Шлюз электронного
Бронирования и продажи билетов

ООО "Дилявер Технолоджи"
Москва 2017г.

Оглавление

1 Введение

Введение

1. Данный шлюз реализован в виде SOAP Server «CGI Stand-alone Executable». Такой формат реализации выбран для поддержания возможности использования как в MS Internet Information Server, так и Apache 2.0+.
2. Все функциональные методы реализованы в виде метода интерфейса SOAP. Каждый метод реализуется в виде функции стандартного формата (Pascal синтаксис):
   function MethodName(strInp: string): string; stdcall;
3. Входной параметр метода **strInp** – строка в формате XML-формата, отличающемся для каждого метода набором параметров.
4. Каждая функция возвращает строку в XML-формате, специфическом для каждого метода. Возвращаемая строка содержит как код возврата и описание возможной ошибки, так и возвращаемые данные.
5. Если какой-либо из входных параметров тега **InputRow**(см. далее) не задается, то он может быть указан как пустая строка ("") или не указан вовсе.
6. Удаленный пользователь авторизуется по указанному во входящем запросе логину и паролю.
7. Порядок атрибутов тегов <**InputRow**> (входные параметры) и <**Row**> (выходные параметры) – произвольный. Атрибуты должны обрабатываться по имени. Регистр входных параметров не имеет значения. На практике, **порядок выходных параметров (атрибутов тегов) может незначительно отличаться от приведенного в данном руководстве.**
8. **Список выходных параметров методов может количественно меняться в большую сторону без предварительного уведомления пользователей. Приведенные в данном руководстве входные параметры могут быть изменены только после предварительного уведомления пользователей!** Вся информация об изменениях размещается в разделе История
9. **В нынешней версии шлюз является только связующим звеном между сайтом и БД билетной системы Profticket®. Методы шлюза позволяют получить информацию о местах проведения мероприятий, репертуаре, наличии свободных мест и т.п. Кроме того, шлюз позволяет забронировать и отметить факт продажи билетов. Обработка, внешний вывод данных, а также подключение к платежной системе, целиком относится к**

подключаемому сайту.

10. Пример строки подключения к шлюзу: http://187.198.34.12/scripts/ProfTicketGate.exe/wsdl/IProfTicket.

187.198.34.12 - адрес шлюза.

/scripts/ папка, где находится исполняемый модуль.

ProfTicketGate.exe сам исполняемый модуль.

/wsdl/IProfTicket - ссылка на соответствующий WSDL.

В некоторых случаях вместо wsdl нужно использовать ***soap*** *(*http://187.198.34.12/scripts/ProfTicketGate.exe/soap/IProfTicket*).*

Это зависит от языка разработки сайта или программы, которые подключаются к шлюзу.

1.1 Алгоритм вызова методов для осуществления бронирования и продажи.

Алгоритм осуществления бронирования и продажи.

- Для осуществления бронирования и продажи необходимо получить информацию о свободных местах на мероприятия. Для этого воспользуйтесь методом **GetEvailPlaceList**.
- Затем необходимо осуществить предварительное бронирование мест, для того, чтобы кассир театра или другой шлюз не могли работать с выбранными местами. Предварительное бронирование осуществляется методом **PreSetReservation**. Отметка о предварительном бронировании устанавливается на определенное время (по умолчанию 15 мин)
- При отказе от дальнейшей работы с местом можно снять отметку о предварительном бронировании можно методом **FreePreSetReservation**.
- Далее места необходимо забронировать, **даже если затем они будут проданы**. Для этого используется метов **SetReservation**.
- **Использование CheckSoldTickets, не является обязательным, но крайне желательным** и позволит избежать ошибок при оплате. Этот метод проверяет наличие доступных для оплаты мест. Сообтветственно он должен быть вызван непосредственно перед отправкой данных в платежную систему.
- При отказе от дальнейшей работы с местом можно снять отметку о бронирование места с помощью метода **FreeReservation**.
- После того, как места забронированы можно их продать.
- Метод **SetSold** устанавливает метку о продаже места по безналичному расчету (кредитной карте) зрителю, на которого место было забронировано.
- Проверить состояние места и историю по данному месту можно следующими методами **GetLog**, и **GetCurrentState**.
- Проверить статус заказа можно с помощью метода GetReservationStatus.

Обратите внимание!
Указаный выше порядок операций является обязательным. Места не могут быть забронированы (и проданы) без процедуры предварительного бронирования, а затем бронирования (т.е. создания

заказа).

Также при работе сайта с платежной системы следует использовать указанный порядок вызова методов. Необходимо осуществлять оплату через платежную систему только после получения номера заказа (вызова метода SetReservation).

1.2 Основные термины и понятия

Основные термины и понятия

- **Место проведения** - Театр, концертный зал и т.д. в котором проходит мероприятие (пример: "Театр Моссовета, Спорткомплекс Олимпийский")

- **Сцена театра** - Сцена театра, в котором проходит мероприятие (пример: "Основная сцена", "Малая сцена")

- **Мероприятие** - Спектакль, концерт, балет и т.д. проходящие в театре.

- **Событие** - "Единица" репертуара - спектакль (концерт, балет и т.д.) проводимый в определенное время на определенной сцене театра.

- **Сектор** - Определенная часть зрительного зала (пример: Партер, Балкон, Бельэтаж, Ложа).

- **Предварительное бронирование (резервирование)** - Установка специального статуса для места (или мест), свидетельствующего о том, что место (места) находятся в работе и остальные пользователи не могут их использовать.

- **Бронирование (Резервирование)**- Создание заказа с присвоением ему номера. В заказ включаются переданные соответствующим методом шлза места.

- **Сессия** - идентификатор сессии. Как правило, используется ID сессии вебсервера. Данный параметр необходим, чтобы шлюз мог отличать соединения. Места объединяются в один заказ в рамках сессии. После оформления заказа или после продажи необходимо сменить номер сессии.

- **Ряд** - ряд передается строкой (макс. длина 4 символа), т.к. встречаются литерные ряды (пример: A, B, C)

- **Место** - место передается строкой (макс. длина 4 символа), т.к. встречаются литерные места (пример: 1A, 2B, 3C)

- **ID транзакции** - Идентификатор транзакции, передаваемый платежной системой. В дальнейшем по этому номеру можно отследить прохождение оплаты и т.д.

- **Дата и время оплаты** - Дата и время прихода извещения платежной системы об оплате.

- **Квота интернета** - места, на которые разрешена продажа через шлюз.

- **Свой сайт** - "Своим сайтом" называется сайт принадлежащий организации, продающей билеты и которой принадлежит сам шлюз.

- **Сторонний сайт** - сайт сторонней организации, которая выступает в качестве агента и продает мероприятия театра

- **Цена билета** - У билета есть две цены - цена номинальная и цена продажи. Для театральной версии и для стороннего сайта обе цены равны. В агентской версии цена продажи может отличаться от номинальной цены на размер агентского вознаграждения.

- **Пагинация** - Постраничный вывод списка записей.

- **Связанное событие (услуга)** - событие, которое привязано к "родительскому" событию. Связанным событием может быть как не зрилищное мероприятие, (подарки, програмки, парковка и т.д.), так и обыкновенное мероприятие. События могут быть привязаны только к событиям, у которых нет признака "услуга" (isService). Событие-услуга может продаваться как отдельно, так и вместе с родительским событием.

1.3 Форматы данных

Форматы данных

1. **Дата**. Даты в строках формата XML представляются строкой в формате "dd.mm.yyyy" вне зависимости от настроек локализации сервера или клиента. Пустая дата – "01.01.1900". Диапазон дат: минимальная 01.01.1753, максимальная 12.31.9999.
2. **Время**. Формат - "hh:mm:ss"
3. **Цена**. Цена в строках XML представляется строкой формата ‘0.00’ – вне зависимости от локализации клиента и сервера. Разделитель – точка.
4. **Координаты**. Координаты в строках XML представляется строкой формата ‘0.0000’ – вне зависимости от локализации клиента и сервера. Разделитель – точка.

1.4 Коды возврата.

Коды возврата.

-1 - Частично успешное действие. Используется в качестве результата всей операции. Означает, что часть запроса выполнена упешно, а часть нет.

0 – успешное действие (может использоваться в качестве результата всей операции и по каждому элементу коллекции);

1 – место занято или отсутствует (может использоваться в качестве результата по каждому элементу коллекции), событие не активно (прошло, отменено, не разрешена продажа);

2 – ошибка интерфейса, неверный формат данных (может использоваться в качестве результата всей операции и по каждому элементу коллекции);

3 – нет ответа, ошибка доступа к базе данных (может использоваться в качестве результата всей операции и по каждому элементу коллекции);

4 – неверный номер сессии или снята предварительная бронь (может использоваться в качестве результата по каждому элементу коллекции);

5 – ошибка авторизации (может использоваться в качестве результата всей операции);

6 – системная ошибка исполнения шлюза;

7 – "Нет такого зрителя" (может использоваться в качестве результата по каждому элементу коллекции);

8 – Не заполнено поле «ФИО зрителя» (может использоваться в качестве результата по каждому элементу коллекции);

9 – Не заполнены контактные данные зрителя (телефон или email) (может использоваться в качестве результата по каждому элементу коллекции);

10 – Не указана транзакция.

11 – Не указана дата транзакции

12 – Не указано время транзакции

13 – Заказ оплачен, место не подлежит удалению.

14 – Пользователь не является пользователем шлюза.

15 – Пользователю запрещен доступ к шлюзу.

16 – Пользователь является пользователем шлюза, но не собственного сайта.

17 – У одной или нескольких записей указан неверный или пустой номер сессии.

19 – Одно или несколько мест не соответствуют заказу, не находятся в брони, заняты другим пользователем или не существуют.

20 – Цена билета не соотвествует имеющейся.

21 – Неправильные или пустные данные XML.

22 – В переданных данных содержится информация о нескольких зрителях.

23 – Пользователь, осуществивший оформление заказа или продажу, не соответствует пользователю шлюза.

24 – Одно или несколько мест заказа оплачено за наличный расчет. Оплата остальных мест заказа не может быть осуществлена.

25 – Не указан адрес доставки.

26 – Не указана дата доставки.

27 – Текущая дата больше даты мероприятия или меньше на 1 день*.

28 – Текущая дата больше даты, после которой бронирование на данное событие запрещено.

29 – Текущая дата больше даты возврата билетов.

30 – Не указано условие доставки.

31 – Не указан признак доставки, но указаны дата доставки или адрес доставки или условие доставки

32 - Одно или несколько мест являются **непроданным**.

33 - На место распечатан билет. Возврат разрешен только через кассу.

34 - Место является абонементом и не может быть забронировано/ продано, т.к. в мероприятиях, присоединенных к абонементу есть занятые места.

35 - Информация недоступна для данного пользователя.

36 - По указанной сессии уже был сформирован заказ и сняты или возвращены места. Использование указанной сессии для повторной операции запрещено.

37 - Часть выбраных мест относится к комплекту. Разрешено выбирать одновременно или все места из комплекта или все места не из комплекта.

38 - Среди переданных мест есть места из разных комплектов.

39 - Комплект не существуют.

40 - Комплект не активен.

41 - В списке мест есть места из мероприятий, не подлежащих объединению в заказе.

42 - Промокод не валиден (Не активный, не на выбранное событие, срок его действия истек или он не относится к указанному событию) .

43 - Цена места с учетом промокода рассчитана неверно.

