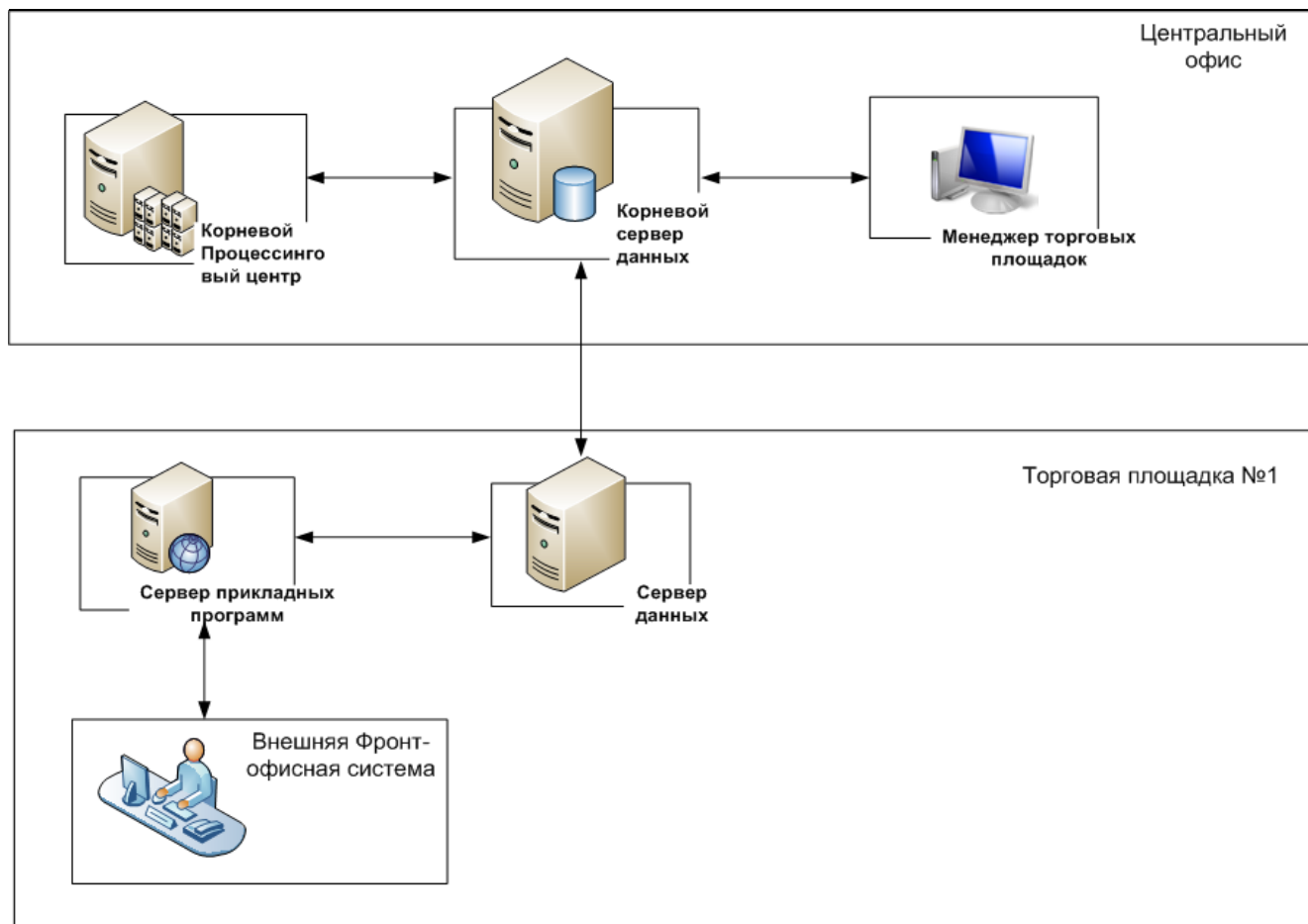
© Copyright Astron Ltd., 2014. All right reserved.

Данная публикация или ее часть не могут быть воспроизведены в любой форме без предварительного письменного разрешения фирмы Астрон ЛТД.

Сервис интеграции.

Назначение

Сервис интеграции предназначен для обеспечения обмена данными между некоторой внешней фронт-офисной (ФО) системой и сервисами PowerPOS.



Описание

Сервис интеграции представляет собой плагин для СПП PowerPOS, с открытым исполнительным модулем (драйвером), который реализуется с учетом особенностей конкретной ФО системы. В качестве базы данных используется база СПП (MobileDataServer), данные в которую попадают в результате штатной синхронизации между сервером данных и СПП.

Использование сервиса интеграции дает возможность использовать стороннюю ФО систему:

- в единой системе централизованного управления, учета и мониторинга PowerPOS;
- в системе централизованного мониторинга при децентрализованном учете;
- в системе лояльности PowerPOS (насколько это позволяет сама ФО система).

Драйвер представляет собой библиотеку, реализующую определенный интерфейс. Принцип его работы – запуск циклического потока, периодически выполняющего определенные действия (передачу справочников во внешнюю ФО систему и/или получение данных о продажах).

Создание драйвера для сервиса интеграции

Драйвер сервиса интеграции является наследником интерфейса ***IIIntegrationService*** описанного в библиотеке ***IntegrationServiceCommon*** (находится в каталоге установки PowerPOS, в случае, если установлен сервер прикладных программ). Там же реализован базовый класс ***IntegrationSettingsBase***, требуемый для одного из свойств интерфейса ***IIIntegrationService***, а также ряд вспомогательных классов, описанных ниже.

