1010 CODE
SOFTWARE

1010Code is a developer of full-scale software for banks and financial service providers.

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Описание шлюза SDKGateway

Payments
ATM Software

1010 CODE SOFTWARE

1010Code is a developer of full-scale software for banks and financial service providers.

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Содержание

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Глоссарий

Платежная система (ПС) — программный комплекс, реализующий возможность интерактивно оплачивать услуги или продукты получателя.

Получатель — человек или организация, предоставляющая услуги клиентам, или продающая товары, за определенную стоимость.

Клиент(плательщик) — человек(покупатель), использующий возможность ПС для осуществления расчетов с получателями.

Платеж — объект ПС, над которым производятся операции. Хранит в себе информацию о денежной операции при расчетах клиента за услуги или продукт с получателем.

Ядро — Основа программного комплекса ПС, осуществляющая возможность принимать информацию о платежах, обрабатывать, сообщать о них получателям, производить автоматизированные финансовые операции и др. Ядро определяет логику обработки платежа. Для осуществления операций ядро пользуется функциями прикладных модулей, каждый из которых обладает определенным функционалом.

Шлюз SDK — объект ПС — прикладной программный модуль, осуществляющий стыковку ПС с программными средствами получателей, которые обрабатывают информацию о финансовых расчетах. Обладает возможностью передать информацию о платеже в программный комплекс получателя. Функциями шлюза пользуется Ядро.

Протокол — набор правил, определяющий последовательность и формат сообщений для обмена информацией между программными (аппаратными средствами)

1010 CODE SOFTWARE

1010Code is a developer of full-scale software for banks and financial service providers.

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Требования к библиотекам разработки и обеспечения работы шлюзов SDK

Для разработки **шлюза SDK** используются классы из пакета **sdkgatewaylibrary.5.0.0.25141.nupkg** или того же пакета с более высокой версией.

Для возможности настройки и работы **шлюзов SDK** необходимо в Operation Hub установить пакет **externalgateway.4.1.2.25295.nupkg** или этот же пакет более высокой версии. Описание установки смотри в документации к Operation Hub и Operation Studio.

Назначение

Для передачи информации о платеже во внешние программные комплексы необходимо реализовать протокол понятный этому программному средству. Для этой цели в системе существует возможность создавать неограниченное количество шлюзов SDK (программных модулей), которые реализуют протоколы конкретных поставщиков услуг для совершения оплат.

С точки зрения реализации, шлюз SDK — это класс, наследующий базовому классу и реализующий необходимые методы, которыми пользуется ядро для отправки платежа. На шлюз можно смотреть как на адаптер, который внешнюю систему со своими правилами и протоколами предоставляет для Operation Hub в понятных ему операциях (методах).

Основными методами шлюза являются:

1. Проверка информации (проверка номера) о платеже (номер телефона, информация о плательщике и др.) для определения возможности создания платежа (**CheckAccount**)
2. Передача информации (создание платежа) о платеже программному комплексу получателя (**Process** и **CheckProcessStatus**)
3. При ошибочных ситуациях, отмена платежа (отзыв платежа) у получателя (**CanRecallPayment** и **RecallPayment**)
4. Сверка по платежам если того требует получатель (**SendRoll**).

Шлюзы SDK хранятся в отдельной сборке(dll) и создаются в момент старта службы Operation Hub или при сохранении/изменении настроек шлюза.

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Для реализации шлюза SDK в другие программные средства пользователю предоставляется возможность самому на любом программном языке для платформы .Net 4.6.2+ или net6+ создать класс (модуль) который реализует тот или иной протокол. В случае, когда зависимости шлюза при использовании netstandard2.0 не будут мешать работе, допустима разработка шлюза на netstandard2.0, но наилучший вариант — выбрать платформу отличную от netstandard*.

Каждый шлюза SDK создается в отдельной сборке, в которой реализуются классы описанные в данном документе (**GatewayCore** — наследник **GatewayCoreBase**, **SettingManager** — наследник **SettingManagerBase**).

Как это работает

При работе на ОС Windows сборка, содержащая внешнюю реализацию (шлюз SDK) грузиться в другой домен приложения, методом теневого копирование (после загрузки файл не занят и его можно заменить), что позволяет заменять реализацию без выгрузки всего приложения, т.к. типы создаются и выгружаются вместе с доменом. При работе на ОС Linux для загрузки сборки используется LoadContextAssembly.

После того как сборка загружена создаются объекты в созданном домене/LoadContextAssembly, а в основном сохраняются прокси на эти объекты.

Первым создается менеджер настроек, определяющий какими данными нужно будет оперировать, затем производиться его инициализация. Если операция произошла успешно, далее создается реализация ядра шлюза, которому будут перенаправляться вызовы. Полный список действий при создании конфига выглядит следующим образом (эти этапы выписываются в лог при успешном завершении либо обрываются сообщением об неуспешном прохождении последнего этапа):

1. Ищется файл внешней сборки заданный в настройках,
2. Создается внешний домен,
3. В домене создается загрузчик внешних библиотек, затем он загружает внешнюю сборку,
4. Создается менеджер настроек, реализованный во внешней библиотеке
5. Запрашиваются данные у менеджера настроек для создания объектов привязки, и происходит их создание
6. Запрашиваются данные (ключи) для получения настроек из стандартного хранилища

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

настроек, для дальнейшей передачи их в шлюз.

7. Создается ядро шлюза
8. С помощью полученных настроек производится инициализация шлюза

Детали лога зависят от версии Operation Hub. Например, для версий, работающих под ОС Linux, создания доменов приложения не происходит.

Пример:
[2007.12.12 17.34.31 : Event]
Шлюз SDK. Конфиг номер 1 .Сборка :
Найден файл внешней библиотеки
[2007.12.12 17.34.31 : Event]
Шлюз SDK. Конфиг номер 1 .Сборка TestGW.dll:
Создан внешний домен приложения FriendlyNameTestGW.dll633330776711001250
[2007.12.12 17.34.31 : Event]
Шлюз SDK. Конфиг номер 1 .Сборка TestGW.dll:
Создан загружчик внешних библиотек
[2007.12.12 17.34.31 : Event]
Шлюз SDK. Конфиг номер 1 .Сборка TestGW.dll:
Создан менеждер настроек.
[2007.12.12 17.34.31 : Event]
Шлюз SDK. Конфиг номер 1 .Сборка TestGW.dll:
Созданны объекты привязки для операций.
[2007.12.12 17.34.31 : Event]
Шлюз SDK. Конфиг номер 1 .Сборка TestGW.dll:
Полученны ключи настроек.
[2007.12.12 17.34.31 : Event]
Шлюз SDK. Конфиг номер 1 .Сборка TestGW.dll:
Созданно ядро шлюза
[2007.12.12 17.34.31 : Event]
Шлюз SDK. Конфиг номер 1 .Сборка TestGW.dll:
Проверка доступности настроек для шлюза прошла успешно
[2007.12.12 17.34.31 : Event]
Шлюз SDK. Конфиг номер 1 .Сборка TestGW.dll:
Инициализация шлюза прошла успешно

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

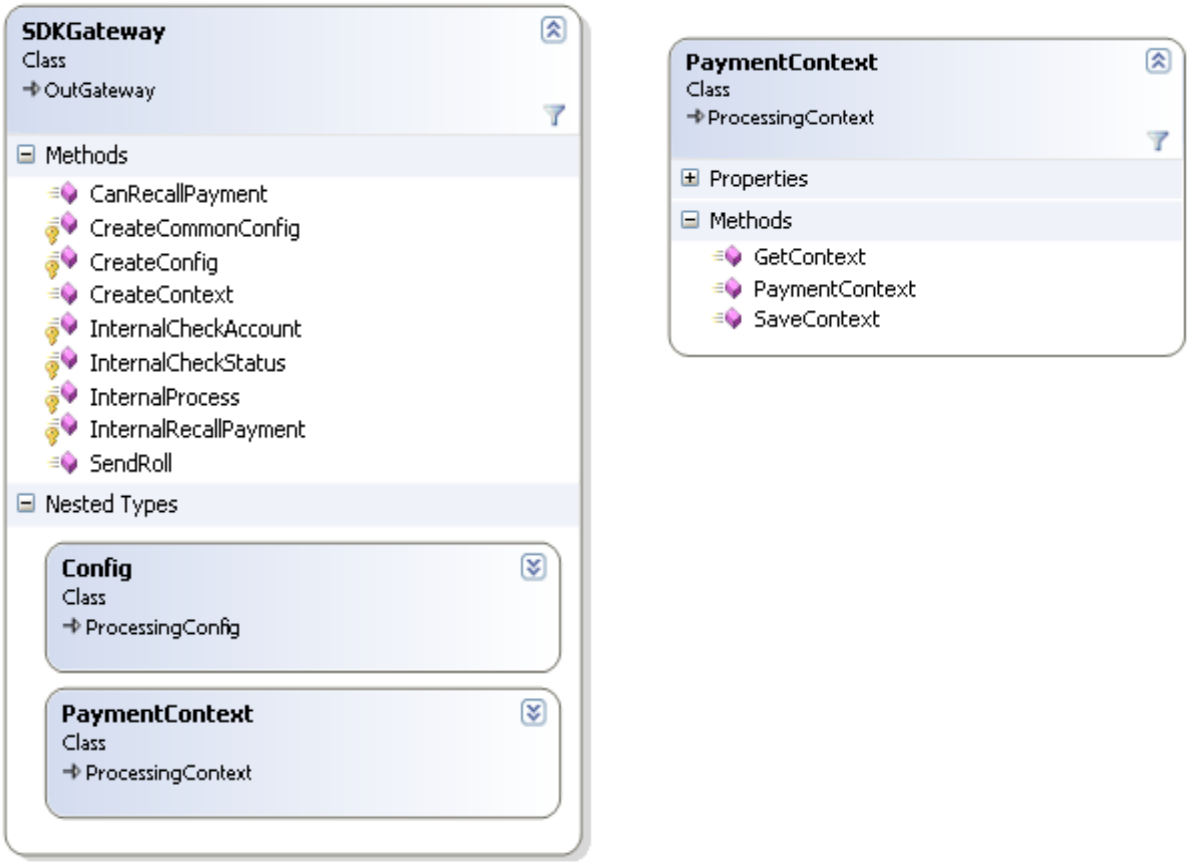
Шлюз SDK в методах принимает аргументом объект контекста. Заполнение контекста происходит только теми объектами, которые нужны для реализации конкретной функции (*если быть точнее, то только теми объектами, для которых* ***SettingsManager*** *предоставил ключи привязки*).

