|  |
| --- |
| УТВЕРЖДЕН |
|  |

ДОПОЛНЕНИЯ (ПЛАГИНЫ) ЕДИНОЙ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ ОТДЕЛЕНИЙ ПОЧТОВОЙ СВЯЗИ

РУКОВОДСТВО ДЛЯ РАЗРАБОТЧИКА

Версия 1.10

2015 г.

Содержание

[1 Введение 3](#_Toc417465192)

[1.1 Цель документа 3](#_Toc417465193)

[1.2 Глоссарий и сокращения 3](#_Toc417465194)

[1.3 Целевая аудитория 3](#_Toc417465195)

[1.4 ЕАС ОПС 3](#_Toc417465196)

[1.5 Вне рамок документа 3](#_Toc417465197)

[2 Платформа Встраивания дополнений 4](#_Toc417465198)

[2.1 Обзор 4](#_Toc417465199)

[2.2 Технология и Компоненты программного обеспечения 4](#_Toc417465200)

[3 Перечень требований к реализации дополнения 5](#_Toc417465201)

[3.1 Общие требования 5](#_Toc417465202)

[3.1.1 Язык программирования реализации 5](#_Toc417465203)

[3.1.2 Пользовательский интерфейс 5](#_Toc417465204)

[3.1.3 Безопасность 5](#_Toc417465205)

[3.1.4 Требования к программной реализации 5](#_Toc417465206)

[3.1.5 Прочие 7](#_Toc417465207)

[3.2 Особые требования к реализации интерфейсов 7](#_Toc417465208)

[3.2.1 Инфраструктура платежей 7](#_Toc417465209)

[4 Состав предоставляемых сборок платформы встраивания 8](#_Toc417465210)

[4.1 Интерфейсы 8](#_Toc417465211)

[4.1.1 Базовые интерфейсы 8](#_Toc417465212)

[4.1.2 Аппаратные интерфейсы 9](#_Toc417465213)

[4.1.3 Интерфейсы доступа к данным 11](#_Toc417465214)

[4.1.4 Интерфейсы для работы с планировщиком заданий 11](#_Toc417465215)

[4.1.5 Интерфейсы инфраструктуры платежей 12](#_Toc417465216)

[4.1.6 Интерфейсы для взаимодействия с сервером Валидата 13](#_Toc417465217)

[4.1.7 Интерфейсы инфраструктуры оформления заказов билетов 14](#_Toc417465218)

[4.2 Атрибуты 15](#_Toc417465219)

[4.2.1 Атрибуты декларирования 15](#_Toc417465220)

[4.2.2 Атрибуты предоставленных интерфейсов 15](#_Toc417465221)

[4.3 Типы 16](#_Toc417465222)

[4.3.1 Вспомогательные типы 16](#_Toc417465223)

[4.3.2 Типы для работы с планировщиком заданий 22](#_Toc417465224)

[4.3.3 Аппаратные типы 24](#_Toc417465225)

[4.3.4 Типы инфраструктуры платежей 26](#_Toc417465226)

[4.3.5 Типы для взаимодействия с сервером Валидата 37](#_Toc417465227)

[4.3.6 Типы инфраструктуры оформления заказов билетов 38](#_Toc417465228)

[5 Перечень изменений в платформе относительно предыдущей версии 1.9 1](#_Toc417465229)

# Введение

## Цель документа

Настоящий документ содержит информацию для партнеров ФГУП “Почта России”, достаточную для разработки собственного дополнения к единой автоматизированной системе отделений почтовой связи (ЕАС ОПС).

Документ является неотъемлемой частью концепции встраивания дополнений в ЕАС ОПС.

## Глоссарий и сокращения

**ОПС** – отделение почтовой связи.

**Партнер** – сторонний поставщик Почты России (СППР).

**Дополнение** – отдельное приложение-расширение ЕАС ОПС СППР, по-другому, независимо [компилируемый](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) [программный модуль](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), динамически подключаемый к основной [программе](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) кассового решения и предназначенный для расширения и использования её возможностей.

## Целевая аудитория

Целевой аудиторией настоящего документа являются разработчики партнеров, отвечающие за создание удаленных приложений, работающих с ЕАС ОПС.

## ЕАС ОПС

ЕАС ОПС – единая автоматизированная система отделений почтовой связи. Единое решение для автоматизации деятельности отделений почтовой связи, обеспечивающее поддержку функций “фронт-энд” и “бэк-офиса” и поддерживающее обмен данными с внешними системами.

## Вне рамок документа

В настоящем документе не описано:

- дизайн плагина

- бизнес-правила обработки операций внутри плагина

- обмен с внешними шлюзом или другими удаленными сервисами Партнера.

# Платформа Встраивания дополнений

## Обзор

Единая система автоматизации отделений почтовой связи (ЕАС ОПС) включает в себя платформу для встраивания дополнений, расширяющих функциональность системы. В рамках данной платформы согласованность действий дополнения и ЕАС ОПС базируется на использовании двух **программных интерфейсов взаимодействия**, сформированных и задекларированных дополнением согласно регламенту (описанному далее в данном документе):

* **Интерфейс взаимодействия дополнения** - интерфейс, предоставленный дополнением к использованию из ЕАС ОПС
* **Интерфейс взаимодействия c ЕАС ОПС** -интерфейс, описывающий требуемую дополнением функциональность от ЕАС ОПС.

Таким образом, разработчику дополнения следует реализовать собственную бизнес-логику, основываясь на том, что взаимодействие с ЕАС ОПС построено на синхронных вызовах методов, содержащихся в задекларированных дополнением интерфейсах.

## Технология и Компоненты программного обеспечения

На схеме показана связь ЕАС ОПС и дополнения партнера.



Как видно из схемы, взаимодействие дополнения с ЕАС ОПС осуществляется исключительно через программный компонент платформы встраивания, которая накладывает ряд регламентных требований к программной реализации дополнений.

# Перечень требований к реализации дополнения

## Общие требования

### Язык программирования реализации

Выбор языка программирования должен быть основан на необходимости поддержки им программной платформы .NET Framework. Причем для разработки должна использоваться версия .NET Framework 4.0.

### Пользовательский интерфейс

При наличии у дополнения пользовательского интерфейса, его разработка должна соответствовать требованиям к пользовательскому интерфейсу на АРМ ОПС ФГУП “Почта России”.

### Безопасность

При необходимости обеспечения конфиденциальности передаваемых данных, контроля целостности и подтверждения авторства следует использовать систему управления сертификатами АПК «Валидата УЦ», требуемые компоненты которой установлены в ОПС.