44 - К месту (местам) уже применен промокод при оформлении заказа.

45 - Осуществлен проход зрителя. Возврат невозможен.

** Если в настройках установлено, что нельзя возвращать через шлюз билеты за день до даты (без учета времени) мероприятия.*

Если зафиксирована ошибка, связанная с общей структурой входного XML-документа,
с логином и паролем, то код возврата указывается в теге ResultCode (в качестве результата всей операции).

Если происходит системная ошибка или ошибка формата (нет необходимого параметра или его тип не может быть преобразован к требуемому) при обработке одного элемента входной коллекции, то в результирующем документе код ошибки Result_code возвращается в атрибутах строки выходной коллекции. При этом в строке ответа возвращается полный набор входных параметров (атрибутов). Если ошибки при обработке строки не возникло, то Result_code="0" или не указывается, а остальные атрибуты выходной строки формируются в соответствии с указанной ниже спецификацией.

Если при запросе справочников тег <**InputRow** />;в запросе был указан больше одного раза, то будет возращено конкатенация соответствующих наборов коллекций.

Аналогично, при запросах с параметрами, несколько тегов <**InputRow** /> обрабатываются независимо друг от друга, и результатом будет сумма коллекций.

1.5 Операции шлюза

Операции шлюза

Указанный ниже перечень операций шлюза используется методом GetLog для описание операций шлюза:

- **NoPlacePreRes** – не найдено место для предварительной брони,
- **ErrorPreRes** – при установке предварительной брони произошла ошибка (был откат транзакции),
- **FailPreRes** – при установке предварительной брони ошибки не было, однако место отмечено не было,
- **PreRes** – предварительная бронь успешно установлена,
- **NoSessionFreePreRes** – при снятии предварительной брони не найден указанный номер сессии для указанного события,
- **NoPlaceFreePreRes** - при снятии предварительной брони место не найдено,
- **FailFreePreRes** – ошибка при снятии предварительной брони,
- **FreePreRes** – успешное снятие предварительной брони,
- **NoSessionSetRes** – при бронировании не найден указанный номер сессии для указанного события,
- **NoPlaceSetRes** – не найдено место при бронировании,
- **FailSetRes** – не удалось забронировать место (не было предварительного бронирования или неверная сессия),
- **ErrorSetRes** – ошибка при бронировании,
- **SetRes** – успешная установка брони,
- **NoSessionFreeRes** – при снятии бронирования не найден указанный номер сессии для указанного события,
- **NoPlaceFreeRes** – не найдено место при снятии бронирования,
- **FailFreeRes** – не удалось разбронировать место (не было предварительного бронирования или неверная сессия),
- **ErrorFreeRes** – ошибка при снятии брони,
- **NoReservationFreeRes** – нет брони для указанного места,
- **FreeRes** – снятие брони с места,
- **NoSessionSetSold** – при отметке о продаже не найден указанный номер сессии для указанного события,
- **NoReservationSetSold** – нет брони для указанного места при отметке о продаже,
- **NoPlaceSetSold** – не найдено место при отметке о продаже,
- **ErrorSetSold** – ошибка при установке отметке о продаже,
- **FailSetSold** – не удалось поставить отметку о продаже, т.к. не тот

номер сессии или бронь с указанного места снята,

- **SetSold** – успешная установка отметки о продаже.
- **SetReturn** – успешный возврат.
- **ErrorSetReturn** – неуспешный возврат.
- **NoSession** – Нет такой сессии.

1.6 Статусы заказа

Статусы заказа		
Номер статуса	**Статус**	**Описание**
0	В обработке	*Заказ принят.*
1	На доставке	*Заказ распечатан и передан в службу доставки*
2	Передан на доставку	*Заказ передан курьеру для доставки*
3	Заказ доставлен	*Заказ доставлен зрителю*
4	Снят заказ.	*Заказ аннулирован*
5	Оплачен по б/н. Билеты не переданы клиенту.	*Заказ оплачен по безналичному расчету. Бланки билетов еще не распечатаны.*
6	Оплачен. Билеты переданы клиенту	*Заказ оплачен за наличный расчет. Бланки билетов распечатаны.*
7	Билеты распечатаны. Заказ ожидает передачу на доставку.	*Заказ распечатан полностью. Ожидается передача курьеру на доставку*
8	Заказ снят с доставки. Билеты не распечатаны.	*Заказ снят с доставки (= установлен на самовывоз), но не аннулирован.*
9	Заказ снят с доставки. Билеты распечатаны.	*Заказ снят с доставки (= установлен на самовывоз), но не аннулирован. Билеты распечатаны.*
10	Заказ ожидает оплату.	*Заказ принят и блокирован на время ожидания оплаты с сайта.*
11	Заказ оплачен по б/н. Билеты распечатаны	*Заказ оплачен по б/н. Билеты распечатаны и переданы клиенту*
12	Билеты распечатаны. Заказ передан на доставку.	*Заказ распечатан полностью и передан на доставку курьеру*
13	Заказ был оплачен по б/н расчету. Билеты не распечатаны. Один или несколько билетов заказа были возвращены.	Заказ оплачен по безналичному расчету. Бланки билетов не распечатаны. Часть (или все) мест из были возвращены.
14	Заказ был оплачен по б/н расчету. Билеты распечатаны. Один или несколько билетов заказа были возвращены	Заказ оплачен по безналичному расчету. Бланки билетов распечатаны. Часть (или все) мест из были возвращены.

1.7 Распространенные ошибки

Распространенные ошибки

1. Нарушение порядка вызова методов.

Наиболее распространенной ошибкой является нарушение порядка вызова методов.

Бывает несколько вариантов этой ошибки.

- Метод SetReservation вызывается уже после оплаты через платежную систему, что не допустимо.
- Метод FreeReservation вызывается вместо FreePreReservation,
- Метод FreePreReservation вызывается вместо FreeReservation
- Метод SetReturn вызывается для возврата мест из заказа вместо FreeReservation

2. Часто атрибуты методов указываются в неправильном регистре. Обращаем ваше внимание на то, что наименования атрибутов и самих тегов является регистрозависимыми.

3. Со стороны сайта не обрабатываются возвращаемые коды ошибок.

4. Частая ошибка любых программистов - копирование / вставка без проверки вставляемой информации.

1.8 История

История

25.01.2017+ Добавлен новый метод GetReservationStatus2 - статус заказа(ов) с учетов снятия.

24.01.2017+ В методе GetEventList добавлены новые поля:
EventsSetID - Идентификатор комплекта
EventsSetEvents - Кол-во событий
EventsSetName - Наименование комплекта

17.01.2017+ В методе GetShowListEventsList добавлены параметры:
ETicketPermitted - разрешение на проход по электронному билету,
cod_t - ID места проведения
cod_h - ID зала места проведения.

15.12.2016Обновлены методы работы с комлектами.
Добавлена работа с промокодами в комлектах.

10.11.2016+ Добавлен новый код возврата 45.

13.10.2016+ Добавлен метод CheckPromoCode для проверки валидности промокодов
+ В методы SetReservation и SetSold добавлены необязательные поля PromoCodeID для работы с промокодами

29.09.2016+ В метод GetEventList добавлены следующие поля:
isService - является ли событие услугой,
LinkedServicesExists - есть ли связанные события у данного события,
PromoExists - есть ли на событие промокоды
+ Добавлен новый метод GetLinkedEventsList - список связанных событий.
+ В метод GetSchemaHallList добавлены следующие поля: entranceId и entranceName - ID и наименование подъезда. До этого эти поля были доступны в методах GetEvailPlaceList и GetPlaceForSector

27.06.2016+ Добавлен новый код возврата - 41.
+ Добавлено поле MultiSelectAllowed в методе GetEventList.
+ Добавлены поля ReservDate и ReservTime в методе GetReservationStatus

20.04.2016 + В метод GetReservationStatus добавлены следующие поля: Qty и Amount - кол-во билетов и сумма заказа.

21.03.2016 + В метод GetEventList добавлены следующие поля: StopOrderTime, StopSaleDate, StopSaleTime, изменено описание метода.

17.02.2016 + Добавлен метод GetShowListEventsList - список комлектов.
+ Добавлены методы PreSetReservationEventsSet, SetReservationEventsSet, SetSoldEventsSet, FreePreReservationEventsSet, FreeReservationEventsSet для работы с местами комплектов.
+ Добавлены новые методы: GetSchemaHallListEventsSet, GetEvailPlacesEventsSet для отображения схемы зала комплекта и списка свободных мест соответственно.
+ Добавлены поля FreePlacesEventsSet, EventsSetID, EventsSetEvents, EventsSetName в метод GetEventList
+ Добавлены поля EventsSetID, isEventSet в метод GetEvailPlacesList
+ Добавлены поля EventsSetID, isEventSet в метод GetPlaceForSector
+ Обновлен список кодов возврата.

23.11.2015 + В методе GetSchemaHallList добавлен вывод двух новых объектов: значка сцены и сектора со свободной рассадкой.

06.11.2015 + Дополнено описание метода GateLog.

03.11.2015 +Добавлен новый параметр SeasonTicketId в метод GetEventList.

02.11.2015 +Добавлен новый параметр SalesOnly в метод GateLog.
Оптимизированы некоторые процедуры шлюза.

15.10.2015 + Добавлены парметры OffSet и Limit, а также обновлен список выходных параметров в методе GetShowList
+ Добавлены параметры cod_show, DateBegin, DateEnd, OffSet и Limit GetEventList

13.10.2015 + Обновлен метод GetCurrentState

12.10.2015 + Добавлен метод GetReservationPlaceOrder
+ Проведены работы по оптимизации некоторыз методов

24.08.2015+ Добавлен параметр SendAdvertising в методе SetReservation

13.07.2015+ Добавлен параметр Date в методе GetEventList

08.07.2015+ Добавлено описание распространенных ошибок

07.07.2015+ Добавлен параметр ShowLabels в методе GetSchemaHallList
- Исключен метод GetLogExt как неиспользуемый.
+ Добавлены параметры в метод GetLog

03.07.2015+ Добавлен параметр cod_show в методе GetShowList
+ Добавлен параметр NomBilKn в методе GetEventList

04.06.2015+ Добавлен метод GetCitiesList

1.9 Абонементы

Абонементы

Для работы с абонементами в репертуаре заводится «событие-абонемент».
В списке событий, возвращаемых методом GetEventList, оно передается с параметром **SeasonTicket**=1.
На этом событии проходят все операции по бронированию и продаже абонементов.

Так как в абонемент входит несколько событий, они также отражены в GetEventList (даже если продажа на эти событие закрыта) и имеют параметр SeasonTicket=2.

И «событие-абонемент», и событие, относящееся к нему, содержат также:
идентификатор абонемента – SeasonTicketID ,
кол-во мероприятий в абонементе – SeasonTicketEvents.

Для событий, к абонементам не относящихся, все перечисленные параметры равны нулю.
Место, забронированное или проданное на «событие-абонемент», отмечается как занятое на всех событиях абонемента.