Состав шлюза SDKGateway

Ядро ПС использует методы шлюза для обработки платежей. В методах шлюза SDK определяется специальная логика для работы с внешними системами или логика без взаимодействия со внешними системами. Это специальная кастомная логика. Основное требование для методов шлюза — это корректная работа с контекстом вызова конкретной операции и корректное выставление статусов обработки этой операции. Основываясь на статусах Operation Hub планирует дальнейшую работу по платежу.

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Классы SDKGateway



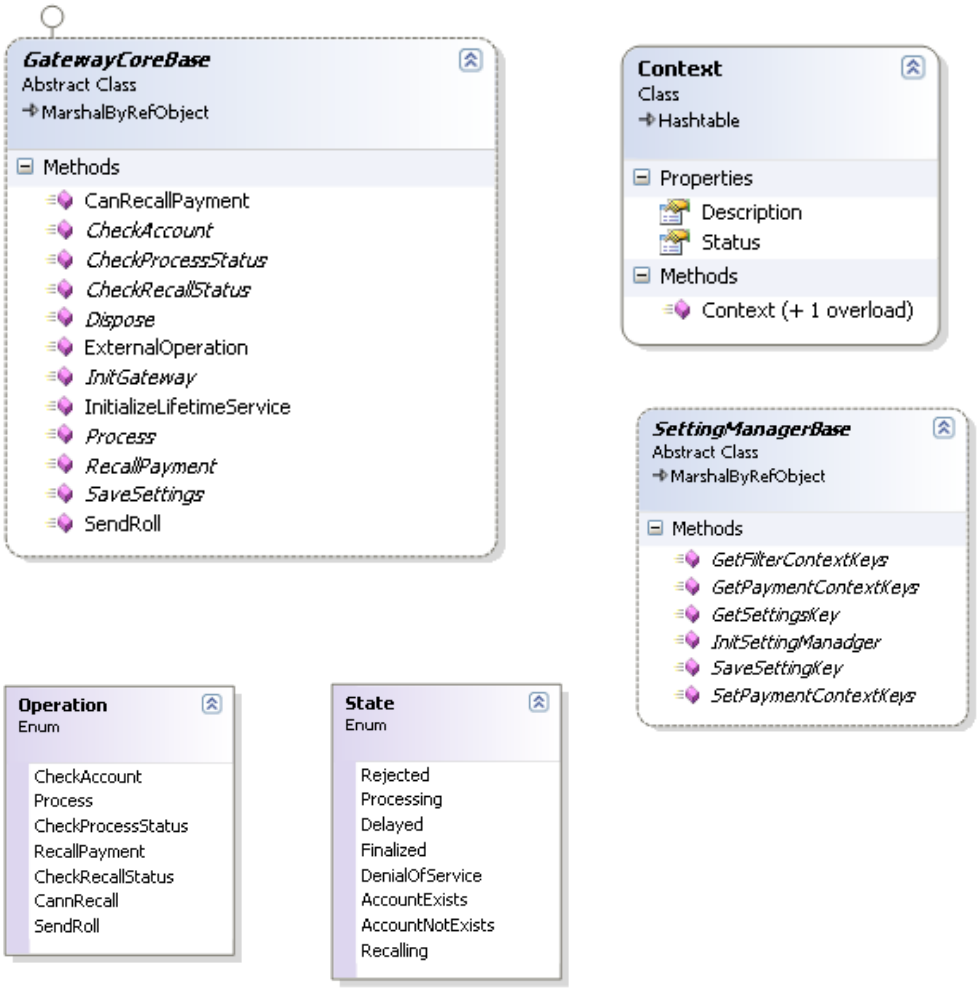
Тут показаны служебные классы, которые не используются шлюзом SDK напрямую.

SDKGateway — это шлюз, написанный на внутреннем SDK, предоставляющий **шлюзу SDK** настройки. Также этот внутренний шлюз производит делегирование вызовов Operation Hub к созданному SDK шлюзу.

PaymentContext — объект системы, хранящий информацию о платеже. Позволяет, используя объекты привязки, заполнять контекст операций необходимыми данными

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Классы внешней библиотеки для реализации своего ядра шлюза



GatewayCoreBase — класс, определяющий интерфейс ядра шлюза

SettingsManagerBase — класс, определяющий интерфейс менеджера настройки

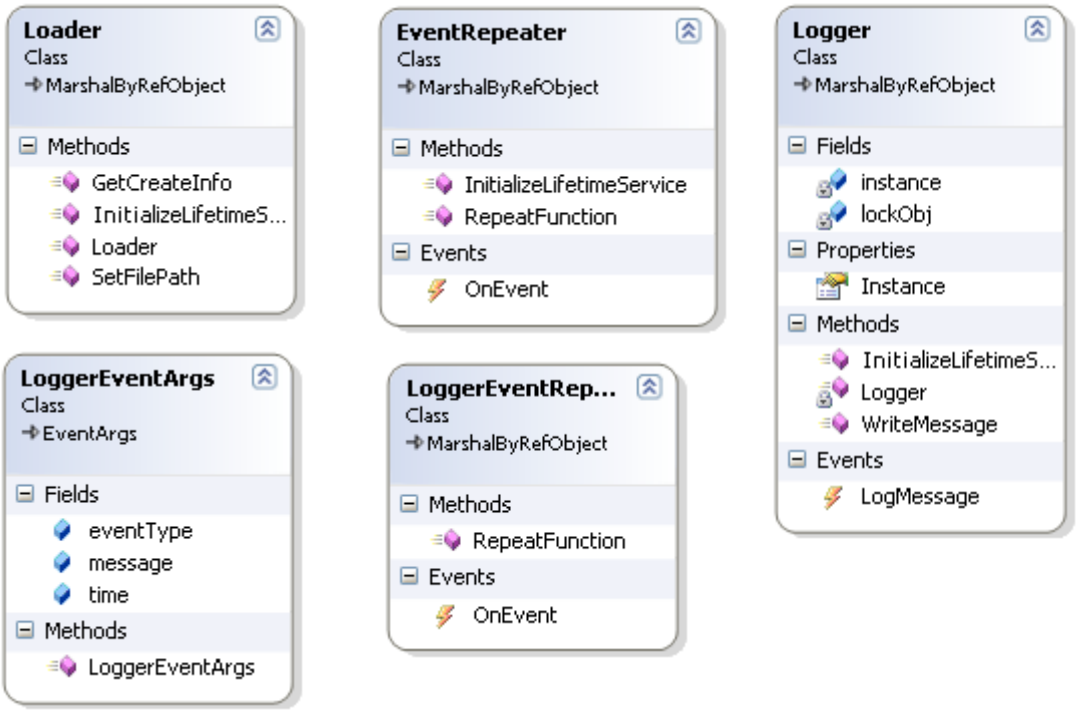
Context — класс, определяющий контекст операций

Operation — перечисление, определяющее операции ядра шлюза

State — состояния платежа в системе понятные шлюзу и системе

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Дополнительные классы в библиотеке (служебные)



Loader — класс, который грузит внешнюю библиотеку с реализацией шлюза.

EventRepeater — класс для передачи сообщения о необработанном исключении в главный домен приложения.

Logger (реализует IBP.SDKGatewayLibrary.ILogger) — класс позволяющий передавать сообщения для логгирования на уровне ядра системы (основной лог).

LoggerEventRepiter — ретранслятор событий логгера между доменами.

LoggerEventArgs — класс аргументов события логгера.

Передача сообщений в лог сервера

Для обеспечения возможности передачи сообщений в лог сервера, в библиотеке SDKGatewayLibrary, создан класс Logger. Используя функцию, WriteMessage(string mess, int type) можно передать сообщения в основной лог.

mess — сообщение для записи в лог

type — тип сообщения, где

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

1 — обычное сообщение
2 — сообщение повышенной важности
3 — сообщение о ошибке
4 — сообщение о критической ошибке

Доступ к логгеру — Свойство шлюза SDK GatewayCoreBase.Logger

Разработка внешней реализации.

Свой шлюз можно написать на любом языке для .net определив его в отдельную сборку.

Требования: сборка должна иметь в зависимостях SDKGateway.dll

И в ней должны быть реализованы два класса:

1. class SettingManager : SettingManagerBase — менеджер настроек
2. class GatewayCore : GatewayCoreBase — ядро шлюза

Внимание! Классы должны иметь именно такие имена, и никак иначе.

Рассмотрим каждый из этих классов подробнее:

SettingManager

```
/// <summary>
/// Базовый класс для менеджера настроек
/// </summary>
public abstract class SettingManagerBase : MarshalByRefObject
{
        /// <summary>
        /// Получить ключи, необходимые для заполения контекста
        /// </summary>
        /// <param name="initString"> строка инициализации </param>
        /// <returns> описание ошибок инициализации </returns>
        public abstract string InitSettingManadger(string initString);
        /// <summary>
        /// Получить ключи, необходимые для заполения контекста
        /// </summary>
        /// <param name="operation"> Тип операции </param>
        /// <returns> Массив ключей </returns>
        public abstract string[] GetPaymentContextKeys(Operation operation);
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
        /// <summary>
        /// Получить ключи, необходимые для сохрания параметров контекста
        /// </summary>
        /// <param name="operation"> Тип операции </param>
        /// <returns> Массив ключей </returns>
        public abstract string[] SetPaymentContextKeys(Operation operation);
        /// <summary>
        /// Получить ключи, необходимые для заполения контекста фильтра при создании реестра ///
</summary>
        /// <returns> Массив ключей </returns>
        public abstract string[] GetFilterContextKeys();
        /// <summary>
        /// Получить ключи, необходимые для получения настроек шлюза
        /// </summary>
        /// <returns> Массив ключей </returns>
        public abstract string[] GetSettingsKey();
        /// <summary>
        /// Получить ключи, необходимые для сохранения настроек шлюза
        /// </summary>
        /// <returns></returns>
        public abstract string[] SaveSettingKey();
        }
```

При создании конфига происходит создания внешнего менеджера настроек.