### Требования к программной реализации

Данный раздел содержит требования к программной реализации дополнения в последовательности, соответствующей его разработке. Примеры написаны на языке программирования C#.

#### Объявление интерфейсов взаимодействия

##### Интерфейс взаимодействия с ЕАС ОПС

Объявляется интерфейс, соответствующий ожидаемой дополнением функциональности от ЕАС ОПС.

Объявленный интерфейс должен:

* Наследоваться от одного или нескольких доступных интерфейсов платформы встраивания, помеченных атрибутом HostSideAttribute
* Быть пустым

public interface IHostInterface:IScalesAccess

{

}

##### Интерфейс взаимодействия дополнения

Объявляется интерфейс, который предоставляет дополнение для взаимодействия с ЕАС ОПС.

Интерфейс должен:

* Наследоваться от базового интерфейса IBasePlugin<THost>
* Наследоваться, при необходимости, от одного или нескольких доступных интерфейсов платформы встраивания, помеченных атрибутом PluginSideAttribute
* Быть пустым

public interface IMyPlugin : IBasePlugin<IHostInterface>, IScannerConsumer

{

}

#### Реализация класса дополнения

Класс дополнения должен:

* Реализовывать интерфейс взаимодействия дополнения
* Иметь публичный конструктор без параметров в случае его явной реализации

public class MyPlugin:IMyPlugin

{

    public void Initialize(TIHost host)

    {

       //Метод должен при необходимости содержать код проверки и формирования требуемой для работы плагина инфраструктуры. Например, инициализация структуры БД, установка сервисов и т.п.

    }

    public void BarcodeReceived(string barcode)

    {

        Console.WriteLine(barcode);

    }

}

#### Декларирование

Для возможности подключения плагина к ЕАС ОПС через платформу встраивания, в одном из файлов проекта (например, в стандартном файле AssemblyInfo.cs) следует прописать атрибут, указывающий декларируемые интерфейс взаимодействия дополнения и его реализацию. Атрибут должен храниться в сборке с расширением dll.

[assembly: PlugInExport(typeof(IMyPlugin), typeof(MyPlugin))]

Также должны быть заполнены стандартные атрибуты, позволяющие получить информацию о плагине и его разработчике.

[assembly: AssemblyDescription("Описание плагина")]

[assembly: AssemblyCompany("GMCS")]

### Прочие

Распространяемый плагин должен:

* Представлять из себя единственный архив, содержащий каталог плагина с файлами, требуемыми ему для работы
* Быть готовым к работе с интеграционной платформой после распаковки архива (без какого-либо дополнительного редактирования настроек в конфигурационных файлах или через собственный интерфейс)
* Не регистрировать службы
* Работать без необходимости запуска от имени администратора
* Хранить временные файлы, редактируемые плагином в подкаталоге по пути *…\AppData\Local\* текущего пользователя (собственный каталог плагина будет доступен только для чтения)

## Особые требования к реализации интерфейсов

### Инфраструктура платежей

#### Интерфейс IPluginPaymentService

* **GetVendorsList** - при реализации метода следует учитывать, что в силу особенностей архитектуры системы ЕАС ОПС он будет использоваться для загрузки списка поставщиков в систему без предварительного вызова метода плагина **Initialize**(). По той же причине на момент вызова не будут доступны компоненты системы управления сертификатами АПК «Валидата УЦ».

#### Интерфейс ISchedulerAccess

* **GetSchedulerTaskDeclarationList** - при реализации метода следует учитывать, что в силу особенностей архитектуры системы ЕАС ОПС он будет использоваться для загрузки задач в систему без предварительного вызова метода плагина **Initialize**(). По той же причине на момент вызова не будут доступны компоненты системы управления сертификатами АПК «Валидата УЦ».

# Состав предоставляемых сборок платформы встраивания

Для возможности реализации дополнения разработчику предоставляется распространяемая сборка платформы встраивания GM\_PluginCommon.dll, в которой содержится перечень базовых типов и интерфейсов, необходимых для разработки.

## Интерфейсы

### Базовые интерфейсы

#### Интерфейс IBasePlugin<THost>

Базовый интерфейс, который должен быть реализован дополнением

namespace GM\_PluginCommon

{

/// <summary>

/// Базовый интерфейс плагина

/// </summary>

/// <typeparam name="TIHost">Ожидаемый интерфейс основного приложения</typeparam>

public interface IBasePlugin<in TIHost>

{

/// <summary>

/// Инициализация плагина (Метод должен при необходимости содержать код проверки и формирования требуемой для работы плагина инфраструктуры. Например, инициализация структуры БД, установка сервисов и т.п.)

/// </summary>

void Initialize(TIHost host);

}

}

#### Интерфейс IterminalInfo

Интерфейс предоставляет необходимую информацию о текущем рабочем окне.

namespace GM\_PluginCommon

{

/// <summary>

/// Интерфейс, предоставляющий информацию о рабочем окне

/// </summary>

[HostSide]

public interface ITerminalInfo

{

/// <summary>

/// Индекс ОПС

/// </summary>

/// <returns></returns>

string GetStoreId();

/// <summary>

/// Код рабочего окна

/// </summary>

/// <returns></returns>

string GetTerminalId();

/// <summary>

/// Имя оператора в чеке

/// </summary>

/// <returns></returns>

string GetOperatorName();

/// <summary>

/// Полное имя оператора

/// </summary>

/// <returns></returns>

string GetOperatorFullName();

/// <summary>

/// Код оператора

/// </summary>

/// <returns></returns>

string GetOperatorId();

}

}

### Аппаратные интерфейсы

Интерфейсы взаимодействия с аппаратной частью АРМ ОПС.