1.10 Комплекты

Комплекты мероприятий

Комлекты мероприятий очень похожи на абонементы. Есть только несколько принципальных отличий.
1. Событие, относящееся к комплекту может содержать как места относящиеся к комплекту, так и обычные места.
2. Электронные билеты оформляются на каждое место в каждом событии, а не 1 билет на все, как в абонементе.
3. Нет "головного" события. Все события комплекта равноправны между собой.
4. Для предварительного резервирования, бронирования и продажи мест комплектов введены новые методы:
Предварительное резервирование: SetPreReservationEventsSet
Отмена предварительного резервирования: FreePreReservationEventsSet
Резервирование: SetReservationsEventsSet
Отмена резервирования: FreeReservationsEventsSet
Продажа SetSoldEventsList
5. В методы предварительного резервирования, бронирования и продажи мест комплектов, а также отмены предварительного резервирования и бронирования передается ID комплекта, а не ID события. В некоторых случаях ошибок или невозможности проведения операции возвращается номер комлекта, сектор, ряд, места, в остальных случаях или в случае удачного проведения операции возвращается весь список мест по комплекту, на все его события.
6. Операции с местами комплекта могут осуществляться только до начала самого раннего из событий комплекта.
7. Возвращать билеты (SetReturn) можно на любое событие комлекта по отдельности, например, на 1 вернуть, а на два нет.

1.11 Промокоды

Промокоды

Промокод позволяет получить скидку на билеты при создании заказа или его оплате.
Промокоды применяются к отдельным событиям. Таким образом в одном заказе могут быть места и с примененным промокодом и без него.
При бронировании билетов на несколько мероприятий зритель может указать более одного промокода . На отдельные события могут действовать несколько промокодов, в т.ч. и указанных зрителем. Из указанных зрителем промокодов необходимо выбрать промокод с наибольшим процентом скидки.
Промокод имеет срок действия (актуальности).
Промокод состоит из набора цифр и (или) букв русского или английского алфавита. Максимальная длина - 10 символов.
Если промокод применен при создании заказа, то он автоматически применяется при оплате. Дополнительный промокод в процессе оплаты при этом вводить запрещено.
Если в заказе был указан промокод и срок действия этого промокода истек к моменту оплаты, то при оплате скидка все равно будет применена.
Если заказ с промокодом был создан через шлюз, а билет выкупается в кассе, то при выкупе применяется указанный в заказе промокод.

Для проверки валидности промокода используется метод CheckPromoCode.
При создании заказа (SetReservation), если необходимо передать промокод, передается его ID в поле PromoCodeID.
Такой же параметр используются при продаже (SetSold) по промокоду.

2 Функциональные методы

Функциональные методы

Список выходных параметров методов может меняться количественно в большую сторону.
Приведенные в данном руководстве параметры могут быть изменены только после предварительного уведомления!

Списки

❖**GetCitiesList** - Список городов
Возвращает список городов, имеющихся в системе

❖**GetLocationList** - Список театров
Возвращает список театров, имеющихся в системе

❖**GetHallList** - Список сцен
Возвращает список сцен для указанного театра

❖**GetCategoriesList** - Получить список категорий
Возращает список категорий мероприятий.

❖**GetShowList** - Список мероприятий
Возвращает список мероприятий (спектаклей)

❖**GetShowCategoriesList** - Получить список категорий для мероприятия
Возращает список категорий для мероприятия.

❖**GetGenresList** - Список жанров мероприятий
Возвращает список жанров мероприятий (спектаклей)

❖**GetShowGenresList** - Получить список жанров для мероприятия
Возращает список жанров для мероприятия.

❖**GetOrganizersList** - Получить список организаторов мероприятий
Возращает список организаторов мероприятий.

❖**GetEventList** - Список событий, имеющихся в продаже
Возвращает список событий, имеющихся в продаже на текущий момент.

❖**GetSectorsList** - Список секторов для событий
Возвращает список секторов.

❖**GetPlaceForSector** - Список мест сектора для события.
Возвращает список мест указанного сектора для указанного события.

❖**GetEvailPlaceList** - Список свободных мест на событие.
Возращает список сводобных для заказа/продажи мест на событие.

❖**GetSchemaHallList** - Получить полный список объектов на событие
Возращает Места, Метки, Линии в зале на определенное событие.

❖**GetSchemaHallListEventsSet** - Получить полный список объектов комплекта
Возращает Места, Метки, Линии в зале на определенного комплекта.

❖**GetShowsSetsList**- Получить список комлектов
Возращает список комплектов мероприятий

Предварительное бронирование
❖**PreSetReservation** - Предварительное резервирование.
Устанавливает метку о предварительном бронировании*.

❖**PreSetReservationEventsSet** - Предварительное резервирование мест комплекта.
Устанавливает метку о предварительном бронировании* мест комплекта.

❖**FreePreSetReservation** - Сниятие предварительного бронирования
Снимает метку о предварительном бронировании

❖**FreePreSetReservationEventsSet** - Сниятие предварительного бронирования с мест комплекта
Снимает метку о предварительном бронировании с мест комплекта

Бронирование и доставка
❖**GetMetroList** - Получить список станций метро
Возращает список станций метро и наименования линий.

❖**GetOfficesList** - Получить список офисов (касс)
Возращает список офисов (касс).

❖**GetDeliveriesList** - Получить список услуг доставки
Возвращает список услуг доставки и их стоимости.

❖**SetReservation** - Бронирование мест на событие
Устанавливает отметку о бронировании.

❖**SetReservationEventsSet** - Бронирование мест комплекта на события
Устанавливает отметку о бронировании мест комплекта.

❖**FreeReservation** - Снятие бронирования мест
Снимает метку о бронировании.

❖**FreeReservationEventsSet** - Снятие бронирования с мест комплекта
Снимает метку о бронировании с мест комплекта.

❖**GetReservationStatus** - Получить статус заказа
Возвращает текущий статус заказа.

❖**GetReservationStatus2** - Получить статус заказа
Возвращает текущий статус заказа с учетов снятых заказов.

❖**CheckPromoCode** - Проверить промокод(ы)
Возвращает состояние указанных промокодов, их валидность, процент скидки и т.д.

Продажа и возврат

❖**CheckSoldTickets** - Проверка мест перед продажей
Проверяет места перед продажей на возможность таковой.

❖**SetSold** - Продажа места
Устанавливает метку о продаже места.

❖**SetSoldEventsSet** - Продажа мест комплекта
Устанавливает метку о продаже мест комплекта.

❖**SetReturn** - Возврат места
Возращает проданные места.

Журналирование

❖**GetLog** - Получить фрагмент журнала операций.
Возвращает список операций за указанный временной интервал.

❖**GetCurrentState** - Сверка состояния проданных и забронированных мест
Возвращает состояние в системе Профтикет® проданных/

забронированных через шлюз мест за указанный временной интервал.

2.1 Списки

❖**GetCitiesList** - Список городов
Возвращает список городов, имеющихся в системе

❖**GetLocationList** - Список мест проведения (театров)
Возвращает список театров, имеющихся в системе

❖**GetHallList** - Список залов места проведения (сцен)
Возвращает список сцен для указанного театра или общий список сцен по театрам

❖**GetCategoriesList** - Получить список категорий
Возращает список категорий мероприятий.

❖**GetShowList** - Список мероприятий
Возвращает список мероприятий (спектаклей)

❖**GetShowCategoriesList** - Получить список категорий для мероприятия
Возращает список категорий для мероприятия.

❖**GetGenresList** - Список жанров мероприятий
Возвращает список жанров мероприятий (спектаклей)

❖**GetShowGenresList** - Получить список жанров для мероприятия
Возращает список жанров для мероприятия.

❖**GetOrganizersList** - Получить список организаторов мероприятий
Возращает список организаторов мероприятий.

❖**GetEventList** - Список событий, имеющихся в продаже
Возвращает список событий, имеющихся в продаже на текущий момент.

❖**GetSectorsList** - Список секторов для событий
Возвращает список секторов.

❖**GetPlaceForSector** - Список мест сектора для события.
Возвращает список мест указанного сектора для указанного события.

❖**GetEvailPlaceList** - Список свободных мест на событие.
Возвращает список сводобных для заказа/продажи мест на событие.

❖**GetSchemaHallList** - Получить полный список объектов на событие
Возращает Места, Метки, Линии в зале на определенное событие.

❖**GetSchemaHallListEventsSet** - Получить полный список объектов комплекта
Возращает Места, Метки, Линии в зале определенного комплекта.

❖**GetShowsSetsList**- Получить список комлектов
Возращает список комплектов мероприятий

❖**GetServicesList**- Список услуг по событию
Возращает список услуг, привязанных к событию

2.1.1 Список городов

Список городов

Function **GetCitiesList**(strInp: string): string;

Метод возвращает список имеющихся городов

Описание полей в выходных данных

- **CityId** ID города
- **CityName** Наименование города
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow/>

 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
       <row CityId="Id города"   CityName="Наименование города"
result_code="код ошибки" result_message="Описание ошибки" />
      ……
 </AnswerBody>
</GateAnswer>
```

2.1.2 Список мест проведения (театров)

Список мест проведения (театров)

Function **GetLocationList**(strInp: string): string;

Метод возвращает список имеющихся мест проведения (театров)

Описание полей в выходных данных

- **cod_t** - ID театра (int)
- **name_t** Наименование театра (string)
- **Address_t** Адрес теата
- **Tel_t** Телефон театра
- **Fax_t** Факс театра
- **Mail_t** Email театра
- **Web_t** Адрес сайта театра
- **cod_city** Код города, в котором расположен театр
- **name_city** Наименование города, в котором расположен театр
- **CityId** Код города, в котором расположен театр
- **CityName** Наименование города, в котором расположен театр
- **result_code** - Код ошибки.

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow/>

 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
```

```
        <row cod_t="ID театра"  name_t="Наименование театра"
Address_t="адрес театра" Tel_t="телефон театра" Email_t="Email
театра" Fax_t="Факс театра"  Web_="http://адрес театра сайта"
cod_city="Код города" name_city="Наименование города"
result_code="код ошибки"/>
        ……
 </AnswerBody>
</GateAnswer>
```

2.1.3 Список залов места проведения (сцен театра)

Список залов места проведения (сцен театра)

Function **GetHallList**(strInp: string): string;

Метод возвращает список сцен для указанного театра

Описание входных параметров.

- **cod_t** - код театра

Описание полей в выходных данных

- **cod_t** ID театра
- **cod_th** - ID сцены театра (int)
- **name_h** Наименование сцены театра (string)
- **name_h2** Краткое наименование сцены театра (string)
- **Address_h** адрес сцены театра (string)
- **Tel_h** Телефон сцены театра (string)
- **Email_h** Email сцены театра (string)
- **Fax_h** Факс сцены театра (string)
- **result_code** - Код ошибки.

Обратите внимание!
Если код театра не указан, то метод возвращает список всех сцен с идентификаторами театров, к которым они относятся!