В целом класс определяет, какие данные нужно получить при каждой операции в контексте, а какие передать обратно и сохранить:

1. void InitSettingManadger(string initString) — эта функция вызывается поле создания менеджера настроек
   initstring — строка инициализации (значение в настройках шлюза SDKGateway под ключем initsettstring,
   если произошли ошибки, функция должна выкинуть AplicationException с описанием ошибки
2. string[] GetPaymentContextKeys(Operation operation) — эта функция возвращает массив ключей для каждой операции, для заполнения данными контекста операции
   operation — значение перечисления Operation, определяющая для какой операции запрашиваются ключи
   return — Массив ключей если операция используется шлюзом иначе null, если null — операция не может быть использована шлюзом
3. string[] SetPaymentContextKeys(Operation operation) — эта функция возвращает массив ключей для каждой операции, сохраняются данные из возвращаемого контекста в объекты системы
   operation– значение перечисления Operation, определяющая для какой операции запрашиваются ключи
   return — Массив ключей если операция используется шлюзом иначе null, если null — не вызывается

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

функция сохранения объектов в системе

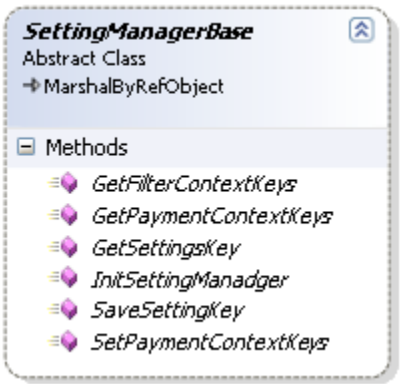
4. string[] GetFilterContextKeys() — эта функция возвращает массив ключей для дополнительных данных по выборке при вызове операции посылка реестра из фильра(объект хранящий параметры выборки)
   return — Массив ключей если операция используется шлюзом иначе null

Эти ключи используются для создания объектов привязки, с помощью которых заполняется контекст(или фильтр-контекст для метода сверки реестра) при вызове операций шлюза, или сохраняются в объектах системы при получении контекста после выполнения операции.

Формат ключей: На самом деле ключи это строка описывающая путь к объекту начиная с базового объекта(PaymentContext или Filter), например “PaymentContext.Payment.Account” — аккаунт платежа. Полный список объкетов и ключей см. в приложении 1

1. string[] GetSettingsKey(); — эта функция возвращает массив ключей для получения натроек из стандартного хранилища настроке и передачи их в шлюз,
   return– ключи
2. string[] SaveSettingKey (); — эта функция возвращает массив ключей для сохранения под ними в стандартном хранилище настрок данных шлюза, перед тем как шлюз будет уничтожен
   return– ключи

Формат ключей: В этих функциях под ключами понимаются строки(названия), определяющие под какими названиями та или иная настройка будет храниться.



Класс наследует MarsahlByReObject т.к. доступ к объекту ядра шлюза к нему происходит из другого домена приложения (для Windows)