#### Интерфейс IFiscalPrinterAccess

Интерфейс, предоставляемый платформой встраивания для работы с фискальным принтером.

namespace GM\_PluginCommon.Hardware.FiscalPrinter

{

/// <summary>

/// Интерфейс для доступа к фискальному принтеру

/// </summary>

[HostSide]

public interface IFiscalPrinterAccess

{

/// <summary>

/// Признак наличия фискального принтера на рабочем месте оператора

/// </summary>

/// <returns></returns>

bool IsFiscalPrinterEnabled();

/// <summary>

/// Надпечатка разрешена

/// </summary>

/// <returns></returns>

bool IsOverPrintAllowed();

/// <summary>

/// Печать свободного текста на чеке (пречек)

/// </summary>

/// <param name="fiscalFreeLines"></param>

void PrintFiscalFree(List<FiscalFreeLine> fiscalFreeLines);

/// <summary>

/// Подкладная печать свободного текста

/// </summary>

/// <param name="fiscalFreeLines"></param>

void PrintPadFree(List<FiscalFreeLine> fiscalFreeLines);

}

}

#### Интерфейс IScannerAccess

Интерфейс, предоставляемый платформой встраивания для получения информации о сканере.

namespace GM\_PluginCommon.Hardware

{

/// <summary>

/// Интерфейс для доступа к сканеру

/// </summary>

[HostSide]

public interface IScannerAccess

{

/// <summary>

/// Признак наличия сканера на рабочем месте оператора

/// </summary>

/// <returns></returns>

bool IsScannerEnabled();

/// <summary>

/// Запрос необходимости считывания ШК

/// </summary>

/// <returns></returns>

void BarcodeRequest();

}

#### }Интерфейс IScannerConsumer

Интерфейс, предоставляемый платформой встраивания для возможности обработки событий со сканера. Реализуется дополнением.

namespace GM\_PluginCommon.Hardware

{

/// <summary>

/// Интерфейс для обработки событий со сканера

/// </summary>

[PluginSide]

public interface IScannerConsumer

{

/// <summary>

/// Событие считывания со сканера

/// </summary>

/// <param name="barcode"></param>

void BarcodeReceived(string barcode);

}

}

#### Интерфейс IScalesAccess

Интерфейс, предоставляемый платформой встраивания для работы с весами.

namespace GM\_PluginCommon.Hardware

{

/// <summary>

/// Интерфейс для доступа к весу с весов

/// </summary>

[HostSide]

public interface IScalesAccess

{

/// <summary>

/// Получение текущего значения веса с весов

/// </summary>

/// <returns></returns>

decimal GetWeight();

/// <summary>

///Признак наличия весов на рабочем месте оператора

/// </summary>

/// <returns></returns>

bool IsScalesEnabled();

}

}

### Интерфейсы доступа к данным

#### Интерфейс IDBAccess

Интерфейс, предоставляемый платформой встраивания, предназначен для подключения плагина к предварительно созданной базе данных. «Владельцем» базы данных выступает плагин - составом и структурой таблиц он управляет сам.

namespace GM\_PluginCommon.DBAccess

{

/// <summary>

/// Интерфейс для доступа к базе данных плагина

/// </summary>

[HostSide]

public interface IDBAccess

{

/// <summary>

/// Получение строки подключения к базе данных

/// </summary>

/// <returns></returns>

string GetConnectionString();

}

}

### Интерфейсы для работы с планировщиком заданий

#### Интерфейс ISchedulerAccess

Интерфейс, предоставляемый платформой встраивания для взаимодействия дополнения с планировщиком заданий.

using System.Collections.Generic;

namespace GM\_PluginCommon.Scheduler

{

/// <summary>

/// Интерфейс для взаимодействия с планировщиком заданий

/// </summary>

[PluginSide]

public interface ISchedulerAccess

{

/// <summary>

/// запуск задания

/// </summary>

/// <returns></returns>

void RunTask(SchedulerTaskKey schedulerTaskKey);

/// <summary>

/// Получение списка задач

/// </summary>

/// <returns></returns>

List<SchedulerTaskDeclaration> GetSchedulerTaskDeclarationList();

}

}

### Интерфейсы инфраструктуры платежей

Интерфейсы, предоставляемые платформой встраивания для реализации дополнений приема платежей.

#### Интерфейс IPluginPaymentService

Интерфейс работы с платежами, реализуемый дополнением.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Интерфейс для реализации плагином, осуществляющим прием платежей

/// </summary>

[PluginSide]

public interface IPluginPaymentService

{

/// <summary>

/// Собственные данные агента

/// </summary>

AgentInfo AgentInfo();

/// <summary>

/// Поиск поставщиков по параметрам

/// </summary>

/// <param name="searchInfo">Заполненная структура, по реквизитам которой требуется производить поиск</param>

/// <returns>Список поставщиков, соответсвующих критериям поиска</returns>

List<SearchResult> FindVendor(SearchInfo searchInfo);

/// <summary>

/// Оформление платежей

/// </summary>

/// <param name="receivePaymentParameters">Параметры оформления платежей</param>

/// <returns>Список оформленных платежей</returns>

List<PaymentInfo> ReceivePayments(ReceivePaymentParameters receivePaymentParameters);

/// <summary>

/// Получение списка поставщиков

/// </summary>

/// <param name="postIndex">Индекс филиала</param>

/// <returns>Список поставщиков</returns>

List<VendorInfo> GetVendorsList(string postIndex);

/// <summary>

/// Оплата платежа

/// </summary>

/// <param name="paymentConfirmInfo">данные подтверждения платежа</param>

void PaymentConfirm(PaymentConfirmInfo paymentConfirmInfo);

/// <summary>

/// Отмена платежа

/// </summary>

/// <param name="paymentKeyInfo">Идентификатор платежа</param>

void PaymentCancel(PaymentKeyInfo paymentKeyInfo);

/// <summary>

/// Запрос возможности актирования платежа

/// </summary>

/// <param name="paymentKeyInfo"></param>

/// <returns></returns>

bool PaymentReturnRequest(PaymentKeyInfo paymentKeyInfo);

/// <summary>

/// Подтверждение актирования платежа

/// </summary>

/// <param name="paymentKeyInfo"></param>

/// <returns></returns>

bool PaymentReturnConfirm(PaymentKeyInfo paymentKeyInfo);

/// <summary>

/// Сверка платежей

/// </summary>

/// <param name="paymentSettlementInfo"></param>

/// <returns>Передача в плагин списка оформленных платежей</returns>

PaymentSettlementInfo Settle(PaymentSettlementInfo paymentSettlementInfo);

}

}

#### Интерфейс IPluginPaymentServiceHost

Интерфейс для работы с платежами, требуемый плагином от хоста.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Интерфейс, используемый плагином, оформляющем платежи

/// </summary>

[HostSide]

public interface IPluginPaymentServiceHost

{

/// <summary>

/// Список разрешенных поставщиков и связанных с ними услуг. Запрашивается плагином у хоста.