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow cod_t="ID театра"/>
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
```

```
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row cod_th ="ID сцены" name_h ="Наименование сцены театра"
result_code="Код ошибки"/>
       ……
 </AnswerBody>
</GateAnswer>
```

2.1.4 Список категорий

Список категорий мероприятий

Function **GetCategoriesList** (strInp: string): string;

Метод возвращает список категорий мероприятий.

Описание полей в выходных данных

- **code_categ -** ID категории**,**
- **name_categ -** Наименование категории
- **result_code -** Код ошибки

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow />
      ……
 </ReqBody>
</GateReq>
```

Пример выходного XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>5</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row code_categ="1" name_categ="Концерт" result_code="0"/>
   <row code_categ="2" name_categ="Балет"  result_code="0"/>
   <row code_categ="3" name_categ="Цирк"  result_code="0"/>
      ……
 </AnswerBody>
</GateAnswer>
```

2.1.5 Список категорий мероприятия

Список категорий мероприятия

Function **GetShowCategoriesList** (strInp: string): string;

Метод возвращает список категорий указанного мероприятия. Если не указан параметр code_show, то метод возвращает список категорий по всем мероприям

Описание входных параметров.

- **code_show** - ID *мероприятия (необязательный параметр)*

Описание полей в выходных данных

- **code_show** - ID мероприятия,
- **code_categ** - ID категории,
- **name_categ** - Наименование категории
- **result_code** - Код ошибки

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow cod_show="24"/>
    ……
 </ReqBody>
</GateReq>
```

Пример выходного XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>5</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row code_show="24" name_show="Синяя птица" code_categ="9"
name_categ="Детский спектакль"/>
   <row code_show="24" name_show="Синяя птица" code_categ="15"
name_categ="Мьюзикл"/>
      ……
```

```
 </AnswerBody>
</GateAnswer>
```

2.1.6 Список жанров

Список жанров мероприятий

Function **GetGenresList** (strInp: string): string;

Метод возвращает список жанров мероприятий.

Описание полей в выходных данных

- **code_genre -** ID жанра**,**
- **name_genre -** Наименование жанра
- **result_code -** Код ошибки

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow />
      ……
 </ReqBody>
</GateReq>
```

Пример выходного XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>5</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row code_genre="1" name_genre="Комедия" result_code="0"/>
   <row code_genre="2" name_genre="Водевиль"  result_code="0"/>
   <row code_genre="3" name_genre="Сказка в 2-х частях"
result_code="0"/>
      ……
 </AnswerBody>
</GateAnswer>
```

2.1.7 Список жанров мероприятия

Список жанров мероприятия

Function **GetShowGenresList** (strInp: string): string;

Метод возвращает список жанров указанного мероприятия. Если не указан ID мероприятия, возвращает все жанры по мероприятиям.

Описание входных параметров.

- **code_show** - ID *мероприятия (необязательный параметр)*

Описание полей в выходных данных

- **code_genre -** ID жанра**,**
- **name_genre -** Наименование жанра
- **result_code -** Код ошибки

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow cod_show="24"/>
    ……
 </ReqBody>
</GateReq>
```

Пример выходного XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>5</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row code_show="28" code_categ="9" name_categ="Комедия"/>
   <row code_show="28" code_categ="15" name_categ="Милодрама"/>
      ……
 </AnswerBody>
</GateAnswer>
```

2.1.8 Список мероприятий (спектаклей)

Список мероприятий (спектаклей)

Function **GetShowList**(strInp: string): string;

Метод возвращает список имеющихся мероприятий

Данный метод возвращает список мероприятий. Для сторонних сайтов метод возвращает только мероприятия, на которые выдана квота и свободные.
Для "своего" сайта, если указан параметр FreeOnly = 0, то выдается список всех мероприятий, если FreeOnly = 1, то всех мероприятий, на которые выдана квота и есть свободные места.
Если указан параметр cod_show, то метод вернет информацию только по мероприятию с указанным ID.

Описание входных параметров.

- **cod_show** ID мероприятия *(необязательный параметр)*
- **FreeOnly** - Признак показывать только свободные *(необязательный параметр). По-умолчанию 1.*
- **OffSet** - При выводе списка записей смещение от начала списка (*необязательный параметр*)
- **Limit** - При выводе списка записей кол-во записей, которые необходимо вернуть (*необязательный параметр*)

В данном методе введена необязательная "пагинация"*. Для ее использования необходимо указать параметры:
Offset - сдвиг от начала списка
Limit - кол-во записей для вывода.
Например: *если указано Offset="0" Limit="5" будет возвращено 5 записей от начала списка.*
если указано Offset="5" Limit="10" будет возвращено 10 записей начиная от шестой (1+5) от начала списка.

Также введен параметр **AllRecords** в XML ответа для получения общего кол-ва записей, если используется "пагинация".
Если "пагинация" не используется, **AllRecords** равен **RecordCount**

Описание полей в выходных данных

- **cod_show** - ID мероприятия (int)
- **name_show** - Наименование мероприятия (string)
- **name_show2** - Наименование мероприятия (дополнительное),
- **author** - автор
- **producer** - режиссер

- **actors** - актеры,
- **annotation** - анотация,
- **duration** - продолжительность *(может отличаться от продолжительности конкретного события)*
- **is_Primera** - премьера *(может отличаться от признака для конкретного события)*
- **WithIntermisson** - признак "с антрактом" *(может отличаться от признака для конкретного события)*
- **note1** - строка примечания 1,
- **note2** - строка примечания 2,
- **note3** - строка примечания 3,
- **note4** - строка примечания 4,
- **age** - Возрастное ограничение (INT), -1 - не определен. *(если передается, например, 16, это означает 16+)*
- **DateBegin** - Первая дата проведения мероприятия
- **TimeBegin** - Время первой даты проведения мероприятия
- **DateEnd** - Последняя дата проведения мероприятия
- **TimeEnd** - Время последней даты проведения мероприятия

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow/>
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
   <AllRecords>0</AllRecords><!-- Общее количество записей для
возврата  -->
 </AnswerResult>
 <AnswerBody>
```

```
   <row cod_show ="ID мероприятия" name_ show ="Имя
мероприятия" />
        ……
 </AnswerBody>
</GateAnswer>
```

2.1.9 Список событий, имеющихся в продаже

Список событий, имеющихся в продаже

Function **GetEventList** (strInp: string): string;

Метод возвращает список актуальных на момент запроса событий (единиц репертуара) .

Данный метод возвращает список актуальных на момент запроса мероприятий.
Для сторонних сайтов метод возвращает только события, на которые выдана квота и есть свободные места.
*Для "своего" сайта, если указан параметр **FreeOnly = 0**, то выдается список всех актуальных событий, если **FreeOnly = 1**, то всех актуальных событий, на которые выдана квота и есть свободные места.*
Если указан параметр NomBilKn, то метод вернет информацию только по событию с указанным ID.

Описание входных параметров.

- **cod_t** - ID места проведения (театра) *Необязательный параметр*
- **cod_th** - ID зала места проведения (сцены) *Необязательный параметр*
- **NomBilKn** - ID события. *Необязательный параметр.*
- **FreeOnly** - Признак показывать только свободные *(необязательный параметр)*. ***По-умолчанию 1***.
- **Date** - Дата, на которую вернуть список мероприятий. Время не учитывается *(необязательный параметр)*.
- **cod_show** - ID мероприятия (спектакля). (*необязательный параметр*)
- **DateFrom** - Дата начала вывода списка событий. Время не учитывается. *(необязательный параметр)*
- **DateTo** - Дата окончания вывода списка событий. Время не учитывается. *(необязательный параметр)*
- **SeasonTicketId** - Идентификатор абонемента. *(необязательный параметр)*.
- **OffSet** - При выводе списка записей смещение от начала списка (*необязательный параметр*)
- **Limit** - При выводе списка записей кол-во записей, которые необходимо вернуть (*необязательный параметр*)

Метод принимает только одну запись входящих параметров.

(Нельзя передать, например: <InputRow NomBilKn="1" /><InputRow NomBilKn="2" />)

В данном методе введена необязательная "пагинация"*. Для ее использования необходимо указать параметры:
Offset - сдвиг от начала списка
Limit - кол-во записей для вывода.
Например: *если указано Offset="0" Limit="5" будет возвращено 5 записей от начала списка.*
если указано Offset="5" Limit="10" будет возвращено 10 записей начиная от шестой (1+5) от начала списка.

Также введен параметр **AllRecords** в XML ответа для получения общего кол-ва записей, если используется "пагинация".
Если "пагинация" не используется, **AllRecords** равен **RecordCount**

Если указан параметр SeasonTicketId то остальные параметры фильтрации не учитываются!
Пагинация учитывается, но рекомендуется в этом случае ее не использовать.

Описание полей в выходных данных

- **NomBilKn** - ID события
- **EventDate** - Дата события *(или дата окончания события, если оно с открытой датой)*
- **EventTime** - Время события *(или время окончания события, если оно с открытой датой)*
- **BeginDate** - Дата начала события *(если событие с открытой датой; если нет, то равно EventDate)*
- **BeginTime** - Время начала события *(если событие с открытой датой; если нет, то равно EventTime)*
- **withOpenDate** - Признак событие с открытой датой
- **StopOrderDate** - Дата, после которой бронирование на данное событие запрещено.
- **StopOrderTime** - Время, после которого бронирование на данное событие запрещено.
- **StopSaleDate** - Дата, после которой продажа на данное событие

запрещено.
- **StopSaleTime** - Время, после которого продажа на данное событие запрещено.
- **cod_show** - ID мероприятия
- **cod_t -** ID места проведения (театра)
- **cod_h -** ID зала места проведения (сцены)
- **EventDuration** Продолжительность события *(может отличаться от продолжительности мероприятия)*
- **is_Primera** - Флаг, премьера или нет *(может отличаться от флага, установленного для мероприятия)*
- **WithIntermission** - Флаг "с антрактом" *(может отличаться от флага, установленного для мероприятия)*
- **EventNote** - Примечания к событию,
- **name_show** - Наименование мероприятия,
- **author** - Автор мероприятия,
- **Note1** - Строка примечаний к мероприятию 1,
- **Note2** - Строка примечаний к мероприятию 2,
- **Note3** - Строка примечаний к мероприятию 3,
- **Note4** - Строка примечаний к мероприятию 4,
- **Age** - Возрастное ограничение (INT), -1 - не определен. *(если передается, например, 16, это означает 16+)*
- **Producer** - Продюссер,
- **Actors** - Актеры,
- **Annotation** - Анотация,
- **OrganizerID** - ID организатора мероприятия,
- **OrganizerName** - Наименование организатора мероприятия,
- **OrganizerTin** - ИНН организатора мероприятия,
- **OrganizerPhone** - Телефон организатора мероприятия,
- **OrganizerEmail** - Email организатора мероприятия,
- **OrganizerAddress** - Адрес организатора мероприятия,

- **MinPrice** - Цена номинал минимальная
- **MinPriceSell** - Цена продажи минимальная
- **MaxPrice** - Цена номинал максимальная
- **MaxPriceSell** - Цена продажи максимальная

- **FreePlacesQty** - Кол-во доступных для шлюза мест.
- **FreePlacesTicketOffice** - Кол-во свободных мест в кассе театра. (Для стороннего сайта всегда равно -1)
- **ETicketPermitted** - Разрешен проход по электронному билету. (1 - разрешено, 0 - нет).

- **SeasonTicket** - Признак абонемента *(0 - обычное событие, 1 - событие-абонемент, 2 - событие относится к абонементу)*
- **SeasonTicketID** - Идентификатор абонемента *(0 - если событие - не абонемент)*
- **SeasonTicketEvents** - Кол-во событий, относящихся к абонементу *(0 - если событие - не абонемент)*.

- **EventsSetID** - Идентификатор комплекта (0, если не комплект)
- **EventsSetEvents** - Кол-во событий, относящихся к комплекту *(0 - если событие - не комплект)*.
- **EventsSetName** - Наименование комплекта.

- **MultiSelectAllowed** - Разрешен множественный выбор мест. *(Если параметр равен 1, места события могут быть объеденены с местами других событий в одном заказе)*
- **isService** - Признак, является ли событие услугой
- **LinkedServicesExists** - Существуют ли привязанные к мероприятию услуги (*услуги могут быть связаны только с событиями, у которых нет признака "услуга"*)
- **PromoExists** - Существуют ли промокоды на событие

Обратите внимание!
Поле FreePlacesTicketOffice возвращает реальное значение только для "своего" сайта. Для стороннего сайта значение этого поля равно -1.