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

GatewayCoreBase

```
/// <summary>
/// Базовый класс шлюза
/// </summary>
public abstract class GatewayCoreBase : MarshalByRefObject, IDisposable
{
        /// <summary>
        /// Инициализация шлюза
        /// </summary>
        /// <param name="settigs"> Данные, полученные с помощью ключей GetSettingsKey </param>
        /// <exception cref="ApplicationException"> Сообщение об ошибке при инициализации </exception>
        public abstract void InitGateway(Hashtable settigs);
        /// <summary>
        /// Сохранить данные перед выгрузкой шлюза
        /// </summary>
        /// <returns> данные для сохрания с помощью ключей SaveSettingKey< </returns>
        public abstract Hashtable SaveSettings();
        /// <summary>
        /// Выполнить проведение платежа
        /// </summary>
        /// <param name="context"> Контекст обработки </param>
        public abstract void Process(ref Context context);
        /// <summary>
        /// Выполнить проверку статуса платежа
        /// </summary>
        /// <param name="context"> Контекст обработки </param>
        public abstract void CheckProcessStatus(ref Context context);
        /// <summary>
        /// Выполнить  проверку данных платежы
        /// </summary>
        /// <param name="context"> Контекст обработки </param>
        public abstract void CheckAccount(ref Context context);
        /// <summary>
        /// Выполнить отзыв платежа
        /// </summary>
        /// <param name="context"></param>
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
public abstract void RecallPayment(ref Context context);
/// <summary>
/// Проверить статус отзыва
/// </summary>
/// <param name="context"> Контекст обработки </param>
public abstract void CheckRecallStatus(ref Context context);
/// <summary>
/// Выпонить проверку возможности отзыва платежа
/// </summary>
/// <param name="context"> Контекст обработки </param>
/// <returns>True — отзыв возможен, False — отзыв невозможен</returns>
public virtual bool CanRecallPayment(ref Context context)
{
        return false;
}
/// <summary>
/// Выполнить внешнюю операцию( Пока не используется)
/// </summary>
/// <param name="operationInfo">контекст операцииparam>
/// <returns> Результат</returns>
public virtual string ExternalOperation(string operationInfo)
{
        throw new NotImplementedException("Îïåðàöèÿ íå ïîääåðæèâàåòñÿ");
}
/// <summary>
/// Отправить реестр
/// </summary>
/// <param name="payments"> массив контекстов операций (платежей)param>
/// <param name="filterContext"> Контест фильтра</param>
/// <returns> True — успешно, False — ошибка<</returns>
public virtual bool SendRoll(Context[] payments, Hashtable filterContext)
{
        return true;
}
/// <summary>
/// Объект будет жить вечно
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
/// </summary>
/// <returns>null</returns>
public override object InitializeLifetimeService()
{
        return null;
}
/// <summary>
/// Освободить занимаемые ресурсы
/// </summary>
public abstract void Dispose();
}
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Описание работы шлюза SDKGateway

Описание работы

При создании ядра шлюза происходит создания объекта класса GatewayCore отнаследованного от базового GatewayCoreBase. Этим объектом пользуется шлюз SDKGateway, делегируя ему вызовы.

Последовательность действий при инициализации шлюза SDK:

1. Создание шлюза
2. Инициализации для создания внутренних объектов по настройкам. Для этого вызывается функция InitGateway(Hashtable settigs) в коллекции у которой строки (значения) полученные из стандартного хранилища под ключами, полученными от менеджера настроек. Если инициализация прошла не успешно, пользовательский код может выкинуть исключения ApplicationException с сообщением об ошибке, которая выпишется в лог.
3. Шлюз готов к работе.

Далее в зависимости от ситуации шлюз сам выбирает, какой функцией ему пользоваться:

1. CheckAccount — проверить информацию на возможность создания платжа у получателя(проверить номер)
2. Process — провести платеж , отослав информацию получателю
3. CheckProcessStatus — проверить статус платежа, если получатель однозначно не ответил что платеж принят
4. CanRecallPayment — проверить возможно ли платеж отозвать не запрашивая у клиента
5. RecallPayment — произвести отзыв платежа
6. CheckRecallStatus — проверить отзываемый платеж, если получатель однозначно не ответил, что платеж отозван или нет.
7. SendRoll — осуществить сверку платежей

Как выполняться каждая операция:

1. SDKGateway получает объект, по которому нужно выполнить какую-либо операцию,
2. С помощью “объектов привязки на заполнение” если они существуют производиться заполнение контекста для операции, затем шлюз вызывает определенную функцию

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

ядра, передовая в нее контекст. Если “объектов привязки на заполнение” не существует, то ядру ПС передается исключение, говорящее о том, что данную нельзя использовать

3. Пользовательский код обрабатывает контекст, делая необходимые операции и меняет статус платежа, сохраняет его в и может добавить любые атрибуты в контекст
4. SDKGateway получая обратно контекст анализирует выставленный статус и присваивает его платежу, также он по необходимости (существует “объекты привязки на сохранение”) сохраняет данные из контекста в объектах системы
5. В некоторых функциях SendRoll, CanRecallPayment, возвращаемый контекст не анализируется, а результат работы определяется возвращаемым значением.

Если при работе пользователь поменяет настройки, SDKGateway произведет обработку всех платежей, уже попавших в него и приступит к созданию нового инстанса шлюза SDK. Перед этим вызвав функцию Dispose() старого шлюза.

При выгрузке происходят следующие действия:

1. Вызывается функция SaveSettings, SDKGateway получает коллекцию объектов и если были заданы ключи для сохранения настроек, производиться попытка сохранить их в стандартном хранилище настроек плюсуя к ключам номер конфига (см. SettingManagerBase)
2. Затем вызывается функция Dispose() — где сторонний разработчик должен правильно освободить занятые им ресурсы, потоки, нити и др.
3. Выгружается домен приложения для Windows или выгружаются сборки из LoadContextAssembly для Linux.

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Диаграммы смены состояния платежа

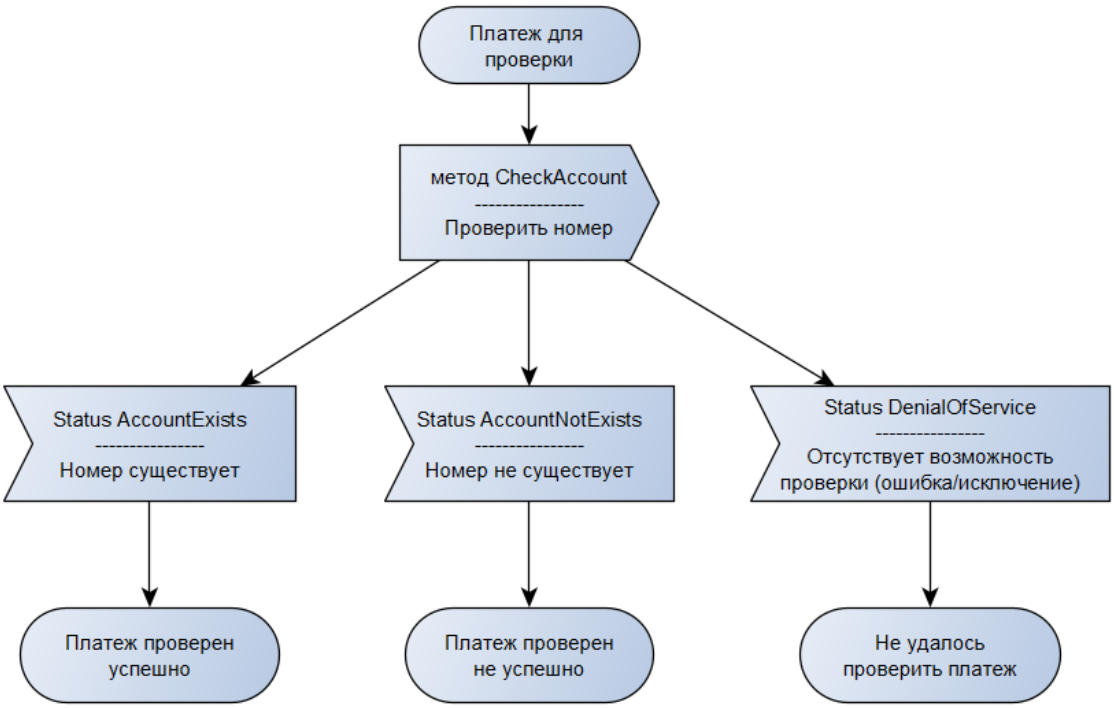


Рис 1. Операция проверки номера

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

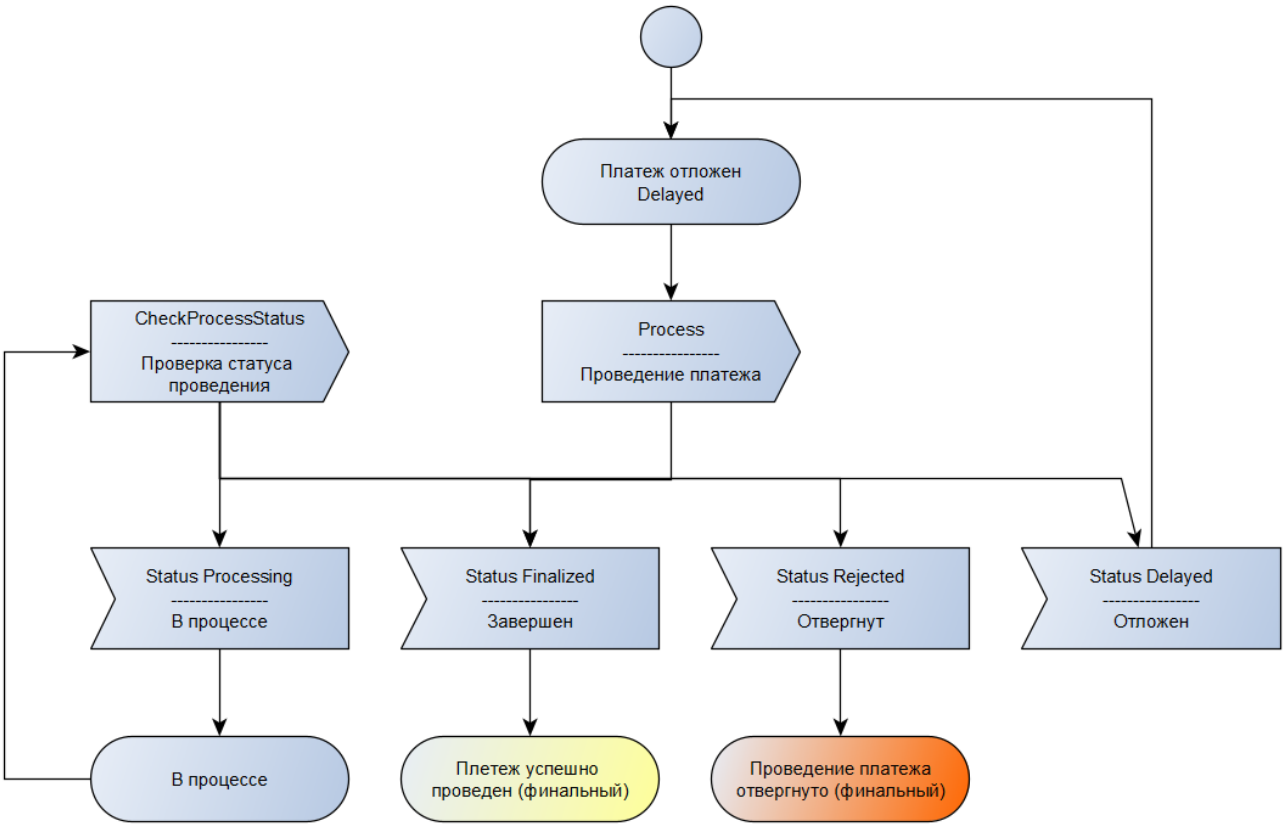


Рис 2. Операция проведения платежа

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

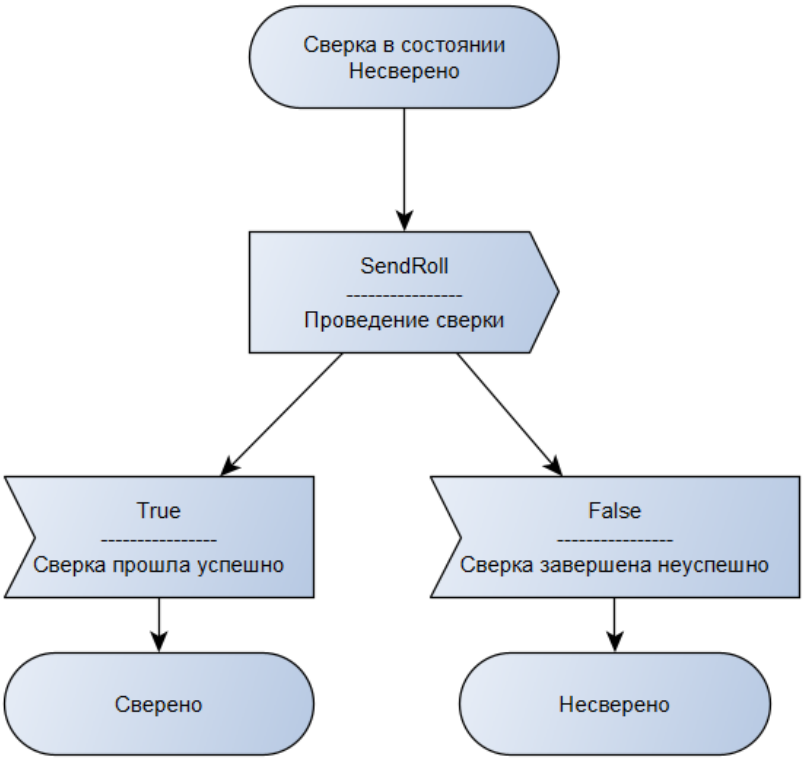


Рис 3. Операция проверки номера

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

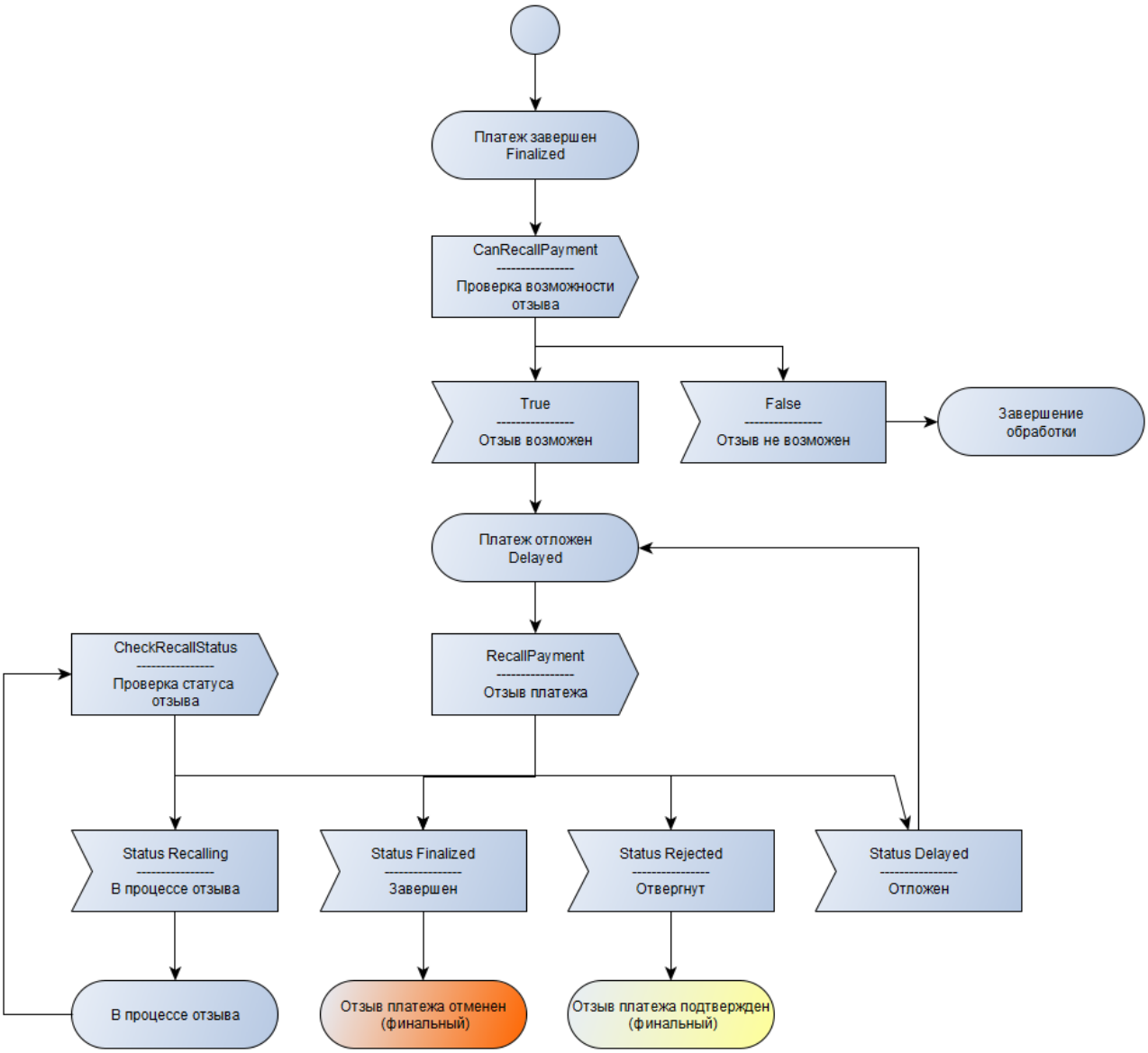


Рис 4. Операция отмены платежа

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Сигнал в GatewayCore — это просто вызов соответствующей операции
Перечисление операций задается в enum Operation

1. CheckAccount — проверить информацию на возможность создания платежа у получателя(проверить номер)
2. Process — провести платеж , отослав информацию получателю
3. CheckProcessStatus — проверить статус платежа, если получатель однозначно не ответил что платеж принят
4. CanRecallPayment — проверить возможно ли платеж отозвать не запрашивая у клиента
5. RecallPayment — произвести отзыв платежа
6. CheckRecallStatus — проверить отзываемый платеж, если получатель однозначно не ответил, что платеж отозван или нет.
7. SendRoll — осуществить сверку платежей

Сигнал от GatewayCore — это поле State выставленное пользовательским кодом или булевская переменная, возвращенная функцией (true, false)

Перечисление состояний определяется в enum State

Статусы при проведении или отзыве

Rejected — отвергнут, платеж зафиксирован в нашей платежной системе, но не принят получателем

Processing — в процессе, платеж попал к получателю но получатель не считае платтеж завершенным

Delayed — отложен , платеж попал под шлюз для выполнеия действий над ним, но не смог отправиться получателю(нужно повторно выполнить туже операцию)

Finalized — завершен , платеж принят получателем

Recalling — отзываеться(сходное состояние с Processing) только при отзыве платежа, получательначал отзывать платеж , но однозначно не сказал что он его отозвал

Статусы на проверке номера

DenialOfService — нет возможности проверить номер

AccountExists — платеж прошел проверку

AccountNotExists — платеж не прошел проверку

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Статус сверки

1. **true** — сверка прошла успешна, платежам устанавливается признак, что они прошли сверку
2. **false** — сверка завершилась неудачно, ядро ПС еще раз вызовет функцию сверку для этих платежей

Объекты, с которыми работают “объекты привязки”

PaymentContext — контекст платежа

Filter — объект определяющий критерии выборки сверки

Полный список свойств и полей, с которыми можно работать смотри в приложении 1.