/// </summary>

/// <returns></returns>

List<VendorInfo> GetAllowedVendorList();

}

}

### Интерфейсы для взаимодействия с сервером Валидата

#### Интерфейс IValidataServerAccess

Интерфейс предоставляет возможность получить параметры доступа к серверу Валидата.

namespace GM\_PluginCommon.ValidataAccess

{

/// <summary>

/// Интерфейс для доступа к серверу Валидата

/// </summary>

[HostSide]

public interface IValidataServerAccess

{

/// <summary>

/// Параметры доступа к серверу

/// </summary>

ValidataServerAccessInfo GetServerAccessInfo();

/// <summary>

/// // Доступность сервера

/// </summary>

bool IsServerEnabled();

}

}

### Интерфейсы инфраструктуры оформления заказов билетов

#### Интерфейс IPluginTicketOrder

Интерфейс для оформления заказа билетов.

namespace GM\_PluginCommon.TicketOrder

{

/// <summary>

/// Интерфейс плагина оформления заказа билетов

/// </summary>

[PluginSide]

public interface IPluginTicketOrder

{

/// <summary>

/// Собственные данные агента

/// </summary>

Payments.AgentInfo AgentInfo();

/// <summary>

/// Оформление билетов

/// </summary>

TicketOrderInfo Purchase();

/// <summary>

/// Подтверждение платежа

/// </summary>

TicketOrderConfirmResult Confirm(TicketOrderConfirmInfo ticketOrderConfirmInfo);

/// <summary>

/// Аннулирование заказа

/// </summary>

void Cancel(TicketOrderKeyInfo ticketOrderKeyInfo);

/// <summary>

/// Запрос возможности возврата денег по заказу

/// </summary>

bool ReturnRequest(TicketOrderKeyInfo ticketOrderKeyInfo);

/// <summary>

/// Возврат денег по заказу

/// </summary>

bool ReturnConfirm(TicketOrderKeyInfo ticketOrderKeyInfo);

}

}

## Атрибуты

### Атрибуты декларирования

#### Атрибут PlugInExportAttribute

Атрибут, с помощью которого дополнение осуществляет свое декларирование платформе встраивания.

namespace GM\_PluginCommon

{

    [AttributeUsage(AttributeTargets.Assembly,AllowMultiple = false)]

    public class PlugInExportAttribute : Attribute

    {

        private readonly Type \_pluginImplementation;

        private readonly Type \_pluginInterface;

        /// <summary>

        /// </summary>

        /// <param name="pluginInterface">Интерфейс реализуемого плагина</param>

        /// <param name="pluginImplementation">Класс-реализация указанного интерфейса</param>

        public PlugInExportAttribute(Type pluginInterface, Type pluginImplementation)

        {

            \_pluginInterface = pluginInterface;

            \_pluginImplementation = pluginImplementation;

        }

        public Type PluginInterface

        {

            get { return \_pluginInterface; }

        }

        public Type PluginImplementation

        {

            get { return \_pluginImplementation; }

        }

    }

}

### Атрибуты предоставленных интерфейсов

Атрибуты, связанные с интерфейсами, предоставленными платформой встраивания.

#### Атрибут HostSideAttribute

Атрибут, с помощью которого обозначается интерфейс платформы встраивания, доступный для наследования интерфейсом взаимодействия с ЕАС ОПС.

namespace GM\_PluginCommon

{

[AttributeUsage(AttributeTargets.Interface)]

public class HostSideAttribute:Attribute

{

}

}

#### Атрибут PluginSideAttribute

Атрибут, с помощью которого обозначается интерфейс платформы встраивания, доступный для наследования интерфейсом взаимодействия дополнения.

namespace GM\_PluginCommon

{

[AttributeUsage(AttributeTargets.Interface)]

public class PluginSideAttribute:Attribute

{

}

}

## Типы

### Вспомогательные типы

#### Класс AddressInfo

Адресная информация.

namespace GM\_PluginCommon.Misc

{

/// <summary>

/// Адресная информация

/// </summary>

public class AddressInfo

{

/// <summary>

/// Индекс

/// </summary>

public string PostIndex { get; set; }

/// <summary>

/// Страна, цифровой код ISO

/// </summary>

public string Country { get; set; }

/// <summary>

/// Регион

/// </summary>

public string Region { get; set; }

/// <summary>

/// Район

/// </summary>

public string District { get; set; }

/// <summary>

/// Населенный пункт

/// </summary>

public string City { get; set; }

/// <summary>

/// Микрорайон

/// </summary>

public string CityDistrict { get; set; }

/// <summary>

/// Улица

/// </summary>

public string Street { get; set; }

/// <summary>

/// № здания

/// </summary>

public string BuildNum { get; set; }

/// <summary>

/// Литера

/// </summary>

public string BuildLetter { get; set; }

/// <summary>

/// Дробь

/// </summary>

public string BuildSlash { get; set; }

/// <summary>

/// Корпус

/// </summary>

public string BuildHousing { get; set; }

/// <summary>

/// Строение

/// </summary>

public string Building { get; set; }

/// <summary>

/// Наименование гостиницы

/// </summary>

public string BuildName { get; set; }

/// <summary>

/// Помещение

/// </summary>

public string Room { get; set; }

/// <summary>

/// Номер (воинской части и пр.)

/// </summary>

public string Number { get; set; }

}

}

#### Класс IdentityCardInfo

Документ удостоверяющий личность.

namespace GM\_PluginCommon.Misc

{

/// <summary>

/// Документ удостоверяющий личность

/// </summary>

public class IdentityCardInfo

{

/// <summary>

/// Тип документа

/// </summary>

public IdentityCardType IdentityCardType { get; set; }

/// <summary>

/// Серия

/// </summary>

public String Series { get; set; }

/// <summary>

/// Номер

/// </summary>

public String Number { get; set; }

/// <summary>

/// Кем выдан

/// </summary>

public String Issuer { get; set; }

/// <summary>

/// Код подразделения

/// </summary>

public String IssuerDepartment { get; set; }

/// <summary>

/// Дата выдачи

/// </summary>

public DateTime IssueDate { get; set; }

/// <summary>

/// Период действия С

/// </summary>

public DateTime DateFrom { get; set; }

/// <summary>

/// Период действия По

/// </summary>

public DateTime DateTo { get; set; }

/// <summary>

/// Дополнительная информация

/// </summary>

public String Extra { get; set; }

}

}

#### Класс MigratoryCardInfo

Миграционное удостоверение.

namespace GM\_PluginCommon.Misc

{

/// <summary>

/// Миграционное удостоверение

/// </summary>

public class MigratoryCardInfo

{

/// <summary>

/// Страна, цифровой код ISO

/// </summary>

public String Country { get; set; }

/// <summary>

/// Серия

/// </summary>

public String Series{ get; set; }

/// <summary>

/// Номер

/// </summary>

public String Number { get; set; }

/// <summary>

/// Дата выдачи

/// </summary>

public DateTime IssueDate { get; set; }

/// <summary>

/// Период пребывания С

/// </summary>

public DateTime DateFrom { get; set; }

/// <summary>

/// Период пребывания По

/// </summary>

public DateTime DateTo { get; set; }

/// <summary>

/// Дополнительная информация

/// </summary>

public String Extra { get; set; }

/// <summary>

/// Признак резидента

/// </summary>

public bool IsResident { get; set; }

}

}

#### Класс ContactInfoBase

Базовая контактная информация.