Обратите внимание!
Дата и время, указанные в StopOrderDate и StopOrderTime это дата и время, после которых не разрешена бронь, но может разрешена продажа (если поля StopSaleDate и StopSaleTime имеют соответствующие значения).

Заказ на событие, где разрешена только продажа, заказ будет снят автоматически сразу после разблокировки. Реализация алгоритма запрета брони, но возможности продажи лежит на стороне подключающегося сайта.

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow [cod_t="Код театра"] [cod_th="Код сцены"]/>
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
  <row NomBilKn ="ID события"  EventDate ="Дата" EventTime ="Время"
cod_show ="ID мероприятия" cod_t="ID театра" cod_h="ID сцены"/>
      ……
 </AnswerBody>
</GateAnswer>
```

2.1.10 Список услуг доставки

Список объектов на событие

Function **GetOgranizersList** (strInp: string): string;

Метод возвращает список организаторов мероприятия

Описание полей в выходных данных

- **OrganizerID -** ID организатора**,**
- **OrganizerName -** Наименование организатора
- **OrganizerTin -** ИНН организатора
- **OrganizerPhone -** Телефон организатора
- **OrganizerEmail -** Email организатора
- **OrganizerAddress -** Адрес организатора

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow />
    ……
 </ReqBody>
</GateReq>
```

Пример выходного XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>5</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row ID="1" Name="Доставка в пределах МКАД" Price="100"/>
   <row ID="2" Name="Доставка по Московской области" Price="300"/>
      ……
 </AnswerBody>
</GateAnswer>
```

2.1.11 Список секторов

Список секторов для события

Function **GetSectorList** (strInp: string): string;

Метод возвращает список секторов либо всех, либо содержащихся в зале для указанного события.

Описание входных параметров.

- **NomBilKn** - ID события (*необязательный параметр*)
- **FreeOnly** - признак все места/только выдеоенные в квоту. (*необязательный параметр*)

Описание полей в выходных данных

- **cod_sec** - ID сектора
- **name_sec** - Наименование сектора
- **FreeOfferSeat** - Признак свободной рассадки (1 - да, 0 - нет)
- **MinPrice** - Минимальная цена номинала доступных для продажи мест в секторе
- **MaxPrice** - Максимальная цена номинала доступных для продажи мест в секторе
- **MinPriceSell** - Минимальная цена продажи доступных для продажи мест в секторе
- **MaxPriceSell** - Максимальная цена продажи доступных для продажи мест в секторе
- **PlacesCount** - Количество доступных для продажи мест в секторе

Если указан ID события, то метод возвращает все сектора, в которых есть доступные для бронирования / продажи места.
Если ID события не указан, метод возвращает полный список секторов.

Параметр FreeOnly актуален только для случая, когда передается ID события. Если FreeOnly равен 1 (по-умолчанию), то передаются только сектора, выделенные в квоту.
Если FreeOnly равен 0, то передаются все сектора, в независимости от выделенной квоты.

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
```

```
 </ReqLogin>
 <ReqBody>
      <InputRow NomBilKn="ID события"/>
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
       <row cod_sec ="Код (целое)" name_sec ="Имя сектора
(фрагмента)"  MinPrice="Мин. цена" MaxPrice="Макс. цена"
PlacesCount="Кол-во мест"/>
      ……
 </AnswerBody>
</GateAnswer>
```

2.1.12 Список мест сектора для события

Список мест сектора для события

Function **GetPlaceForSector**(strInp: string): string;

Метод возвращает список рядов и мест, содержащихся в указанном фрагменте указанного события (или на все событие, если не указан идентификатор сектора). Если в ответе шлюза для места указана цена, то оно доступно для продажи. Если цена равно нулю, то место недоступно для продажи.

Описание входных параметров.

- **NomBilKn** - ID события
- **cod_sec** - ID сектора *(необязательный параметр)*

Описание полей в выходных данных

- **cod_hs** ID места (int)
- **NomBilKn** - ID события
- **cod_sec** - ID сектора
- **name_sec** - Наименование сектора
- **row** ряд (string[4])
- **seat** место (string[4])
- **FreeOfferSeat** - признак свободной рассадки (1 - да, 0 - нет)
- **entranceId** - ID подъезда
- **entranceName** - Наименование подъезда
- **Price** - Цена номинал.
- **PriceSell** - Цена продажи.
- **EventsSetID** - ID комлекта *(0 если не комлект)*
- **isEventSet** - признак, что место относится к комлекту (1-да, 0 - нет)
- **result_code** - Код ошибки.

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow NomBilKn="ID события" cod_sec="ID сектора"/>
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row cod_hs ="ID места" row ="ряд" seat ="место"/>
       ……
 </AnswerBody>
</GateAnswer>
```

2.1.13 Список свободных мест на событие

Список свободных мест на событие

Function **GetEvailPlaceList**(strInp: string): string;

Метод возвращает список свободных мест на событие. Каждый элемент списка содержит информацию об одном месте – идентификатор места, ряд, место и цена.

Описание входных параметров.

- **NomBilKn** - ID события

Описание полей в выходных данных

- **cod_hs** - ID места (int)
- **cod_sec** - ID сектора (int).
- **row** - ряд (string[4])
- **seat** - место (string[4])
- **FreeOfferSeat** - признак свободной рассадки (1 - да, 0 - нет)
- **Price** - Цена билета (номинал)*
- **PriceSell** - Цена билета (цена продажи)*
- **EventsSetID** - ID комлекта *(0 если не комлект)*
- **isEventSet** - признак, что место относится к комлекту (1-да, 0 - нет)
- **entranceId** - ID подъезда
- **entranceName** - Наименование подъезда

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow NomBilKn="ID события"/>
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
```

```
</AnswerResult>
<AnswerBody>
  <row cod_hs ="ID места" cod_sec ="ID сектора"  row ="ряд" seat
="место" Price="Цена билета" PriceSell="Цена продажи"/>
      ……
</AnswerBody>
</GateAnswer>
```

2.1.14 Список свободных мест комплекта

Список свободных мест комплекта

Function **GetEvailPlaceListEventsSet**(strInp: string): string;

Метод возвращает список свободных мест комплекта. Каждый элемент списка содержит информацию об одном месте – идентификатор места, ряд, место и цена.

Описание входных параметров.

- **EventsSetID** - ID комплекта (int)

Описание полей в выходных данных

- **EventsSetID** - ID комплекта (int)
- **cod_sec** - ID сектора (int).
- **row** - ряд (string[4])
- **seat** - место (string[4])
- **FreeOfferSeat** - признак свободной рассадки (1 - да, 0 - нет)
- **Price** - Цена билета (номинал)*
- **PriceSell** - Цена билета (цена продажи)*

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow NomBilKn="ID события"/>
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row EventsSetID ="ID комплекта" cod_sec ="ID сектора"  row
="ряд" seat ="место" Price="Цена билета" PriceSell="Цена продажи"/
>
```

```
……
</AnswerBody>
</GateAnswer>
```

2.1.15 Список объектов на событие

Список объектов на событие

Function **GetSchemaHallList** (strInp: string): string;

Метод возвращает список записей о местах зала и метках на определенное событие.
"Метка ряда" - это метка, которая отображает номер ряда и нераздельно связана с первым или последним местов в ряду.
"Метка" - это текстовая метка используемая для написания наименования секторов, обозначения сцены и т.д.

Описание входных параметров.
- **NomBilKn** - ID события
- **cod_sec** - ID сектора *(необязательный параметр)*
- **ShowLabels** - Показывать метки и все отличные от мест объекты *(необязательный параметр)*

Описание полей в выходных данных
- **cod_hs -** ID объекта**,**
- **NomBilKn -** ID события**,**
- **ObjectType** - тип объекта (Place - место, Label - метка, FreeOfferSeatObject**, StageSign***)
- **PlaceSize -** Размер кресла *(Для метки не используется)***,**
- **Width -** Ширина кресла (метки)**,**
- **Height -** Высота кресла (метки)**,**
- **CX -** Координата левого верхнего угла по оси X**,**
- **CY -** Координата левого верхнего угла по оси Y**,**
- **CX2 -** Координата правого нижнего угла по оси X**,**
- **CY2 -** Координата правого нижнего угла по оси Y**,**
- **Angle -** Угол поворота кресла (метки)**,**
- **Row -** Ряд *(Для метки не используется)*,
- **Seat -** Место *(Для метки не используется)*,
- **cod_sec -** ID сектора *(Для метки не используется)*,
- **Name_sec -** Наименование сектора (Для метки - текст метки),
- **entranceId** - ID подъезда,
- **entranceName** - Наименование подъезда,
- **FreeOfferSeat -** признак свободной рассадки (1 - да, 0 - нет) *(Для метки не используется)*,
- **BackColor** - Цвет ценовой зоны или объекта *(Для метки не используется)*
- **FontColor** - Цвет шрифта *(Для места не используется)*

- **FontSize** - размер шрифта *(Для места не используется)*
- **Label -** Наличие и расположение метки ряда*
- **MinX -** Минимальная координата зала по оси X,
- **MinY -** Минимальная координата зала по оси Y,
- **MaxX -** Максимальная координата зала по оси X,
- **MaxY -** Максимальная координата зала по оси Y

***Label**

Для кресла: метка ряда, которая находится рядом с креслом. 0 - нет метки ряда, 1 - метка ряда слева, 2 - метка ряда справа, 3 - метка ряда сверху, 4 - метка ряда снизу.	**Для метки:** Первая цифра - выравнивание по горизонтали 1 - По левому краю, 2 - По центру, 3 - По правому краю Вторая цифра - выравнивание по вертикали 1 - По верхнему краю, 2 - По центру, 3 - По нижнему краю

Например, 21 - выравнивание по центру по горизонтали и по верхнему краю по вертикали.

** FreeOfferSeatObject - данный объект нужен для того, чтобы не отображать на схеме места со свободной рассадкой соответствующего сектора,
а заменить их прямоугольным блоком. При нажатие на него должна вызываться продажа этих мест. Разработчик сайта должен обеспечить соответствующий функционал.

*** StageSign - указатель сцены. Этот объет указывает на расположение сцены в макете.

Для отображения меток и объектов используются следующие атрибуты:

Место:
ObjectID="0" ObjectName="Place" ObjectType="Place"

Метка (надпись):
ObjectID="1" ObjectName="Label" ObjectType="Label"

Указатель сцены:
ObjectID="1" ObjectName="StageSign" ObjectType="StageSign"

Указатель сектора со свободной рассадкой:
ObjectID="2" ObjectName="FreeOfferSeatObject"
ObjectType="FreeOfferSeatObject"

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow NomBilKn="ID события" [cod_sec="ID сектора"] />
      ……
 </ReqBody>
</GateReq>
```