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Приложения

Приложение 1. Объекты системы.

Все объекты системы можно посмотреть в файле, SDKGateways\HelpDocs\, он обновляется автоматически, при загрузке сервера

Табл.1 Объекты системы которыми можно заполнить контекст операции.

Объект	**Путь**	**Тип**	**Описание**
PaymentContext		**Сложный тип, не используется**	**Платежный контекст**
Trace	PaymentContext.Trace	Boolean :get=True, set=False	Включена ли трассировка шлюза
Payment		**Сложный тип, не используется**	**Платеж**
IsChecked	PaymentContext.Payment.IsChecked	Boolean :get=True,set=True	Пошел ли платеж проверку
Value	PaymentContext.Payment.Value	Decimal :get=True,set=False	Сумма платежа для получателя
Total	PaymentContext.Payment.Total	Decimal :get=True,set=False	Сумма принятая от клиета
InputDate	PaymentContext.Payment.InputDate	DateTime :get=True,set=False	Время создания платежа на терминале
ServerTime	PaymentContext.Payment.ServerTime	DateTime :get=True,set=False	Время когда платеж пришел на сервер
StateTime	PaymentContext.Payment.StateTime	DateTime :get=True,set=False	Время когда платеж упал в конечный статус
PointId	PaymentContext.Payment.PointId	Guid :get=True,set=False	Уникальный идентификатор точки
ServiceId	PaymentContext.Payment.ServiceId	Guid :get=True,set=False	Уникальный идентификатор сервиса
RecipientId	PaymentContext.Payment.RecipientId	Guid :get=True,set=True	Уникальный идентификатор получателя
RecipientName	PaymentContext.Payment.RecipientName	String :get=True,set=False	Имя получателя в системе
ServiceName	PaymentContext.Payment.ServiceName	String :get=True,set=False	Имя сервиса в системе
PointName	PaymentContext.Payment.PointName	String :get=True,set=False	Имя точки в системе
ClientName	PaymentContext.Payment.ClientName	String :get=True,set=False	Имя клиента в системе
GatewayName	PaymentContext.Payment.GatewayName	String :get=True,set=False	Имя шлюза в системе
FullState	PaymentContext.Payment.FullState	IBP.PaymentProject2.Common.PaymentState :get=True,set=True	Полное состояние платежа
WorkState	PaymentContext.Payment.WorkState	IBP.PaymentProject2.Common.PaymentState :get=True,set=True	Работе состояние платежа
Account	PaymentContext.Payment.Account	String :get=True,set=False	Номер платежа
TicketNumber	PaymentContext.Payment.TicketNumber	Int32 :get=True,set=True	Номер чека
Comission	PaymentContext.Payment.Comission	Decimal :get=True,set=False	Коммисия платежа
UserId	PaymentContext.Payment.UserId	Guid :get=True,set=True	Уникальный идентификатор пользователя создавшего платеж
Number	PaymentContext.Payment.Number	String :get=True,set=False	Номер чека платежа
GatewayType	PaymentContext.Payment.GatewayType	IBP.PaymentProject2.Common.GatewayType :get=True,set=True	Тип шлюза
StartBalance	PaymentContext.Payment.StartBalance	Decimal :get=True,set=False	Стартовый баланс
Balance	PaymentContext.Payment.Balance	Decimal :get=True,set=False	Баланс
HasSource	PaymentContext.Payment.HasSource	Boolean :get=True,set=False	Имеет ли платеж сдачу
Sources	PaymentContext.Payment.Sources	String[] :get=True,set=False	Номера платежей со здачей
Amount	PaymentContext.Payment.Amount	Decimal :get=True,set=True	Сумма проводки по платежу

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Объект	**Путь**	**Тип**	**Описание**
ItemType	PaymentContext.Payment.ItemType	String :get=True,set=False	Текстовое описание объекта
RedirectState	PaymentContext.Payment.RedirectState	IBP.PaymentProject2.Common.RedirectState :get=True,set=True	Статус переотправки
RedirectDescription	PaymentContext.Payment.RedirectDescription	String :get=True,set=False	Описание статуса переотправки
HasParent	PaymentContext.Payment.HasParent	Boolean :get=True,set=False	Имеет ли платеж родителя(возникает при переотправке)
HasChild	PaymentContext.Payment.HasChild	Boolean :get=True,set=False	Имеет ли платеж наследника(возникает при переотправке)
Comment	PaymentContext.Payment.Comment	String :get=True,set=True	Комментарий статуса(каждого в который переходит платеж)
Generation	PaymentContext.Payment.Generation	Int32 :get=True,set=True	Поколение платежа
IsArchieved	PaymentContext.Payment.IsArchieved	Boolean :get=True,set=False	Архивный платеж
WaitTime	PaymentContext.Payment.WaitTime	Int32 :get=True,set=True	Через сколько платеж попадет под шлюз, если стал отложен
HasAttributes	PaymentContext.Payment.HasAttributes	Boolean :get=True,set=False	Имеет ли платеж дополнительные атрибуты
Payment[stringKey]	PaymentContext.Payment[stringKey]	String :get=True,set=True	Именованная коллекция до атрибутов
Count	PaymentContext.Payment.Count	Int32 :get=True,set=False	Кол-во доп. атриибутов
Id	PaymentContext.Payment.Id	Guid :get=True,set=False	Уникальный номер платежа
Serial	PaymentContext.Payment.Serial	Int32 :get=True,set=True	Уникальный порядковый номер платежа в базе
Old	PaymentContext.Payment.Old	Boolean :get=True,set=False	Устаревший платеж
Service		**Сложный тип, не используется**	**Сервис приема платежей**
Destination	PaymentContext.Service.Destination	String :get=True,set=True	Описание проводки по сервису
Cts	PaymentContext.Service.Cts	Int16 :get=True,set=True	Символ кассового плана
Alias	PaymentContext.Service.Alias	String :get=True,set=True	Алиас сервиса (по протоколу)
Name	PaymentContext.Service.Name	String :get=True,set=True	Имя сервиса в системе
MinValue	PaymentContext.Service.MinValue	Decimal :get=True,set=True	Минимальная величина суммы платежа по данному сервису
MaxValue	PaymentContext.Service.MaxValue	Decimal :get=True,set=True	Максимальная величина суммы платежа по данному сервису
Fee	PaymentContext.Service.Fee	Single :get=True,set=True	Вознаграждение для этого сервиса по умолчанию
RecipientId	PaymentContext.Service.RecipientId	Guid :get=True,set=True	Айди получателя кому пойдет платеж с данного сервиса
ItemType	PaymentContext.Service.ItemType	String :get=True,set=False	Текстовое описание типа
HasBindingAttributes	PaymentContext.Service.HasBindingAttributes	Boolean :get=True,set=False	Есть ли у сервиса Обязательные атрибуты
HasOptionalAttributes	PaymentContext.Service.HasOptionalAttributes	Boolean :get=True,set=False	Есть ли у сервиса Опциональные атрибуты
HasOutAttributes	PaymentContext.Service.HasOutAttributes	Boolean :get=True,set=False	Есть ли у сервиса Выходные атрибуты
BindingAttributes	PaymentContext.Service.BindingAttributes	String[] :get=True,set=False	Значения ключей обязательный атрибутов
OptionalAttributes	Service.OptionalAttributes	String[] :get=True,set=False	Значения ключей необязательных атрибутов
InAttributes	PaymentContext.Service.InAttributes	String[] :get=True,set=False	Значения ключей входных

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Объект	**Путь**	**Тип**	**Описание**
			атрибутов
Count	PaymentContext.Service.Count	Int32 :get=True,set=False	Кол-во доп атрибутов
HasAttributes	PaymentContext.Service.HasAttributes	Boolean :get=True,set=False	Есть ли дополнительные атрибуты
Service[string Key]	PaymentContext.Service[stringKey]	String :get=True,set=True	Именованная коллекция атрибутов у сервиса
Id	PaymentContext.Service.Id	Guid :get=True,set=False	Уникальный идентификатор получателя в системе
Serial	PaymentContext.Service.Serial	System.Int32 :get=True,set=True	Уникальный порядковый номер получателя в системе
Old	PaymentContext.Service.Old	Boolean :get=True,set=False	Устарел ли объект
Recipient		**Сложный тип, не используется**	**Получатель платежей**
Name	PaymentContext.Recipient.Name	String :get=True,set=True	Имя получателя платежа
Address	PaymentContext.Recipient.Address	String :get=True,set=True	Адрес получателя платежа
Account	PaymentContext.Recipient.Account	String :get=True,set=True	Счет получателя платежа
Destination	PaymentContext.Recipient.Destination	String :get=True,set=True	Описание проводки
Info	PaymentContext.Recipient.Info	String :get=True,set=True	Информация о получателе
TownId	PaymentContext.Recipient.TownId	Guid :get=True,set=True	Уникальный номер города которому принадлежит получатель
Email	PaymentContext.Recipient.Email	String :get=True,set=True	Электронный адрес получателя
HasEmail	PaymentContext.Recipient.HasEmail	Boolean :get=True,set=False	Есть ли электронный адрес у получателя
GatewayType	PaymentContext.Recipient.GatewayType	IBP.PaymentProject2.Common.GatewayType :get=True,set=True	Тип шлюза используемый получателем
InputDate	PaymentContext.Recipient.InputDate	DateTime :get=True,set=False	Время создания получателя в системе
Fee	PaymentContext.Recipient.Fee	Single :get=True,set=True	Вознаграждение
OperationRequired	PaymentContext.Recipient.OperationRequired	Boolean :get=True,set=False	Нужна ли вторая проводка
ItemType	PaymentContext.Recipient.ItemType	String :get=True,set=False	Текстовое описание типа
AllowCustomRoll	PaymentContext.Recipient.AllowCustomRoll	Boolean :get=True,set=True	Разрешена отправка реестра через операциониста
HasAttributes	PaymentContext.Recipient.HasAttributes	Boolean :get=True,set= False	Есть ли дополнительные атрибуты у получателя