namespace GM\_PluginCommon.Misc

{

public class ContactInfoBase

{

/// <summary>

/// E-mail

/// </summary>

public String Email { get; set; }

/// <summary>

/// Телефон

/// </summary>

public List<String> PhoneNumber { get; set; }

/// <summary>

/// Факс

/// </summary>

public List<String> FaxNumber { get; set; }

}

}

#### Класс ContactInfo

Контактная информация.

namespace GM\_PluginCommon.Misc

{

/// <summary>

/// Контактная информация

/// </summary>

public class ContactInfo : ContactInfoBase

{

/// <summary>

/// Мобильный телефон

/// </summary>

public String MobilePhoneNumber { get; set; }

/// <summary>

/// Номер для отправки SMS

/// </summary>

public String SMSMobilePhoneNumber { get; set; }

/// <summary>

/// Дополнительная информация

/// </summary>

public String Extra { get; set; }

/// <summary>

/// Предпочительный способ контакта

/// </summary>

public ContactMethod ContactMethod { get; set; }

/// <summary>

/// Согласие на получение информации о новых сервисах и услугах

/// </summary>

public bool AdvertisingAgreement { get; set; }

}

}

#### Перечисление ContactMethod

Способы контакта.

namespace GM\_PluginCommon.Misc

{

public enum ContactMethod

{

/// <summary>

/// Почта

/// </summary>

Post=0,

/// <summary>

/// Телефон

/// </summary>

Phone=1,

/// <summary>

/// Мобильный телефон

/// </summary>

MobilePhone=2,

/// <summary>

/// Факс

/// </summary>

Fax=3,

/// <summary>

/// Электронная почта

/// </summary>

Email=4

}

}

#### Перечисление IdentityCardType

Типы документов, удостоверяющих личность.

namespace GM\_PluginCommon.Misc

{

public enum IdentityCardType

{

/// <summary>

/// Паспорт РФ

/// </summary>

Passport = 1,

/// <summary>

/// Паспорт иностранного гражданина

/// </summary>

PassportForeign = 2,

/// <summary>

/// Военный билет

/// </summary>

MilitaryTicket = 3,

/// <summary>

/// Паспорт моряка

/// </summary>

PassportSailor = 4,

/// <summary>

/// Временное удостоверение личности

/// </summary>

TempIdentityCard = 5,

/// <summary>

/// Удостоверение беженца

/// </summary>

RefugeeIdentityCard = 6,

/// <summary>

/// Вид на жительство

/// </summary>

ResidencePermit = 7,

/// <summary>

/// Справка гражданина, находящегося под стражей

/// </summary>

ArrestedPersonCertificate = 8,

/// <summary>

/// Удостоверение члена Совета Федерации РФ

/// </summary>

CouncilFederationMemberCertificate = 9,

/// <summary>

/// Удостоверение депутата Государственной Думы РФ

/// </summary>

StateDumaMemberCertificate = 10,

/// <summary>

/// Свидетельство о рассмотрении ходатайства о признании беженцем на территории РФ

/// </summary>

RefugeePetitionCertificate = 11,

/// <summary>

/// Разрешение на временное проживание

/// </summary>

TemporaryHabitationPermit = 12,

/// <summary>

/// Свидетельство о предоставлении временного убежища

/// </summary>

TemporaryRefugeCertificate = 13,

/// <summary>

/// Служебный паспорт РФ

/// </summary>

OfficialPassport = 14,

/// <summary>

/// Дипломатический паспорт гражданина РФ

/// </summary>

DiplomaticPassport = 15,

/// <summary>

/// Временное удостоверение (взамен военного билета)

/// </summary>

TempMilitaryTicket = 16,

/// <summary>

/// Удостоверение личности лица, проходящего военную службу

/// </summary>

MilitaryServiceIdentityCard = 17,

/// <summary>

/// Иной документ иностранного гражданина

/// </summary>

OtherForeignIdentityCard = 18,

}

}

### Типы для работы с планировщиком заданий

#### Класс SchedulerRecurrence

Класс конфигурирования повторений.

namespace GM\_PluginCommon.Scheduler

{

/// <summary>

/// Класс конфигурирования повторений

/// </summary>

public class SchedulerRecurrence

{

/// <summary>

/// Дата начала

/// </summary>

public DateTime StartDateTime { get; set; }

/// <summary>

/// Тип повторения

/// </summary>

public SchedulerRecurrenceType RecurrenceType { get; set; }

/// <summary>

/// Повторять каждые

/// </summary>

public int CountRecurrence { get; set; }

}

}

#### Класс SchedulerTaskKey

Класс, описывающий ключевые поля задания плагина.

namespace GM\_PluginCommon.Scheduler

{

/// <summary>

/// Идентификатор задания плагина

/// </summary>

public class SchedulerTaskKey

{

/// <summary>

/// Идентификатор задания плагина

/// </summary>

public String TaskId { get; set; }

}

}

#### Класс SchedulerTask

Класс, описывающий задание плагина.

using System;

namespace GM\_PluginCommon.Scheduler

{

/// <summary>

/// Задание плагина

/// </summary>

public class SchedulerTask : SchedulerTaskKey

{

/// <summary>

/// Описание задания плагина

/// </summary>

public String Description { get; set; }

}

}

#### Класс SchedulerTaskDeclaration

Класс декларации дополнением задания с указанием его повторения.

namespace GM\_PluginCommon.Scheduler

{

/// <summary>

/// Декларация задания плагина

/// </summary>

public class SchedulerTaskDeclaration

{

/// <summary>

/// Задания плагина

/// </summary>

public SchedulerTask SchedulerTask { get; set; }

/// <summary>

/// Конфигурация повторения

/// </summary>

public SchedulerRecurrence SchedulerRecurrence { get; set; }

}

}

#### Перечисление SchedulerRecurrenceType

Доступные типы повторений.

namespace GM\_PluginCommon.Scheduler

{

/// <summary>

/// Типы повторений

/// </summary>

public enum SchedulerRecurrenceType

{

/// <summary>

/// Минута

/// </summary>

Minutes = 0,

/// <summary>

/// Час

/// </summary>

Hours = 1,

/// <summary>

/// День

/// </summary>

Days = 2,

/// <summary>

/// Неделя

/// </summary>

Weeks = 3,

/// <summary>

/// Месяц

/// </summary>

Months = 4,

/// <summary>

/// Год

/// </summary>

Years = 5

}

}

### Аппаратные типы

#### Класс FiscalFont

Класс, позволяющий задать настройки шрифта фискального принтера при использовании нефискальной печати.