Пример выходного XML.

```
<?xml version="1.0" encoding="windows-1251"?>
<GateAnswer>
 <AnswerResult>
  <ResultCode>0</ResultCode>
  <RecordCount>907</RecordCount>
 </AnswerResult>
 <AnswerBody>
  <Row cod_hs="326607" NomBilKn="478" ObjectID="0" ObjectName="Place" PlaceSize="22" Width="22" Height="22"
PointIndex="0" GroupPointIndex="0" CX="837" CY="155" Angle="0" Row="2" Seat="3" cod_sec="1"
Name_sec="ПАРТЕР" Label="" BackColor="" FontColor="" FontSize="0" MinX="44" MinY="20" MaxX="1094"
MaxY="792"/>
  <Row cod_hs="326608" NomBilKn="478" ObjectID="0" ObjectName="Place" PlaceSize="22" Width="22" Height="22"
PointIndex="0" GroupPointIndex="0" CX="815" CY="155" Angle="0" Row="2" Seat="4" cod_sec="1"
Name_sec="ПАРТЕР" Label="" BackColor="" FontColor="" FontSize="0" MinX="44" MinY="20" MaxX="1094"
MaxY="792"/>
  <Row cod_hs="344773" NomBilKn="478" ObjectID="1" ObjectName="Point" PlaceSize="0" Width="0" Height="0"
PointIndex="1" GroupPointIndex="1" CX="121" CY="452" Angle="0" Row="" Seat="" cod_sec="0" Name_sec=""
Label="" BackColor="000000" FontColor="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094" MaxY="792"/>
  <Row cod_hs="344769" NomBilKn="478" ObjectID="1" ObjectName="Point" PlaceSize="0" Width="0" Height="0"
PointIndex="2" GroupPointIndex="1" CX="121" CY="473" Angle="0" Row="" Seat="" cod_sec="0" Name_sec=""
Label="" BackColor="000000" FontColor="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094" MaxY="792"/>
  <Row cod_hs="344770" NomBilKn="478" ObjectID="1" ObjectName="Point" PlaceSize="0" Width="0" Height="0"
PointIndex="3" GroupPointIndex="1" CX="935" CY="473" Angle="0" Row="" Seat="" cod_sec="0" Name_sec=""
Label="" BackColor="000000" FontColor="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094" MaxY="792"/>
  <Row cod_hs="344771" NomBilKn="478" ObjectID="1" ObjectName="Point" PlaceSize="0" Width="0" Height="0"
PointIndex="4" GroupPointIndex="1" CX="935" CY="297" Angle="0" Row="" Seat="" cod_sec="0" Name_sec=""
Label="" BackColor="000000" FontColor="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094" MaxY="792"/>
  <Row cod_hs="344772" NomBilKn="478" ObjectID="1" ObjectName="Point" PlaceSize="0" Width="0" Height="0"
PointIndex="5" GroupPointIndex="1" CX="121" CY="297" Angle="0" Row="" Seat="" cod_sec="0" Name_sec=""
Label="" BackColor="000000" FontColor="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094" MaxY="792"/>
  <Row cod_hs="327415" NomBilKn="478" ObjectID="2" ObjectName="Label" PlaceSize="0" Width="0" Height="0"
PointIndex="0" GroupPointIndex="0" CX="972" CY="616" Angle="0" Row="" Seat="" cod_sec="0" Name_sec=""
Label="ЛОЖА №1" BackColor="FFFFFF" FontColor="000000" FontSize="11" MinX="44" MinY="20" MaxX="1094"
MaxY="792"/>
  <Row cod_hs="326782" NomBilKn="478" ObjectID="2" ObjectName="Label" PlaceSize="0" Width="0" Height="0"
```

PointIndex="0" GroupPointIndex="0" CX="902" CY="594" Angle="0" Row ="" Seat="" cod_sec="0" Name_sec=""
Label="2" BackColor="FFFFFF" FontColor="000000" FontSize="14" MinX="44" MinY="20" MaxX="1094" MaxY="792"/>
</AnswerBody>
</GateAnswer>

2.1.16 Список объектов комплекта

Список объектов комплекта

Function **GetSchemaHallListEventsSet** (strInp: string): string;

Метод возвращает список записей о местах зала и метках определенное событие.
"Метка ряда" - это метка, которая отображает номер ряда и нераздельно связана с первым или последним местов в ряду.
"Метка" - это текстовая метка используемая для написания наименования секторов, обозначения сцены и т.д.

Описание входных параметров.
- **EventsSetID** - ID комплекта
- **cod_sec** - ID сектора *(необязательный параметр)*
- **ShowLabels** - Показывать метки и все отличные от мест объекты *(необязательный параметр)*

Описание полей в выходных данных
- **EventsSetID** - ID комплекта
- **ObjectType** - тип объекта (Place - место, Label - метка, FreeOfferSeatObject**, StageSign***)
- **PlaceSize -** Размер кресла *(Для метки не используется)***,**
- **Width -** Ширина кресла (метки)**,**
- **Height -** Высота кресла (метки)**,**
- **CX -** Координата левого верхнего угла по оси X**,**
- **CY -** Координата левого верхнего угла по оси Y**,**
- **CX2 -** Координата правого нижнего угла по оси X**,**
- **CY2 -** Координата правого нижнего угла по оси Y**,**
- **Angle -** Угол поворота кресла (метки)**,**
- **Row -** Ряд *(Для метки не используется)*,
- **Seat -** Место *(Для метки не используется)*,
- **cod_sec -** ID сектора *(Для метки не используется)*,
- **Name_sec -** Наименование сектора (Для метки - текст метки),
- **FreeOfferSeat -** признак свободной рассадки (1 - да, 0 - нет) *(Для метки не используется)*,
- **BackColor** - Цвет ценовой зоны или объекта *(Для метки не используется)*
- **FontColor** - Цвет шрифта *(Для места не используется)*
- **FontSize** - размер шрифта *(Для места не используется)*
- **Label -** Наличие и расположение метки ряда*

- **MinX -** Минимальная координата зала по оси X,
- **MinY -** Минимальная координата зала по оси Y,
- **MaxX -** Максимальная координата зала по оси X,
- **MaxY -** Максимальная координата зала по оси Y

Список объектов схема зала выбираются из самого раннего события, относящегося к комплекту.

***Label**

Для кресла: метка ряда, которая находится рядом с креслом. 0 - нет метки ряда, 1 - метка ряда слева, 2 - метка ряда справа, 3 - метка ряда сверху, 4 - метка ряда снизу.	**Для метки:** Первая цифра - выравнивание по горизонтали 1 - По левому краю, 2 - По центру, 3 - По правому краю Вторая цифра - выравнивание по вертикали 1 - По верхнему краю, 2 - По центру, 3 - По нижнему краю

Например, 21 - выравнивание по центру по горизонтали и по верхнему краю по вертикали.

** FreeOfferSeatObject - данный объект нужен для того, чтобы не отображать на схеме места со свободной рассадкой соответствующего сектора,
а заменить их прямоугольным блоком. При нажатие на него должна вызываться продажа этих мест. Разработчик сайта должен обеспечить соответствующий функционал.

*** StageSign - указатель сцены. Этот объет указывает на расположение сцены в макете.

Для отображения меток и объектов используются следующие атрибуты:

Место:
ObjectID="0" ObjectName="Place" ObjectType="Place"

Метка (надпись):
ObjectID="1" ObjectName="Label" ObjectType="Label"

Указатель сцены:
ObjectID="1" ObjectName="StageSign" ObjectType="StageSign"

Указатель сектора со свободной рассадкой:
ObjectID="2" ObjectName="FreeOfferSeatObject"
ObjectType="FreeOfferSeatObject"

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow NomBilKn="ID события" [cod_sec="ID сектора"] />
    ……
 </ReqBody>
</GateReq>
```

Пример выходного XML.

```
<?xml version="1.0" encoding="windows-1251"?>
<GateAnswer>
  <AnswerResult>
    <ResultCode>0</ResultCode>
    <RecordCount>907</RecordCount>
  </AnswerResult>
  <AnswerBody>
    <Row cod_hs="326607" NomBilKn="478" ObjectID="0" ObjectName="Place" PlaceSize="22" Width="22" Height="22"
PointIndex="0" GroupPointIndex="0" CX="837" CY="155" Angle="0" Row ="2" Seat="3" cod_sec="1"
Name_sec="ПАРТЕР" Label="" BackColor="" FontColor="" FontSize="0" MinX="44" MinY="20" MaxX="1094"
MaxY="792"/>
    <Row cod_hs="326608" NomBilKn="478" ObjectID="0" ObjectName="Place" PlaceSize="22" Width="22" Height="22"
PointIndex="0" GroupPointIndex="0" CX="815" CY="155" Angle="0" Row ="2" Seat="4" cod_sec="1"
Name_sec="ПАРТЕР" Label="" BackColor="" FontColor="" FontSize="0" MinX="44" MinY="20" MaxX="1094"
MaxY="792"/>
    <Row cod_hs="344773" NomBilKn="478" ObjectID="1" ObjectName="Point" PlaceSize="0" Width="0" Height="0"
PointIndex="1" GroupPointIndex="1" CX="121" CY="452" Angle="0" Row ="" Seat="" cod_sec="0" Name_sec=""
Label="" BackColor="000000" FontColor="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094" MaxY="792"/>
    <Row cod_hs="344769" NomBilKn="478" ObjectID="1" ObjectName="Point" PlaceSize="0" Width="0" Height="0"
PointIndex="2" GroupPointIndex="1" CX="121" CY="473" Angle="0" Row ="" Seat="" cod_sec="0" Name_sec=""
Label="" BackColor="000000" FontColor="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094" MaxY="792"/>
    <Row cod_hs="344770" NomBilKn="478" ObjectID="1" ObjectName="Point" PlaceSize="0" Width="0" Height="0"
PointIndex="3" GroupPointIndex="1" CX="935" CY="473" Angle="0" Row ="" Seat="" cod_sec="0" Name_sec=""
Label="" BackColor="000000" FontColor="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094" MaxY="792"/>
    <Row cod_hs="344771" NomBilKn="478" ObjectID="1" ObjectName="Point" PlaceSize="0" Width="0" Height="0"
PointIndex="4" GroupPointIndex="1" CX="935" CY="297" Angle="0" Row ="" Seat="" cod_sec="0" Name_sec=""
Label="" BackColor="000000" FontColor="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094" MaxY="792"/>
    <Row cod_hs="344772" NomBilKn="478" ObjectID="1" ObjectName="Point" PlaceSize="0" Width="0" Height="0"
PointIndex="5" GroupPointIndex="1" CX="121" CY="297" Angle="0" Row ="" Seat="" cod_sec="0" Name_sec=""
Label="" BackColor="000000" FontColor="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094" MaxY="792"/>
    <Row cod_hs="327415" NomBilKn="478" ObjectID="2" ObjectName="Label" PlaceSize="0" Width="0" Height="0"
PointIndex="0" GroupPointIndex="0" CX="972" CY="616" Angle="0" Row ="" Seat="" cod_sec="0" Name_sec=""
Label="ЛОЖА №1" BackColor="FFFFFF" FontColor="000000" FontSize="11" MinX="44" MinY="20" MaxX="1094"
MaxY="792"/>
```

<Row cod_hs="326782" NomBilKn="478" ObjectID="2" ObjectName="Label" PlaceSize="0" Width="0" Height="0" PointIndex="0" GroupPointIndex="0" CX="902" CY="594" Angle="0" Row ="" Seat="" cod_sec="0" Name_sec="" Label="2" BackColor="FFFFFF" FontColor="000000" FontSize="14" MinX="44" MinY="20" MaxX="1094" MaxY="792"/>
</AnswerBody>
</GateAnswer>

2.1.17 Список комплектов

Список комплектов

Function **GetShowListEventsList**(strInp: string): string;

Метод возвращает список комплектов

Описание входных параметров.
- **EventsSetID** - ID комплекта *(необязательный параметр)*
- **ListOnly** - показывать только список комплектов *(необязательный параметр)*

Если параметр **ListOnly** = 1, то метод возвращает только список комплектов.
Если параметр **ListOnly** = 0 или опущен, то метод возвращает список событий, входящих в комплекты, а также кол-во свободных мест по всему событию и мест, относящихся к комплекту.

Если указан **EventsSetID**, то метод возвращает список мероприятий, входящий в комплект с указанным ID.
Метод игнорирует параметр EventsSetID при ListOnly = 1.

Описание полей в выходных данных
- **EventsSetID** ID комлекта
- **EventsSetName** Наименование комплекта
- **CountEvents** - Кол-во мероприятий, которое должно входить в комплект
- **NomBilKn** - ID события (отображается только если ListOnly = 0 или опущен)
- **cod_show** - ID мероприятия (отображается только если ListOnly = 0 или опущен)
- **cod_t** ID места проведения (театра)
- **cod_h** ID зала места проведения (сцены)
- **EventDate** - Дата проведения мероприятия (отображается только если ListOnly = 0 или опущен)
- **EventTime** - Время проведения мероприятия (отображается только если ListOnly = 0 или опущен)
- **EventFullName** - Наменование мероприятия (отображается только если ListOnly = 0 или опущен)
- **FreePlaces** - Кол-во свободных мест на данное событие (отображается только если ListOnly = 0 или опущен)
- **FreePlacesSet** - Кол-во свободных мест на данное событие, входящих в комплект (отображается только если ListOnly = 0 или

опущен)
- **ETicketPermitted -** Разрешен проход по электронному билету. (1 - разрешено, 0 - нет).
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow/>

 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
       <row EventsSetID="Id комплекта"  EventsSetName="Наименование комлпекта" CountEvents="Кол-во мероприятий" result_code="код ошибки" result_message="Описание ошибки" />
      ……
 </AnswerBody>
</GateAnswer>
```