Recipient[stringKey]	PaymentContext.Recipient[stringKey]	String :get=True,set=True	Именованная коллекция атрибутов у получателя
Count	PaymentContext.Recipient.Count	Int32 :get=True,set=False	Кол-во доп. Атрибутов у получателя
Id	PaymentContext.Recipient.Id	Guid :get=True,set=False	Уникальный идентификатор получателя в системе
Serial	PaymentContext.Recipient.Serial	System.Int32 :get=True,set=True	Уникальный порядковый номер получателя в системе
Old	PaymentContext.Recipient.Old	Boolean :get=True,set=False	Устарел ли объект
Point		**Сложный тип, не используется**	**Точка приема платежей**
ClientId	PaymentContext.Point.ClientId	Guid :get=True,set=True	Уникальный номер клиента которому принадлежит данная точка
Account	PaymentContext.Payment.Point.Account	String :get=True,set=True	Номер счета у точки
CashId	PaymentContext.Point.CashId	Int32 :get=True,set=True	Айди кассы

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Объект	**Путь**	**Тип**	**Описание**
IsCashIn	PaymentContext.Point.IsCashIn	Boolean :get=True,set=False	Автоматизированный аппарат для приема платежей (Кешин)
Name	PaymentContext.Point.Name	String :get=True,set=True	Имя точки в системе
IsEnabled	PaymentContext.Point.IsEnabled	Boolean :get=True,set=True	Состояние разблокировки точки
PublicKeyId	PaymentContext.Point.PublicKeyId	Guid :get=True,set=True	Идентификатор публичного ключа точки
InputDate	PaymentContext.Point.InputDate	DateTime :get=True,set=True	Время создания точки в системе
Percent	PaymentContext.Point.Percent	Single :get=True,set=True	неиспользуеться
CurrentNumber	PaymentContext.Point.CurrentNumber	Int32 :get=True,set=True	неиспользуеться
PointType	PaymentContext.Point.PointType	IBP.PaymentProject2.Common.PointType :get=True,set=True	Тип точки
CanUseCheck	PaymentContext.Point.CanUseCheck	Boolean :get=True,set=False	Разрешено ли точке использовать сдачу
ItemType	PaymentContext.Point.ItemType	String :get=True,set=False	Текстовое описание объекта
HasAttributes	PaymentContext.Point.HasAttributes	Boolean :get=True,set= False	Есть ли дополнительные атрибуты у точки
Point[stringKey]	PaymentContext.Point[stringKey]	String :get=True,set=True	Именованная коллекция атрибутов у точки
Count	PaymentContext.Point.Count	Int32 :get=True,set=False	Кол-во доп. Атрибутов у точки
Id	PaymentContext.Point.Id	Guid :get=True,set=False	Уникальный идентификатор точки в системе
Serial	PaymentContext.Point.Serial	System.Int32 :get=True,set=True	Уникальный порядковый номер точки в системе
Old	PaymentContext.Point.Old	Boolean :get=True,set=False	Устарел ли объект
Client		**Сложный тип, не используется**	**Клиент, владеющий точкой приема платежей**
Name	PaymentContext.Client.Name	String :get=True,set=True	Имя клиента в системе
IsEnabled	PaymentContext.Client.IsEnabled	Boolean :get=True,set=True	Состояние разблокировки клиента
InputDate	PaymentContext.Client.InputDate	DateTime :get=True,set=False	Время создания клиента в системе
TownId	PaymentContext.Client.TownId	Guid :get=True,set=True	Уникальный идентификатор города которому принадлежит клиент
RollFormat	PaymentContext.Client.RollFormat	IBP.PaymentProject2.Common.RollFormat :get=True,set=True	Формат реестра для клиента
Email	PaymentContext.Client.Email	String :get=True,set=True	Электронный адрес клиента
ItemType	PaymentContext.Client.ItemType	String :get=True,set=False	Текстовое описание типа
OperPermission	PaymentContext.Client.OperPermission	IBP.PaymentProject2.Common.Client+Permission :get=True,set=True	Маска разрешений для операциониста
Id	PaymentContext.Client.Id	Guid :get=True,set=False	Уникальный идентификатор точки в системе
Serial	PaymentContext.Client.Serial	System.Int32 :get=True,set=True	Уникальный порядковый номер точки в системе
Old	PaymentContext.Client.Old	Boolean :get=True,set=False	Устарел ли объект
Account		**Сложный тип, не используется**	**Счет**
ClientId	PaymentContext.Client.Account.ClientId		Айди клиента использующего этот счет
InputDate	PaymentContext.Client.Account.InputDate	DateTime :get=True,set=False	Время создания счета в системе
HasRealAccount	PaymentContext.Client.Account.HasRealAccount	Boolean :get=True,set=False	Есть ли реальный счет в системе (выгрузка в абс)

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Объект	Путь	Тип	Описание
Balance	PaymentContext.Client.Account.Balance	Decimal :get=True,set=True	Баланс счета
Limit	PaymentContext.Client.Account.Limit	Decimal :get=True,set=True	Лимит счета
Credit	PaymentContext.Client.Account.Credit	Decimal :get=True,set=True	Кредит счета
RealAccount	PaymentContext.Client.Account.RealAccount	String :get=True,set=True	Реальный счет закрепленный за эти м в системе
ItemType	PaymentContext.Client.Account.ItemType	String :get=True,set=False	Тексовое описание типа
HasAttributes	PaymentContext.Client.Account.HasAttributes	Boolean :get=True,set=False	Есть ли доп. атрибуты у счета
Account[string Key	PaymentContext.Client.Account[stringKey]	String :get=True,set=True	Именованная коллекция атрибутов у счета
Count	PaymentContext.Client.Account.Count	Int32 :get=True,set=False	Кол-во доп. атрибутов, у города
Id	PaymentContext.Client.Account.Id	Guid :get=True,set=False	Уникальный идентификатор счета в системе
Serial	PaymentContext.Client.Account.Serial	System.Int32 :get=True,set=True	Уникальный порядковый номер счтеа в системе
Old	PaymentContext.Client.Account.Old	Boolean :get=True,set=False	Устарел ли объект
AllowedServices		**Сложный тип, не используется**	**Список доступных клиенту сервисов**
Collection	PaymentContext.Client.AllowedServices.Collection	System.Collections.Generic.IEnumerable`1[IBP.PaymentProject2.Common.Client+Service] :get=True,set=False	Список разрешенных клиенту сервисов
Count	PaymentContext.Client.AllowedServices.Count	Int32 :get=True,set=False	Кол-во разрешенных клиенту сервисов
Town		**Сложный тип, не используется**	**Город (абс)**
ExportPath	PaymentContext.Town.ExportPath	String :get=True,set=True	Путь для выгрузки проводок
BadAccount	PaymentContext.Town.BadAccount	String :get=True,set=True	Счет невыясненных поступлений
BadDescription	PaymentContext.Town.BadDescription	String :get=True,set=True	Описание проводки на счет невыясненных поступлений
Name	PaymentContext.Town.Name	String :get=True,set=True	Имя города в системе
InputDate	PaymentContext.Town.InputDate	DateTime :get=True,set=False	Время создания проводки в системе
BIC	PaymentContext.Town.BIC	String :get=True,set=True	Банковский иденгтификатор
SystemAccount	PaymentContext.Town.SystemAccount	String :get=True,set=True	Счет платежной системы
BufferAccoun	PaymentContext.Town.BufferAccoun	String :get=True,set=True	Буферный счет
SubsidiaryCode	PaymentContext.Town.SubsidiaryCode	String :get=True,set=True	Код филиала
ProfitDescription	PaymentContext.Town.ProfitDescription	String :get=True,set=True	Описание проводки на счет доходов
ProfitAccount	PaymentContext.Town.ProfitAccount	String :get=True,set=True	Счет доходов
AllDescription	PaymentContext.Town.AllDescription	String	Общее описание всех проводок

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Объект	**Путь**	**Тип**	**Описание**
		:get=True,set=True	между буферным и счетом платежной системы
ItemType	PaymentContext.Town.ItemType	String :get=True,set=True	Текстовое описание типа
HasAttributes	PaymentContext.Town.HasAttributes	Boolean :get=True,set=False	Есть ли доп. атрибуты у города
Town[stringKey]	PaymentContext.Town[stringKey]	String :get=True,set=True	Именованная коллекция атрибутов у города
Count	PaymentContext.Town.Count	Int32 :get=True,set=False	Кол-во доп. атрибутов, у города
Id	PaymentContext.Town.Id	Guid :get=True,set=False	Уникальный идентификатор точки в системе
Serial	PaymentContext.Town.Serial	System.Int32 :get=True,set=True	Уникальный порядковый номер точки в системе
Old	PaymentContext.Town.Old	Boolean :get=True,set=False	Устарел ли объект

Объект	**Путь**	**Тип**	**Описание**
Filter		**Сложный тип, не используется**	**Фильтр выборки платежей**
StartTime	Filter.StartTime	DateTime :get=True,set=True	Время начала временного интервала выборки
EndTime	Filter.EndTime	DateTime :get=True,set=True	Время конца временного интервала выборки
И другие см. в файле			

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Приложение 2. Шлюз реализация.

Техническое задание

Реализовать прокси-шлюз который будет вызывать операции webserver, которые в свою очередь будут реализовывать вызовы конкретного протокола.

Кол–во параметров в запросах переменное в зависимости от протокола должно настраиваться.

Операции:

1. CheckAccount
2. Process
3. CheckProcessStatus
4. RecallPayment
5. CheckRecallStatus

,реализуются вызовом GET запроса, вся необходимая информация в параметрах запроса

Пример:

Запрос: *http://localhost?operation=CheckAccount&Account=123456*

Ответ: *state=7000&description=vipaccount*

Реестр платежей должен передавать на webserver в multipart/form-data в виде xml файла, параметры платежей в реестре должны настраиваться

1. SendRoll — один параметр get + post реестр

Пример:

Запрос: *http://localhost?operation=SendRoll*

+ телом