namespace GM\_PluginCommon.Hardware.FiscalPrinter

{

public class FiscalFont

{

/// <summary>

/// Альтернативный шрифт

/// </summary>

public bool IsAltFont { get; set; }

/// <summary>

/// Жирность

/// </summary>

public bool IsBold { get; set; }

/// <summary>

/// Символы удвоенной высоты

/// </summary>

public bool IsDblHeight { get; set; }

/// <summary>

/// Символы удвоенной ширины

/// </summary>

public bool IsDblWidth { get; set; }

/// <summary>

/// Символы с подчеркиванием

/// </summary>

public bool IsUnderline { get; set; }

}

}

#### Класс FiscalReceiptInfo

Информация о чеке.

using System;

namespace GM\_PluginCommon.Hardware.FiscalPrinter

{

/// <summary>

/// Информация о чеке

/// </summary>

public class FiscalReceiptInfo

{

/// <summary>

/// Номер чека

/// </summary>

public String Number { get; set; }

/// <summary>

/// Сумма

/// </summary>

public Decimal Amount { get; set; }

/// <summary>

/// Способ оплаты

/// </summary>

public PaymentMethod PaymentMethod { get; set; }

}

}

#### Класс FiscalFreeLine

Класс, позволяющий при использовании нефискальной печати формировать коллекцию строк с привязкой к требуемому шрифту и отправлять ее на печать.

namespace GM\_PluginCommon.Hardware.FiscalPrinter

{

public class FiscalFreeLine

{

/// <summary>

/// Шрифт

/// </summary>

public FiscalFont Font { get; set; }

/// <summary>

/// Текст

/// </summary>

public string Line { get; set; }

}

}

#### Перечисление PaymentMethod

Способ оплаты.

namespace GM\_PluginCommon.Hardware.FiscalPrinter

{

public enum PaymentMethod

{

/// <summary>

/// наличный

/// </summary>

Cash = 0,

/// <summary>

/// безналичный

/// </summary>

Сashless = 1

}

}

### Типы инфраструктуры платежей

#### Класс ClientInfo

Данные о клиенте.

using System;

using GM\_PluginCommon.Misc;

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Информация о клиенте

/// </summary>

public class ClientInfo

{

/// <summary>

/// Личный номер

/// </summary>

public string PersonalNumber { get; set; }

/// <summary>

/// Имя

/// </summary>

public string FirstName { get; set; }

/// <summary>

/// Отчество

/// </summary>

public string MiddleName { get; set; }

/// <summary>

/// Фамилия

/// </summary>

public string LastName { get; set; }

/// <summary>

/// Адрес фактический

/// </summary>

public AddressInfo AddressInfoFact { get; set; }

/// <summary>

/// Адрес регистрации

/// </summary>

public AddressInfo AddressInfoRegistration { get; set; }

/// <summary>

/// Дата рождения

/// </summary>

public DateTime BirthDate { get; set; }

/// <summary>

/// Место рождения

/// </summary>

public String BirthPlace { get; set; }

/// <summary>

/// ИНН

/// </summary>

public String INN { get; set; }

/// <summary>

/// Признак ИПДЛ

/// </summary>

public bool IsFPO { get; set; }

/// <summary>

/// Контактные данные

/// </summary>

public ContactInfo ContactInfo { get; set; }

/// <summary>

/// Документ удостоверяющий личность

/// </summary>

public IdentityCardInfo IdentityCardInfo { get; set; }

/// <summary>

/// Миграционное удостоверение

/// </summary>

public MigratoryCardInfo MigratoryCardInfo { get; set; }

/// <summary>

/// Гражданство. Указывается страна, цифровой код ISO

/// </summary>

public String Citizenship { get; set; }

}

}

#### Класс FederalClientInfo

Федеральный клиент.

using System;

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Федеральный клиент

/// </summary>

public class FederalClientInfo

{

/// <summary>

/// Идентификатор

/// </summary>

public String Id { get; set; }

/// <summary>

/// Наименование организации

/// </summary>

public String Name { get; set; }

/// <summary>

/// ОГРН

/// </summary>

public String OGRN { get; set; }

/// <summary>

/// ИНН

/// </summary>

public String INN { get; set; }

/// <summary>

/// КПП

/// </summary>

public String KPP { get; set; }

/// <summary>

/// Расчетный счет

/// </summary>

public String BankAccount { get; set; }

/// <summary>

/// Наименование банка

/// </summary>

public String BankName { get; set; }

/// <summary>

/// БИК

/// </summary>

public String BankBIK { get; set; }

/// <summary>

/// Корреспондентский счет

/// </summary>

public String BankCorrAccount { get; set; }

/// <summary>

/// КГП пункта выплаты

/// </summary>

public String PaymentKGPNumber { get; set; }

}

}

#### Класс VendorKeyInfo

Идентификатор поставщика.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Идентификатор поставщика

/// </summary>

public class VendorKeyInfo

{

/// <summary>

/// ИНН

/// </summary>

public string INN { get; set; }

/// <summary>

/// КПП

/// </summary>

public string KPP { get; set; }

/// <summary>

/// Наименование

/// </summary>

public string Name { get; set; }

}

}

#### Класс VendorInfo

Данные о поставщике.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Информация о поставщике

/// </summary>

public class VendorInfo : VendorKeyInfo

{

/// <summary>

/// Список доступных услуг в адрес поставщика

/// </summary>

public List<ServiceInfo> ServiceInfoList { get; set; }

/// <summary>

/// Адрес

/// </summary>

public String Address { get; set; }

/// <summary>

/// Номер телефона

/// </summary>

public String PhoneNumber { get; set; }

/// <summary>

/// Контрагент мастер-системы Почты России

/// </summary>

public MasterVendorInfo MasterVendorInfo { get; set; }

/// <summary>

/// Принадлежность поставщика почтамтам

/// </summary>

public List<String> PostIndexList { get; set; }

}

}

#### Класс ServiceKeyInfo

Идентификатор услуги агента.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Идентификатор услуги

/// </summary>

public class ServiceKeyInfo

{

/// <summary>

/// Код услуги

/// </summary>

public string ServiceId { get; set; }

}

}

#### Класс ServiceInfo

Данные об услуге агента.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Информация о услуге