2.1.18 Список связанных событий

Список услуг по событию

Function **GetLinkedEventsList**(strInp: string): string;

Метод возвращает связанных событий

Описание входных параметров.

- **ParentNomBilKn** - ID "родительского" события.

Описание полей в выходных данных

- **NomBilKn** - ID связанного события,
- **EventDate** - Дата связанного события (для событий с открытой датой - дата окончания события),
- **EventTime** - Время связанного события (для событий с открытой датой время окончания события),
- **BeginDate** - Дата начала связанного события (для событий с открытой датой - дата начала события),
- **BeginTime** - Время начала связанного события (для событий с открытой датой - время начала события),
- **withOpenDate** - Признак связанного событие с открытой датой,
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow ParentNomBilKn="5"/>

 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>1</RecordCount><!-- Количество записей в
```

```
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
        <Row NomBilKn="1404" EventDate="01.12.2016"
EventTime="19:00:00" BeginDate="01.12.2016" BeginTime="19:00:00"
withOpenDate="0" result_code="0" result_message="OK" />
 </AnswerBody>
</GateAnswer>
```

2.2 Бронирование

❖**PreSetReservation** - Предварительное резервирование.
Устанавливает метку о предварительном бронировании*.

❖**PreSetReservationEventsSet** - Предварительное резервирование мест комплекта.
Устанавливает метку о предварительном бронировании* мест комплекта.

❖**FreePreSetReservation** - Снятие предварительного бронирования
Снимает метку о предварительном бронировании

❖**FreePreSetReservationEventsSet** - Снятие предварительного бронирования мест комплекта
Снимает метку о предварительном бронировании с мест комплекта

❖**GetMetroList** - Получить список станций метро
Возращает список станций метро и наименования линий.

❖**GetOfficesList** - Получить список офисов (касс)
Возращает список офисов (касс).

❖**GetDeliveriesList** - Получить список услуг доставки
Возращает список услуг доставки и их стоимости.

❖**SetReservation** - Бронирование мест на событие
Устанавливает отметку о бронировании.

❖**SetReservationEventsSet** - Бронирование мест комплекта на событие
Устанавливает отметку о бронировании на места комплекта.

❖**FreeSetReservation** - Снятие бронирования мест
Снимает метку о бронировании.

❖**FreeReservationEventsSet** - Снятие бронирования мест комплекта
Снимает метку о бронировании с мест комплекта.

❖**GetReservationStatus** - Получить статус заказа
Возвращает текущий статус заказа.

❖**GetReservationStatus2** - Получить статус заказа
Возвращает текущий статус заказа с учетов снятых заказов.

❖**CheckPromoCode** - Проверить промокод(ы)

Возвращает состояние указанных промокодов, их валидность, процент скидки и т.д.

2.2.1 Предварительное резервирование мест

Предварительное резервирование мест

Function **PreSetReservation** (strInp: string): string;

Метод осуществляет резервирование указанных мест в БД таким образом, чтобы эти места на заданное время (глобальный параметр системы, от 3 до 20 мин.) стали недоступны для резервирования или бронирования с рабочего места системы «ProfTicket» и вызовам данного метода с другим значением идентификатора сессии. В том случае, если при попытке резервирования выяснится, что все или часть мест не являются свободными, резервирование таких мест не выполняется. Метод возвращает в качестве результата коллекцию кодов возврата, каждый элемент которой соответствует элементу исходной коллекции мест, и содержит код, указывающий на успешность или неуспешность выполнения резервирования.

Описание входных параметров.
- **NomBilKn** - ID события
- **cod_sec** ID сектора (int).
- **row** ряд (string[4])
- **seat** место (string[4])
- **session** - ID сессии
- **RemoteAddress -** IP адрес зрителя *(необязательное поле)*

Описание полей в выходных данных
- **cod_sec** ID сектора (int).
- **name_sec** Наименование сектора.
- **row** ряд (string[4]).
- **seat** место (string[4]).
- **price** Цена билета (номинал).*
- **priceSell** Цена билета (продажи)*
- **TTL** - Время в секундах, после которого места, находящиеся в предварительном резервировании могут быть сняты автоматически, если они не попали в заказ.
- **result_code** - Код ошибки.
- **result_message -** описание ошибки

Время, через которое предварительное резервирование будет снято автоматически, может немного отличаться от TTL в большую сторону. Это зависит от времени запуска сервиса автоматического снятия предварительного резервирования.

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow NomBilKn="ID события" cod_sec ="ID сектора"  row
="ряд" seat ="место" session="ID сессии"/>
      ……
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row NomBilKn="ID события" cod_sec ="ID сектора" row ="ряд"
seat ="место" price="Цена билета" result_code="Код ошибки"/>
      ……
 </AnswerBody>
</GateAnswer>
```

2.2.2 Предварительное резервирование мест комплекта

Предварительное резервирование мест комплекта

Function **PreSetReservationEventsSet** (strInp: string): string;

Метод осуществляет резервирование указанных мест в БД таким образом, чтобы эти места на заданное время (глобальный параметр системы, от 3 до 20 мин.) стали недоступны для резервирования или бронирования с рабочего места системы «ProfTicket» и вызовам данного метода с другим значением идентификатора сессии. В том случае, если при попытке резервирования выяснится, что все или часть мест не являются свободными, резервирование таких мест не выполняется. Метод возвращает в качестве результата коллекцию кодов возврата, каждый элемент которой соответствует элементу исходной коллекции мест, и содержит код, указывающий на успешность или неуспешность выполнения резервирования.

Описание входных параметров.
- **EventsSetID** - ID комплекта (int)
- **cod_sec** ID сектора (int).
- **row** ряд (string[4])
- **seat** место (string[4])
- **session** - ID сессии
- **RemoteAddress -** IP адрес зрителя *(необязательное поле)*

Описание полей в выходных данных
- **NomBilKn** ID события (int).
- **EventsSetID** - ID комплекта (int)
- **cod_sec** ID сектора (int).
- **name_sec** Наименование сектора.
- **row** ряд (string[4]).
- **seat** место (string[4]).
- **session** - ID сессии
- **price** Цена билета (номинал).*
- **priceSell** Цена билета (продажи)*
- **TTL** - Время в секундах, после которого места, находящиеся в предварительном резервировании могут быть сняты автоматически, если они не попали в заказ.
- **result_code** - Код ошибки.
- **result_message -** описание ошибки

Время, через которое предварительное резервирование будет снято автоматически, может немного отличаться от TTL в большую сторону. Это зависит от времени запуска сервиса автоматического снятия предварительного резервирования.

Обратите внимание!

В случае ошибки или невозможности осуществить операцию, возвращаемые данные содержат только ID комплекта, ID сектора, ряд и место.

В случае удачного осуществеления операции возвращается полный список мест по всем событиям комлекта.

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow EventsSetID="ID комплекта" cod_sec ="ID сектора"
row ="ряд" seat ="место" session="ID сессии"/>
      ……
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row NomBilKn="ID события" cod_sec ="ID сектора" row ="ряд"
```

seat ="*место*" **price**="*Цена билета*" **result_code**="*Код ошибки*"/>
……
</AnswerBody>
</GateAnswer>

2.2.3 Снятие предварительного резервирование мест

Снятие предварительного резервирования мест

Function **FreePreReservation** (strInp: string): string;

Метод осуществляет снятие резервирования указанных мест в БД, если они находятся в состоянии разервирования за сессией с указанным идентификатором.

Описание входных параметров.

- **NomBilKn** - ID события
- **cod_sec** ID сектора (int).
- **row** ряд (string[4])
- **seat** место (string[4])
- **session** - ID сессии
- **RemoteAddress** - IP адрес зрителя *(необязательное поле)*

Описание полей в выходных данных

- **cod_sec** ID сектора (int).
- **row** ряд (string[4]).
- **seat** место (string[4]).
- **price** Цена билета (номинал).*
- **priceSell** Цена билета (продажи)*
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow NomBilKn="ID события" cod_sec ="ID сектора" row
="ряд" seat ="место" session="ID сессии"/>
    ……
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
```

```
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
     <Row NomBilKn="ID события" cod_sec ="ID сектора" row ="ряд"
seat ="место" price="Цена билета" result_code="Код ошибки"/>
       ……
 </AnswerBody>
</GateAnswer>
```

2.2.4 Снятие предварительного резервирования мест комплекта

Снятие предварительного резервирования мест комплекта

Function **FreePreReservationEventsSet** (strInp: string): string;

Метод осуществляет снятие резервирования указанных мест в БД, если они находятся в состоянии разервирования за сессией с указанным идентификатором.

Описание входных параметров.
- **EventsSetID** - ID комплекта (int)
- **cod_sec** ID сектора (int).
- **row** ряд (string[4])
- **seat** место (string[4])
- **session** - ID сессии
- **RemoteAddress -** IP адрес зрителя *(необязательное поле)*

Описание полей в выходных данных
- **NomBilKn** - ID события (int)
- **cod_sec** ID сектора (int).
- **row** ряд (string[4]).
- **seat** место (string[4]).
- **price** Цена билета (номинал).*
- **priceSell** Цена билета (продажи)*
- **result_code** - Код ошибки.
- **result_message -** описание ошибки

Обратите внимание!

В случае ошибки или невозможности осуществить операцию, возвращаемые данные содержат только ID комплекта, ID сектора, ряд и место.

В случае удачного осуществеления операции возвращается полный список мест по всем событиям комплекта.