Описание интерфейса ***IIIntegrationService***.

Свойства интерфейса:

String ServiceID - возвращает уникальный идентификатор конкретного драйвера, по сути является GUID-ом.

String ServiceFriendlyName – наименование драйвера с учетом локализации, под этим именем драйвер будет доступен в списке драйверов сервиса интеграции.

String ServiceDescription – описание драйвера – произвольный текст, который будет отображен в поле «Описание драйвера» в настройках сервиса интеграции.

Boolean IsRunning – флаг состояния потока драйвера.

IntegrationSettingsBase SettingsProvider – класс, наследник ***IntegrationSettingsBase*** с настройками драйвера сервиса интеграции.

Методы интерфейса:

void Start() – запуск потока драйвера.

void Stop() – остановка потока драйвера.

Описание базового класса настроек ***IntegrationSettingsBase***.

Для работы необходимо создать свой класс, наследник от ***IntegrationSettingsBase***, где реализовать абстрактную ф-цию *createDefaultSettings()*, заполняющую настройками по умолчанию переменную *protected List<IntegrationSettingsRow> defaultSettingsList* (эти настройки будут установлены при первом запуске).

Свойства класса:

List<IntegrationSettingsRow> SettingsList – лист с настройками.

List<IntegrationSettingsRow> DefaultSettingsList – лист с настройками по умолчанию.

Методы класса:

SettingsValue getSettings(String settingsKey) – возвращает значение настройки по ее ключу.

Описание класса ***IntegrationSettingsRow***.

Свойства класса:

String SettingsKey – ключевое слово, идентифицирующее настройку.

String SettingsValue – значение настройки, преобразованное к строке.

String SettingsDescription – описание настройки, некий текст, который доступен в настройках сервиса интеграции, применительно к данной настройке (например «имя пользователя», «использовать да/нет» и т.д.).

Описание класса ***SettingsValue***.

Свойства класса:

String Value – текстовое представление значения настройки.

int IntValue – целое представление значения настройки.

bool BoolValue – булево представление значения настройки.

bool IsIntValue – если целое представление настройки определено – true, иначе false.

bool IsBoolValue – если булево представление настройки определено – true, иначе false.

Преобразование строки происходит при помощи стандартной ф-ции *Convert.ToInt32(value)* и *Convert.Boolean(value)*. Соответствующий флаг выставляется в false в случае возникновения исключения при преобразовании.

Для упрощения разработки драйвера сервиса интеграции можно использовать базовый класс ***IntegrationServiceBase***.

Этот класс уже является наследником ***IntegrationService***. В нем реализован собственно поток, управляемый методами *Start* и *Stop*, а выполняемая функция является виртуальной и называется *ServiceProc()* (т.е. достаточно просто в наследнике написать реализацию этой ф-ции). Все исключения этой функции перехватываются и логируются с использованием штатной системы логирования (т.е. для того, чтобы что-либо логировалось достаточно сгенерировать соответствующее исключение).

Для работы с базой СПП в базовом классе доступен специальный объект *protected IntegrationBaseAccessor iBaseAccessor*, который автоматически открывает подключение к базе СПП на основе настроек подключения самого СПП и обеспечивает автовосстановление этого подключения. Объект типа *IntegrationBaseAccessor* может быть создан и без использования базового класса, непосредственно при реализации интерфейса.

Для формирования чека PowerPOS с расчетом скидок и бонусов можно использовать класс ***IntegrationReceipt***.

Описание класса *IntegrationReceipt*.

Конструктор:

IntegrationReceipt(int sareId, int systemId, int workDayId, int sessId, int cashprofileId, int cashierId) - параметры конструктора соответственно: номер торговой площадки, номер системы, номер рабочего дня, номер сессии, идентификатор профиля кассы, идентификатор кассира.

При выборе профиля для виртуальной кассы, следует ориентироваться на стандартный профиль для киосков.

Свойства класса:

enum paymentTypes – перечисление, типы оплат.

bool IsOpen – признак того, что чек открыт.

int SareId – номер торговой площадки, для которой открыт чек.

int SystemId – номер системы, для которой открыт чек.

int SessId – номер сессии, для которой открыт чек.

int SrecNum – системный номер текущего чека.

decimal TotalDiscont – сумма скидок на чек в копейках.

decimal ReceiptSum – сумма чека с учетом всех скидок в копейках.

Методы чека:

void OpenReceipt(paymentTypes pType) – открывает новый чек, с указанным типом.

void AddDiscountCard(String code) – устанавливает в чеке дисконтную карточку.

void AddReceiptItem(int packId, decimal quantity) – добавляет в чек новую товарную позицию с кодом упаковки «packId» и количеством «quantity».

void AddReceiptItem(int packId, decimal quantity, decimal price) – добавляет в чек новую товарную позицию с кодом упаковки «packId», количеством «quantity» и ценой товарной позиции «price».

void RemoveReceiptItem(int artCode) – удаляет из фактуры соответствующую товарную позицию.

void CloseReceipt() – оплачивает чек.

void CloseReceipt(decimal paimentSum) – оплачивает чек суммой, полученной от клиента «paimentSum» в копейках.

void CanselReceipt() – отменяет чек.

В случае ошибок при работе с чеком, генерируются исключения с описанием возникших проблем.