/// </summary>

public class ServiceInfo : ServiceKeyInfo

{

/// <summary>

/// Описание

/// </summary>

public string Description { get; set; }

/// <summary>

/// Вид услуги

/// </summary>

public PaymentType PaymentType { get; set; }

}

}

#### Класс PaymentKeyInfo

Идентификатор платежа.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Идентификатор платежа

/// </summary>

public class PaymentKeyInfo

{

/// <summary>

/// Код платежа

/// </summary>

public string PaymentId { get; set; }

/// <summary>

/// Дата платежа

/// </summary>

public DateTime PaymentDate { get; set; }

}

}

#### Класс FiscalReceiptPaymentInfo

Дополнительная информация о чеке.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Информация о чеке при приеме платежа

/// </summary>

public class FiscalReceiptPaymentInfo

{

private int \_copyQty = 1;

/// <summary>

/// Содержимое чека в произвольной текстовой форме с адаптированным форматированием

/// </summary>

public string Text { get; set; }

/// <summary>

/// Количество копий

/// </summary>

public int CopyQty

{

get { return \_copyQty; }

set { \_copyQty = value; }

}

}

}

#### Класс PaymentInfo

Данные о платеже.

using System;

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Информация о платеже

/// </summary>

public class PaymentInfo : PaymentKeyInfo

{

/// <summary>

/// Информация о поставщике

/// </summary>

public VendorInfo VendorInfo { get; set; }

/// <summary>

/// Идентификатор услуги

/// </summary>

public ServiceKeyInfo ServiceKeyInfo { get; set; }

/// <summary>

/// Информация о клиенте

/// </summary>

public ClientInfo ClientInfo { get; set; }

/// <summary>

/// Описание

/// </summary>

public string Description { get; set; }

/// <summary>

/// Сумма платежа (в том числе сумма комиссии)

/// </summary>

public decimal Amount { get; set; }

/// <summary>

/// Сумма комиссии

/// </summary>

public decimal AmountCommission { get; set; }

/// <summary>

/// Родительский платеж

/// </summary>

public PaymentKeyInfo ParentPaymentKeyInfo { get; set; }

/// <summary>

/// Описание комиссии

/// </summary>

public String DescriptionCommission { get; set; }

/// <summary>

/// Данные о переводе

/// </summary>

public TransferPaymentInfo TransferPaymentInfo { get; set; }

/// <summary>

/// Дополнительная информация о печати фискального чека

/// </summary>

public FiscalReceiptPaymentInfo FiscalReceiptPaymentInfo { get; set; }

/// <summary>

/// Печать нефискального чека

/// </summary>

public FiscalReceiptPaymentInfo NoneFiscalReceiptPaymentInfo { get; set; }

}

}

#### Класс PaymentConfirmInfo

Данные подтверждения платежа.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Структура данных при подтверждении платежа

/// </summary>

public class PaymentConfirmInfo

{

/// <summary>

/// Идентификатор платежа

/// </summary>

public PaymentKeyInfo PaymentKeyInfo { get; set; }

/// <summary>

/// Коллекция чеков

/// </summary>

public List<FiscalReceiptInfo> FiscalReceiptInfo { get; set; }

/// <summary>

/// Информация о переводе

/// </summary>

public TransferInfo TransferInfo { get; set; }

}

}

#### Класс SearchInfo

Критерии подбора поставщиков.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Критерии подбора постащиков

/// </summary>

public class SearchInfo

{

/// <summary>

/// Данные клиента

/// </summary>

public ClientInfo ClientInfo { get; set; }

/// <summary>

/// Данные поставщика

/// </summary>

public VendorInfo VendorInfo { get; set; }

/// <summary>

/// Штрих-код

/// </summary>

public string BarCode { get; set; }

}

}

#### Класс SearchResult

Результат подбора поставщиков.

using System.Collections.Generic;

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Результат подбора поставщиков

/// </summary>

public class SearchResult

{

/// <summary>

/// Идентификатор поставщика

/// </summary>

public VendorKeyInfo VendorBaseInfo { get; set; }

/// <summary>

/// Список клиентов

/// </summary>

public List<ClientInfo> ClientInfoList { get; set; }

}

}

#### Класс RecievePaymentParameters

Параметры оформления платежей.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Параметры оформления платежей

/// </summary>

public class ReceivePaymentParameters

{

/// <summary>

/// Режим множественного оформления платежей

/// </summary>

public bool IsMultiMode { get; set; }

/// <summary>

/// Идентификатор поставщика

/// </summary>

public VendorKeyInfo VendorKeyInfo { get; set; }

/// <summary>

/// Информация для автозаполнения по данным поиска

/// </summary>

public SearchInfo SearchInfo { get; set; }

/// <summary>

/// Код услуги

/// </summary>

public ServiceKeyInfo ServiceKeyInfo { get; set; }

}

}

#### Класс TransferKeyInfo

Идентификатор перевода.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Ключевые поля перевода

/// </summary>

public class TransferKeyInfo

{

/// <summary>

/// Идентификатор перевода

/// </summary>

public String TransferNum { get; set; }

/// <summary>

/// Дата перевода

/// </summary>

public String TransferDate { get; set; }

/// <summary>

/// КГП пункта приема

/// </summary>

public String KGPNumber { get; set; }

}

}

#### Класс TransferInfo

Информация о переводе.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Информация о переводе

/// </summary>

public class TransferInfo : TransferKeyInfo

{

/// <summary>

/// Сумма перевода

/// </summary>

public decimal TransferValue { get; set; }

/// <summary>

/// Стоимость перевода

/// </summary>

public decimal TransferAmount { get; set; }

/// <summary>

/// Сумма тарифа за доп. услуги

/// </summary>

public decimal AddServiceAmount { get; set; }

/// <summary>

/// Сумма приема итого

/// </summary>

public decimal TransferTotalAmount{ get; set; }

}

}

#### Класс TransferPaymentInfo

Данные перевода.

using System;

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Данные перевода

/// </summary>

public class TransferPaymentInfo

{

/// <summary>

/// Федеральный клиент

/// </summary>

public FederalClientInfo FederalClientInfo { get; set; }

/// <summary>

/// Сообщение перевода

/// </summary>

public String TransferMessage { get; set; }

}

}

#### Класс AgentInfo

Информация об агенте.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Информация об агенте

/// </summary>

public class AgentInfo

{

/// <summary>

/// Наименование

/// </summary>

public string Name { get; set; }

/// <summary>

/// ИНН

/// </summary>

public string INN { get; set; }

/// <summary>

/// Адрес фактический

/// </summary>

public AddressInfo AddressInfoFact { get; set; }

/// <summary>

/// Адрес регистрации

/// </summary>

public AddressInfo AddressInfoRegistration { get; set; }

/// <summary>

/// Контактная информация

/// </summary>

public ContactInfoBase ContactInfo { get; set; }

}

}

#### Класс MasterVendorInfo

Контрагент мастер-системы.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Контрагент мастер-системы Почты России, отвечающей за НСИ