```
<roll>
<filter starttime="18.12.2007" />
<payments>
        <payment alias="12312321321" value="3232.00" total="3232.00"
time="18.12.2007" />
        <payment alias="12312321321" value="3232.00" total="3232.00"
time="18.12.2007" />
</payments>
</roll>
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Ответ:

1. state=5000&description=rolled — сверилось
2. state=1000&description=notrolled — не сверилось

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Реализация

Настройки шлюза SDKGateway:

Для подключения внешней реализации:

path_ext_dll1 — путь к внешней сборке;

initsettstring1 — путь к файлу инициализации менеджера настроек

Для самого шлюза

URL[n] — адрес

USER[n] — пользователь

PASSWORD[n] — пароль

ENCODING[n] — имя кодировки для передачи запросов.

Файл инициализации менеджера настроек — xml описывающий, какие атрибуты нужно передавать в запросе на каждую операцию, где

<get alias="alias" format="format" path="PaymentContext.Payment.Account" /> — заполнение контекста

<set alias="transid" path="PaymentContext.Payment["TRAN;"]"/> — сохранение данных после выполнения операции

Настройка объектов привязки для шлюза SDKGateway,

1.alias — имя под чем пойдет значение на webserever

2.path — путь для объекта привязки

3.format — формат для приведения объекта в строку(если объект поддерживает форматирование)

Вид файла настройки:

```
<settings>
<operations>
        <CheckAccount>
                <get alias="alias" format="format" path="PaymentContext.Payment.Account" />
        </CheckAccount>
        <Process>
                <get alias="alias" format="format" path="PaymentContext.Payment.Account" />
                <get alias="value" format="f2" path="PaymentContext.Payment.Value" />
                <!—сохраним номер транзакции полученный от получателя -->
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
            <set alias="transid" path="PaymentContext.Payment[&amp;quot;TRAN;&amp;quot;]"/>
        </Process>
        <SendRoll>
            <get alias="alias" format="format" path="PaymentContext.Payment.Account" />
            <get alias="value" format="f2" path="PaymentContext.Payment.Value" />
            <get alias="total" format="f2" path="PaymentContext.Payment.Total" />
            <get alias="time" format="dd.MM.yyyy"
path="PaymentContext.Payment.ServerTime" />
        </SendRoll>
</operations>
<filter>
        <get alias="starttime" format ="dd.MM.yyyy" path ="Filter.StartTime" />
</filter>
```