/// </summary>

public class MasterVendorInfo

{

/// <summary>

/// Идентификатор

/// </summary>

public string Id { get; set; }

}

}

#### Класс PaymentSettlementInfo

Структура данных для сверки платежей.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Структура данных при сверке платежей

/// </summary>

public class PaymentSettlementInfo

{

/// <summary>

/// Список платежей

/// </summary>

public List<PaymentConfirmInfo> PaymentConfirmInfoList { get; set; }

/// <summary>

/// Сверка на дату

/// </summary>

public DateTime SettlementDate { get; set; }

}

}

#### Перечисление PaymentType

Поддерживаемые ЕАС ОПС виды платежей.

namespace GM\_PluginCommon.Payments

{

/// <summary>

/// Виды платежа

/// </summary>

public enum PaymentType

{

/// <summary>

/// Коммунальные платежи

/// </summary>

Utility = 1,

/// <summary>

/// Сотовая связь

/// </summary>

Сellular = 2,

/// <summary>

/// Электросвязь

/// </summary>

Telecom = 3,

/// <summary>

/// Прочие

/// </summary>

Other = 4

}

}

### Типы для взаимодействия с сервером Валидата

#### Класс ValidataServerAccessInfo

Параметры доступа к серверу Валидата.

namespace GM\_PluginCommon.ValidataAccess

{

/// <summary>

/// Параметры доступа к серверу Валидата

/// </summary>

public class ValidataServerAccessInfo

{

/// <summary>

/// IP адрес

/// </summary>

public string IPAddress { get; set; }

/// <summary>

/// Имя сессии

/// </summary>

public string SessionName { get; set; }

/// <summary>

/// Порт

/// </summary>

public ushort Port { get; set; }

/// <summary>

/// Пароль

/// </summary>

public string Password { get; set; }

}

}

### Типы инфраструктуры оформления заказов билетов

#### Класс FeeInfo

Информация о сборе.

namespace GM\_PluginCommon.TicketOrder

{

/// <summary>

/// Информация о сборе

/// </summary>

public class FeeInfo

{

/// <summary>

/// Наименование сбора

/// </summary>

public string FeeName { get; set; }

/// <summary>

/// Сумма

/// </summary>

public decimal Amount { get; set; }

}

}

#### Класс TicketInfo

Информация о билете.

namespace GM\_PluginCommon.TicketOrder

{

/// <summary>

/// Информация о билете

/// </summary>

public class TicketInfo

{

/// <summary>

/// Тип билета

/// </summary>

public TicketType TicketType { get; set; }

/// <summary>

/// Номер билета

/// </summary>

public string Number { get; set; }

/// <summary>

/// Признак платности

/// </summary>

public bool IsComercial { get; set; }

/// <summary>

/// Сумма

/// </summary>

public decimal Amount { get; set; }

/// <summary>

/// Список сборов

/// </summary>

public List<FeeInfo> FeeInfoList { get; set; }

}

}

#### Класс TicketOrderKeyInfo

Идентификатор заказа.

namespace GM\_PluginCommon.TicketOrder

{

/// <summary>

/// Идентификатор заказа

/// </summary>

public class TicketOrderKeyInfo

{

/// <summary>

/// Идентификатор заказа

/// </summary>

public string Id { get; set; }

}

}

#### Класс TicketOrderInfo

Информация о заказе.

namespace GM\_PluginCommon.TicketOrder

{

/// <summary>

/// Информация о заказе

/// </summary>

public class TicketOrderInfo : TicketOrderKeyInfo

{

/// <summary>

/// Номер заказа

/// </summary>

public string OrderId { get; set; }

/// <summary>

/// Дата и время оформления заказа

/// </summary>

public DateTime OrderDateTime { get; set; }

/// <summary>

/// Дата и время истечения срока

/// </summary>

public DateTime ExpiredDateTime { get; set; }

/// <summary>

/// Номер транзакции

/// </summary>

public string TransactionId { get; set; }

/// <summary>

/// Список билетов

/// </summary>

public List<TicketInfo> TicketInfoList { get; set; }

/// <summary>

/// Список сборов по заказу в целом

/// </summary>

public List<FeeInfo> FeeInfoList { get; set; }

/// <summary>

/// Печать нефискального чека

/// </summary>

public Payments.FiscalReceiptPaymentInfo NoneFiscalReceiptPaymentInfo { get; set; }

}

#### Класс TicketOrderConfirmInfo

Передаваемая информация при подтверждении заказа.

namespace GM\_PluginCommon.TicketOrder

{

/// <summary>

/// Структура данных при подтверждении заказа

/// </summary>

public class TicketOrderConfirmInfo

{

/// <summary>

/// Идентификатор заказа

/// </summary>

public TicketOrderKeyInfo TicketOrderKeyInfo { get; set; }

/// <summary>

/// Коллекция чеков

/// </summary>

public List<Hardware.FiscalPrinter.FiscalReceiptInfo> FiscalReceiptInfo { get; set; }

}

}

#### Перечисление TicketType

Допустимые типы билетов.

namespace GM\_PluginCommon.TicketOrder

{

/// <summary>

/// Тип билета

/// </summary>

public enum TicketType

{

/// <summary>

/// Авиа

/// </summary>

Avia = 1,

/// <summary>

/// Ж/д

/// </summary>

Railway = 2,

/// <summary>

/// Автобус

/// </summary>

Autobus = 3

}

}

#### Перечисление TicketOrderConfirmResult

Результат подтверждения заказа.

namespace GM\_PluginCommon.TicketOrder

{

/// <summary>

/// Результат подтверждения заказа

/// </summary>

public enum TicketOrderConfirmResult

{

/// <summary>

/// Успех

/// </summary>

Success = 1,

/// <summary>

/// Неудача

/// </summary>

Failure = 2,

/// <summary>

/// Неопределенный результат

/// </summary>

Undetermined = 3,

/// <summary>

/// Ошибка

/// </summary>

Error = 4

}

}

# Перечень изменений в платформе относительно предыдущей версии 1.9

|  |  |  |  |
| --- | --- | --- | --- |
| Тип | Имя | Действие | Описание |
| Класс | PaymentSettlementInfo | Добавлен |  |
| Интерфейс | IScannerAccess | Обновлен | Метод BarcodeRequest - добавлен |
| Интерфейс | IPluginPaymentService | Обновлен | Метод Settle - добавлен |