</settings>

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

Код внешней сборки

```
using System;
using System.Collections.Generic;
using System.Text;
using IBP.SDKGatewayLibrary;
using System.Collections;
using System.IO;
using System.Xml;
using System.Globalization;
using System.Net;

namespace TESTGateway
{
        /// <summary>
        /// Реализация менеджера  внешней настройки
        /// </summary>
        class SettingManager : SettingManagerBase
        {

                public override string[] GetPaymentContextKeys(Operation operation)
                {
                        if (GetKeys[operation] == null)
                                return null;
                        else
                        {

                                List<ValueKey> keys = this.GetKeys[operation];
                                List<string> outkeys = new List<string>();
                                foreach (ValueKey key in keys)
                                        outkeys.Add(key.path);
                                return outkeys.ToArray();
                        }

                }
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
public override string[] SetPaymentContextKeys(Operation operation)
{

        if (SetKeys[operation] == null)
                return null;
        else
        {
                List<ValueKey> keys = SetKeys[operation];
                List<string> outkeys = new List<string>();
                foreach (ValueKey key in keys)
                        outkeys.Add(key.path);
                return outkeys.ToArray();
        }
}

public override string[] GetFilterContextKeys()
{
        List<string> outkeys = new List<string>();
        foreach (ValueKey key in this.rollkeys)
                outkeys.Add(key.path);
        return outkeys.ToArray();
}

public override string[] GetSettingsKey()
{
        return new string[] { "URL", "USER", "PASSWORD", "ENCODING" };
}

public override string[] SaveSettingKey()
{
        return null;
}

public Dictionary<Operation, List<ValueKey>> GetKeys = new Dictionary<Operation,
List<ValueKey>>();
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
        public Dictionary<Operation, List<ValueKey>> SetKeys = new Dictionary<Operation,
List<ValueKey>>();
        public List<ValueKey> rollkeys = new List<ValueKey>();

        public class ValueKey
        {
            public readonly string format;
            public readonly string path;
            public readonly string alias;
            internal ValueKey(string path, string alias, string format)
            {
                this.alias = alias;
                this.format = format;
                this.path = path;
            }
            internal ValueKey(XmlNode node)
            {
                this.alias = node.Attributes["alias"].Value;
                if (node.Attributes["format"] != null)
                    this.format = node.Attributes["format"].Value;
                this.path = node.Attributes["path"].Value;
            }

            public string ValueToString(object value)
            {
                if (value is IFormattable && this.format != null)
                    return ((IFormattable)value).ToString(this.format,
CultureInfo.InvariantCulture);
                else
                    return value.ToString();
            }

        }

        public override string InitSettingManadger(string initstring)
        {
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
StringBuilder _error = new StringBuilder();
if (initstring == null)
{
        _error.AppendLine("Не указан путь к конфигу");
        return _error.ToString();
}
else
{
        XmlDocument doc = new XmlDocument();
        try
        {
                doc.Load(initstring);
        }
        catch (Exception _ex)
        {
                _error.AppendLine("Ошибка загрузки конфига");
                return _error.ToString();
        }
        try
        {
                // операции
                XmlElement operations =
(XmlElement)doc.GetElementsByTagName("operations")[0];
                // разбор операций
                foreach (string opername in Enum.GetNames(typeof(Operation)))
                {
                        XmlElement xmloperation = null;
                        Operation operation = (Operation)Enum.Parse(typeof(Operation),
opername);
                        XmlNodeList operlist =
operations.GetElementsByTagName(opername);
                        if (operlist.Count == 0)
                        {
                                this.GetKeys[operation] = null;
                                this.SetKeys[operation] = null;
                                continue;
                        }
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
                else
                {
                    xmloperation = (XmlElement)operlist[0];

                    XmlNodeList gets =
xmloperation.GetElementsByTagName("get");
                    List<ValueKey> getkeys = null;
                    if (gets.Count > 0)
                        getkeys = new List<ValueKey>();
                    foreach (XmlNode get in gets)
                        getkeys.Add(new ValueKey(get));

                    XmlNodeList sets =
xmloperation.GetElementsByTagName("set");
                    List<ValueKey> setkeys = null;
                    if (sets.Count > 0)
                        setkeys = new List<ValueKey>();
                    foreach (XmlNode set in sets)
                        setkeys.Add(new ValueKey(set));
                    //заполнение
                    this.GetKeys[operation] = getkeys;
                    this.SetKeys[operation] = setkeys;
                }

            }
            // заполнение фильтра
            XmlElement filter = (XmlElement)doc.GetElementsByTagName("filter")[0];
            {
                XmlNodeList gets = filter.GetElementsByTagName("get");
                List<ValueKey> getkeys = new List<ValueKey>(gets.Count);
                foreach (XmlNode get in gets)
                    this.rollkeys.Add(new ValueKey(get));

            }
        }
        catch (Exception _ex)
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
                {
                    _error.Append(_ex.Message + "\r\n" + _ex.StackTrace);
                    _error.ToString();
                }
            }
            if (_error.Length > 0)
                throw new ApplicationException(_error.ToString());
            // сохраним ссылку для использования в шлюзе
            SettingManager.SM = this;


        }

        public static SettingManager SM = null;

    }


    /// <summary>
    /// Реализация шлюза
    /// </summary>
    class GatewayCore : GatewayCoreBase
    {
        /// <summary>
        /// Конфиг шлюза
        /// </summary>
        class Config
        {
            public readonly string URL;
            public readonly string USER;
            public readonly string PASSWORD;
            public Encoding Enc;
            public Config(string Url, string User, string Password, string Enc)
            {
                this.URL = Url;
                this.USER = User;
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
                this.PASSWORD = Password;
                Encoding _enc = Encoding.GetEncoding(Enc);
                this.Enc = (_enc != null) ? _enc : Encoding.GetEncoding(1251);

        }

}

private Config GWConfig = null;

public override void InitGateway(Hashtable settigs)
{
        string URL = (string)settigs["URL"];
        string USER = (string)settigs["USER"];
        string PASSWORD = (string)settigs["PASSWORD"];
        string ENCODING = (string)settigs["ENCODING"];
        this.GWConfig = new Config(URL, USER, PASSWORD, ENCODING);

}

public override System.Collections.Hashtable SaveSettings()
{
        return null;
}

/// <summary>
/// Послать запрос на сервер и получить ответ
/// </summary>
/// <param name="request"></param>
Response SendHttpRequest(string requeststring, MultipartFormData roll)
{
        string _respstring = null;
        string _contenttype = (roll == null) ? null : roll.ContentType;
        HttpWebRequest _webreq = this.GetRequest(requeststring, _contenttype);

        if (roll != null)
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
                {
                        byte[] data = roll.GetData(Encoding.GetEncoding(1251));

                        using (Stream _requestStream = _webreq.GetRequestStream())
                        {
                                IAsyncResult _ar = _requestStream.BeginWrite(data, 0, data.Length, null,
null);

                                _requestStream.EndWrite(_ar);
                        }

                }

                HttpWebResponse _webresp = (HttpWebResponse)_webreq.GetResponse();
                using (StreamReader _reader = new StreamReader(_webresp.GetResponseStream(),
Encoding.ASCII))
                {
                        _respstring = _reader.ReadToEnd();
                }
                Response _resp = new Response(_respstring);
                return _resp;
        }

        // Создать  web запрос
        HttpWebRequest GetRequest(string requeststring, string contenttype)
        {
                IWebProxy _proxy = WebRequest.GetSystemWebProxy();
                _proxy.Credentials = CredentialCache.DefaultCredentials;
                HttpWebRequest _request = (HttpWebRequest)WebRequest.Create(this.GWConfig.URL + "?"
+ requeststring);
                _request.ContentType = (contenttype != null) ? contenttype : "application/x-www-form-
urlencoded";
                _request.Method = (contenttype != null) ? "POST": "GET";
                _request.Proxy = _proxy;
                _request.PreAuthenticate = false;
                _request.Timeout = -1;
                _request.Credentials = new NetworkCredential(this.GWConfig.USER,
this.GWConfig.PASSWORD);
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
            return _request;
}

/// <summary>
/// Парсер ответа
/// </summary>
class Response
{

            const string STATUS = "state";
            const string DESCRIPTION = "description";

            Dictionary<string, string> _attributes = null;
            State status;
            string discription;

            private string respstring = null;

            internal Response(string respounce)
            {
                        this.respstring = respounce;
                        try
                        {
                                    _attributes = this.GetResponseAttributes(this.respstring);

                                    if (!_attributes.ContainsKey(STATUS))
                                                throw new FormatException("Неверный ответ сервера.");
                                    this.status = (State)int.Parse(_attributes[STATUS]);

                                    if (!_attributes.ContainsKey(DESCRIPTION))
                                                throw new FormatException("Неверный ответ сервера.");
                                    this.discription = _attributes[DESCRIPTION];
                        }
                        catch (FormatException)
                        {
                                    this.status = State.DenialOfService;
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
                        this.discription = "Неверный олтвет сервера";
                    }
                }

                Dictionary<string, string> GetResponseAttributes(string responseString)
                {
                    string[] _buff = responseString.Split(new char[] { '&' },
StringSplitOptions.RemoveEmptyEntries);
                    Dictionary<string, string> _attributes = new Dictionary<string,
string>(_buff.Length);

                    foreach (string _pair in _buff)
                    {
                        string[] _attr = _pair.Split(new char[] { '=' },
StringSplitOptions.RemoveEmptyEntries);
                        if (_attr.Length < 1 || _attr.Length > 2)
                            throw new FormatException("Неверный ответ сервера.");
                        _attributes[_attr[0]] = _attr.Length == 2 ? _attr[1] : "";
                    }

                    return _attributes;
                }
                /// <summary>
                /// Статус
                /// </summary>
                public State Status { get { return this.status; } }
                /// <summary>
                /// Описание статуса
                /// </summary>
                public string Description { get { return this.discription; } }

                /// <summary>
                /// Доп атрибуты платежа
                /// </summary>
                /// <param name="key"></param>
                /// <returns></returns>
                public string this[string key]
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
        {
            get { return this._attributes[key]; }
        }
}

/// <summary>
/// Получить строку
/// </summary>
/// <param name="operation">тип операции</param>
/// <param name="context">контекст операции</param>
/// <returns> строка запроса</returns>
private string GetRequestString(Operation operation, Context context)
{
        StringBuilder _reqstring = new StringBuilder();
        _reqstring.AppendFormat("operation={0}&",
            operation.ToString()
        );

        foreach (SettingManager.ValueKey value in SettingManager.SM.GetKeys[operation])
            _reqstring.AppendFormat("{0}={1}&",
                value.alias,
                value.ValueToString(context[value.path])
                );
        return _reqstring.ToString();
}
/// <summary>
/// Выполнить операцию
/// </summary>
/// <param name="operation">Тип операции</param>
/// <param name="context">Контекст опреации</param>
private void ExecuteOperation(Operation operation, Context context)
{
        string _reqstring = this.GetRequestString(operation, context);
        Response _resp;
        try
        {
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
                _resp = this.SendHttpRequest(_reqstring, null);
            }
            catch (Exception _ex)
            {
                // если произошло исключение статус не меняем
                string _respstring="state={0}&description=" + _ex.Message;

                switch (operation)
                {
                    case Operation.CheckAccount:
                        _respstring = string.Format(_respstring,
((int)State.DenialOfService).ToString(CultureInfo.InvariantCulture));
                        break;
                    default:
                        _respstring = string.Format(_respstring,
((int)context.Status).ToString(CultureInfo.InvariantCulture));
                        break;

                }
                _resp = new Response(_respstring);
            }
            //обнуляем контекст чтобы обратно не передовать лишние данные

            context.Clear();
            context.Status = _resp.Status;
            context.Description = _resp.Description;

            // сохранение настроек в системе
            List<SettingManager.ValueKey> setkeys = SettingManager.SM.SetKeys[operation];

            if (setkeys != null)
            {
                foreach (SettingManager.ValueKey value in setkeys)
                {
                    context[value.path] = _resp[value.alias];
                }
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
        }
}
/// <summary>
/// Выполнить операцию посылки реестра
/// </summary>
/// <param name="roll"></param>
/// <returns></returns>
private Response ExecuteRollOperation(MultipartFormData roll)
{
        return this.SendHttpRequest("operation=SendRoll", roll);
}



// реализация методов шлюза

public override void Process(ref Context context)
{
        this.ExecuteOperation(Operation.Process, context);
}

public override void CheckAccount(ref Context context)
{
        this.ExecuteOperation(Operation.CheckAccount, context);
}

public override void RecallPayment(ref Context context)
{
        this.ExecuteOperation(Operation.RecallPayment, context);
}

public override void CheckProcessStatus(ref Context context)
{
        this.ExecuteOperation(Operation.CheckProcessStatus, context);
}
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
        public override void CheckRecallStatus(ref Context context)
        {
                this.ExecuteOperation(Operation.CheckRecallStatus, context);
        }

        public override bool SendRoll(Context[] payments, Hashtable filtercontext)
        {
                StringBuilder _roll = new StringBuilder();
                _roll.AppendFormat("<roll><filter ");

                foreach (SettingManager.ValueKey value in SettingManager.SM.rollkeys)
                {
                        _roll.Append(value.alias + "=\"" + value.ValueToString(filtercontext[value.path]) +
"\" ");
                }
                _roll.Append(" />");
                _roll.Append("<payments> ");
                foreach (Context payment in payments)
                {
                        _roll.Append("<payment ");

                        foreach (SettingManager.ValueKey value in
SettingManager.SM.GetKeys[Operation.SendRoll])
                        {
                                _roll.Append(value.alias + "=\"" +
value.ValueToString(payment[value.path]) + "\" ");
                        }
                        _roll.Append(" /> ");
                }
                _roll.Append("</payments></roll>");

                MultipartFormData formdata = new MultipartFormData();
                formdata.AddFile("roll", "roll", this.GWConfig.Enc.GetBytes(_roll.ToString()));
                Response _resp = this.ExecuteRollOperation( formdata);
                return (_resp.Status == State.Finalized);
        }
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
public class MultipartFormData
{
	string boundary;
	StringBuilder fields = new StringBuilder();
	string fieldName;
	string fileName;
	byte[] fileData;

	public MultipartFormData()
	{
		this.boundary = DateTime.Now.Ticks.ToString();
	}

	public string ContentType { get { return "multipart/form-data; boundary=" + this.boundary; }
}


	/// <summary>
	/// Boundary
	/// </summary>
	public string Boundary { get { return this.boundary; } }

	/// <summary>
	/// Добавляем поле
	/// </summary>
	/// <param name="fieldName">имя поля</param>
	/// <param name="fieldValue">значение</param>
	public void AddField(string fieldName, string fieldValue)
	{
		this.fields.AppendFormat(
			"--{1}{0}Content-Disposition: form-data; name=\"{2}\"{0}{0}{3}{0}",
			"\r\n",
			this.boundary,
			fieldName,
			fieldValue
		);
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
        }
        /// <summary>
        /// Добавляем поток(файл)
        /// </summary>
        /// <param name="fieldName">имя поля</param>
        /// <param name="fileName">имя файла</param>
        /// <param name="data">поток файла</param>
        public void AddFile(string fieldName, string fileName, byte[] fileData)
        {
                this.fieldName = fieldName;
                this.fileName = fileName;
                this.fileData = fileData;
        }

        /// <summary>
        /// получить данные для отправки
        /// </summary>
        /// <returns></returns>
        public byte[] GetData(Encoding encoding)
        {
                this.fields.AppendFormat(
                        "--{1}{0}Content-Disposition: form-data; name=\"{2}\";
filename=\"{3}\"{0}Content-Type: text/xml{0}{0}",
                        "\r\n",
                        this.boundary,
                        fieldName,
                        fileName
                );
                string _head = this.fields.ToString();
                this.fields = new StringBuilder();
                string _tail = string.Format("{0}--{1}--", "\r\n", this.boundary);

                //резервируем место под результат
                byte[] _result = new byte[
                        this.fileData.Length +
                        encoding.GetByteCount(_head) +
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
                                        encoding.GetByteCount(_tail)
                                    ];

                                    //копируем в массив результат
                                    int _bytesCopied = encoding.GetBytes(_head, 0, _head.Length, _result, 0);
                                    Array.Copy(this.fileData, 0, _result, _head.Length, this.fileData.Length);
                                    _bytesCopied += this.fileData.Length;
                                    encoding.GetBytes(_tail, 0, _tail.Length, _result, _head.Length +
this.fileData.Length);

                                    return _result;
                                }
                            }
                            public override void Dispose()
                            {

                            }
                        }
}
```

Пример скрипта на котором тестировался шлюз

```
<?
$result = "";
$fp = fopen ("readme", "a");
fwrite($fp,$_GET["operation"]."\r\n");
switch($_GET["operation"])
{
 case 'CheckAccount':
   $result="state=7000&description=Vip Account";
   break;
 case 'Process':
          $result="state=5000&description=принял&transid=21";
   break;
 case 'CheckProcessStatus':
         $result="state=5000&description=123213";
   break;
```

38, Zhanar street,
Nur-Alatau mcrd.,
Almaty, 050045,
Republic of Kazakhstan
email: sales@tentencode.com

```
 case 'RecallPayment':
        $result="state=1000&description=123213";
  break;
 case 'CheckRecallStatus':
        $result="state=1000&description=123213";
  break;
 case 'SendRoll':
        fwrite($fp,"File Name : ".$_FILES['roll']['name']."\r\n");
        if (is_uploaded_file($_FILES['roll']['tmp_name']))
        {
          if (move_uploaded_file($_FILES['roll']['tmp_name'],"files/".$_FILES['roll']['name']))
                {
                        $result="state=5000&description=Rolled";
                }
                else
                        $result="state=1000&description=Error";
        }
        else
                $result="state=1000&description=Error";

        break;
}
fwrite($fp,$result."\r\n");
fclose($fp);
echo $result;
?>
```