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow EventsSetID="ID комплекта" cod_sec ="ID сектора"
row ="ряд" seat ="место" session="ID сессии"/>
    ……
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
    <Row NomBilKn="ID события" cod_sec ="ID сектора" row ="ряд"
seat ="место" price="Цена билета" result_code="Код ошибки"/>
    ……
 </AnswerBody>
</GateAnswer>
```

2.2.5 Список станций метро

Список станций метро

Function **GetMetroList** (strInp: string): string;

Метод возвращает список записей о станция метро и линиях.

Описание полей в выходных данных

- **ID -** ID станции**,**
- **Name -** Наименование станции**,**
- **LineName -** Наименование линии (ветки) метро, к которой относится станция

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow />
      ……
 </ReqBody>
</GateReq>
```

Пример выходного XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>144</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row ID="25" Name="Парк Культуры" LineName = "Сокольническая"/>
   <row ID="89" Name="Парк Культуры" LineName = "Кольцевая"/>
   <row ID="36" Name="Кантемировская" LineName = "Замоскворецкая"/
>
      ……
 </AnswerBody>
</GateAnswer>
```

2.2.6 Список офисов

Список офисов

Function **GetOfficesList** (strInp: string): string;

Метод возвращает список офисов и касс.

Описание полей в выходных данных

- **ID -** ID офиса**,**
- **Name -** Наименование офиса

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow />
      ……
 </ReqBody>
</GateReq>
```

Пример выходного XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>5</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row ID="1" Name="Центральный офис"/>
   <row ID="2" Name="Доп. офис"/>
   <row ID="3" Name="Касса №1 (Отрадное)"/>
   <row ID="4" Name="Касса №2 (Марьино)"/>
      ……
 </AnswerBody>
</GateAnswer>
```

2.2.7 Список услуг доставки

Список услуг доставки

Function **GetDeliveriesList** (strInp: string): string;

Метод возвращает список записей об услугах доставки и их стоимости.

Описание полей в выходных данных

- **ID -** ID услуги**,**
- **Name -** Наименование услуги
- **Price -** цена

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow />
      ……
 </ReqBody>
</GateReq>
```

Пример выходного XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>5</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row ID="1" Name="Доставка в пределах МКАД" Price="100"/>
   <row ID="2" Name="Доставка по Московской области" Price="300"/>
      ……
 </AnswerBody>
</GateAnswer>
```

2.2.8 Бронирование мест на событие

Бронирование мест на событие

Function **SetReservation** (strInp: string): string;

Бронирование указанных мест осуществляется в БД при условии, что они находятся в состоянии резервирования за сессией с указанным идентификатором. Реквизиты заказа заносятся в БД таким образом, чтобы администратор программы мог получить информацию о реквизитах заказа для каждого места.

Описание входных параметров.

- **NomBilKn** - ID события
- **cod_sec** ID сектора (int).
- **row** ряд (string[4])
- **seat** место (string[4])
- **session** - ID сессии
- **NameSpektator -** ФИО зрителя
- **TelSpektator -** Телефон зрителя
- **EmailSpektator -** Email зрителя
- **SendAdvertising -** Согласие на рассылку (рекламу). *(необязательное поле)*. **0 - не согласен (по-умолчанию), 1 - согласен**. *Данный параметр записывается в справочник зрителей. Если зритель уже существует в справочнике, параметр перезаписывается.*
- **PinCode -** Пин-код заказа. (Int 4) *Необязательное поле. Если указан пин-код, то при выкупе заказа зритель должен назвать его кассиру. В противном случае заказ не будет распечатан.* ***Пин-код формируется сайтом и передается в шлюз.***
- **Delivery -** Наличие/отсутствие доставки (1 курьерская доставка, 0 - нет доставки) *(необязательное поле при отсутствии доставки)*
- **DeliveryDate -** Дата доставки *(необязательное поле при отсутствии доставки)*
- **Address -** Адрес доставки *(необязательное поле при отсутствии доставки)*
- **MetroID -** ID станции метро. *(необязательное поле)*
- **OfficeID -** ID офиса. *(необязательное поле) Указывается при самовывозе (т.е. отсутсвии доставки). Это предполагаемый офис выкупа билетов. Если забронирован один или несколько "живых" билетов, то указать ID офиса обязательно.*
- **Notes -** Комментарий *(необязательное поле)*
- **OrderID -** ID заказа в БД сайта или платежной системе *(необязательное поле)*
- **DeliveryService -** ID услуги доставки *(необязательное поле при*

отсутствии доставки)
- **RemoteAddress -** IP адрес зрителя *(необязательное поле)*
- **PromoCodeID -** ID промокода *(необязательное поле)*

Обратите внимание!
При наличии доставки поля: "Delivery" "DeliveryDate", "Address" и "DeliveryService" являются обязательными!

После оформления зрителем заказа, если продажа не последует, следует сменить номер сессии для зрителя!

Описание полей в выходных данных
- **NomBilKn** - ID события.
- **cod_sec** ID сектора.
- **name_sec** Наименоване сектора
- **row** ряд (string[4]).
- **seat** место (string[4]).
- **price** - Цена билета. (номинал)*
- **priceSell** - Цена билета (продажи)* с учетом введенного промокода
- **ID Spectator** ID зрителя.
- **NameSpectator** ФИО зрителя
- **TelSpectator** Телефон зрителя
- **EmailSpectator** Email зрителя
- **Delivery** Наличие/отсутствие доставки (1 есть доставка, 0 - нет доставки)
- **DeliveryDate -** Предполагаемая дата доставки
- **Address -** Адрес доставки
- **MetroID -** ID станции метро.
- **Metro -** Наименование станции метро.
- **OfficeID -** ID офиса.
- **OfficeName -** Наименование офиса.

- **Notes -** Комментарий
- **OrderID -** ID заказа в БД сайта или платежной системе
- **DeliveryService -** ID услуги доставки
- **session -** ID сессии
- **ReservID** - номер заказа (в случае ошибки равен 0)
- **Barcode** - штрих-код заказа (он может быть распечатан и предъявлен в кассу для идентификации заказа)
- **ReservDate** - дата "выкупа" заказа (в случае ошибки пустая)

- **ReservTime** - время "выкупа" заказа (в случае ошибки пустое)
- **BlockTTL** - Время блокировки заказа в секундах. (Время, в течение которого заказ блокирован для любых действий с ним, кроме как через шлюз)
- **PromoCodeID** - ID промокода
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

ReservDate и **ReservTime** - дата и время "выкупа" заказа. По истечение данного времени заказ считается "просроченным" и может быть снят сотрудниками театра/агентства либо автоматическим сервисом. На данный момент времени, заказы, оформленные на доставку автоматически не снимаются.
После оплаты (метод SetSold) заказ не может быть снят и его дата и время "выкупа" становятся равной максимальной дате и времени мероприятий входящих в его состав.

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow NomBilKn="ID события" cod_sec ="ID сектора" Row ="ряд"
Seat ="место" NameSpektator="ФИО зрителя" TelSpektator="Телефон
зрителя" EmailSpektator="email зрителя" session="ID сессии"/>
      ……
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в возвращаемом
"RecordSet-e" -->
 </AnswerResult>
 <AnswerBody>
   <row NomBilKn="ID события" cod_sec ="ID сектора" row ="ряд" seat
="место" price="Цена билета" reservid="№ заказа" reservdate="дата
```

заказа" **NameSpektator**="*ФИО зрителя*" **TelSpektator**="*Телефон зрителя*" **EmailSpektator**="*Email зрителя*" **result_code**="*код ошибки*"/>

2.2.9 Бронирование мест комплекта на событие

Бронирование мест комплекта на событие

Function **SetReservationEventsSet** (strInp: string): string;

Бронирование указанных мест осуществляется в БД при условии, что они находятся в состоянии резервирования за сессией с указанным идентификатором. Реквизиты заказа заносятся в БД таким образом, чтобы администратор программы мог получить информацию о реквизитах заказа для каждого места.

Описание входных параметров.

- **EventsSetID** - ID комплекта (int)
- **cod_sec** ID сектора (int).
- **row** ряд (string[4])
- **seat** место (string[4])
- **session** - ID сессии
- **NameSpektator -** ФИО зрителя
- **TelSpektator -** Телефон зрителя
- **EmailSpektator -** Email зрителя
- **SendAdvertising -** Согласие на рассылку (рекламу). *(необязательное поле).* **0 - не согласен (по-умолчанию), 1 - согласен**. *Данный параметр записывается в справочник зрителей. Если зритель уже существует в справочнике, параметр перезаписывается.*
- **PinCode -** Пин-код заказа. (Int 4) *Необязательное поле. Если указан пин-код, то при выкупе заказа зритель должен назвать его кассиру. В противном случае заказ не будет распечатан.* ***Пин-код формируется сайтом и передается в шлюз.***
- **Delivery -** Наличие/отсутствие доставки (1 курьерская доставка, 0 - нет доставки) *(необязательное поле при отсутствии доставки)*
- **DeliveryDate -** Дата доставки *(необязательное поле при отсутствии доставки)*
- **Address -** Адрес доставки *(необязательное поле при отсутствии доставки)*
- **MetroID -** ID станции метро. *(необязательное поле)*
- **OfficeID -** ID офиса. *(необязательное поле) Указывается при самовывозе (т.е. отсутсвии доставки). Это предполагаемый офис выкупа билетов. Если забронирован один или несколько "живых" билетов, то указать ID офиса обязательно.*
- **Notes -** Комментарий *(необязательное поле)*
- **OrderID -** ID заказа в БД сайта или платежной системе

(необязательное поле)
- **DeliveryService -** ID услуги доставки *(необязательное поле при отсутствии доставки)*
- **RemoteAddress -** IP адрес зрителя *(необязательное поле)*

Обратите внимание!
При наличии доставки поля: "Delivery" "DeliveryDate", "Address" и "DeliveryService" являются обязательными!

После оформления зрителем заказа, если продажа не последует, следует сменить номер сессии для зрителя!

Обратите внимание!

В случае ошибки или невозможности осуществить операцию, возвращаемые данные содержат только ID комплекта, ID сектора, ряд и место.

В случае удачного осуществеления операции возвращается полный список мест по всем событиям комлекта.

Описание полей в выходных данных
- **NomBilKn** - ID события.
- **cod_sec** ID сектора.
- **name_sec** Наименоване сектора
- **row** ряд (string[4]).
- **seat** место (string[4]).
- **price** - Цена билета. (номинал)*
- **priceSell** - Цена билета (продажи)*
- **ID Spectator** ID зрителя.
- **NameSpectator** ФИО зрителя
- **TelSpectator** Телефон зрителя
- **EmailSpectator** Email зрителя
- **Delivery** Наличие/отсутствие доставки (1 есть доставка, 0 - нет доставки)
- **DeliveryDate -** Предполагаемая дата доставки
- **Address -** Адрес доставки

- **MetroID -** ID станции метро.
- **Metro -** Наименование станции метро.
- **OfficeID -** ID офиса.
- **OfficeName -** Наименование офиса.

- **Notes -** Комментарий
- **OrderID -** ID заказа в БД сайта или платежной системе
- **DeliveryService -** ID услуги доставки
- **session -** ID сессии
- **ReservID** - номер заказа (в случае ошибки равен 0)
- **Barcode** - штрих-код заказа (он может быть распечатан и предъявлен в кассу для идентификации заказа)
- **ReservDate** - дата "выкупа" заказа (в случае ошибки пустая)
- **ReservTime** - время "выкупа" заказа (в случае ошибки пустое)
- **BlockTTL** - Время блокировки заказа в секундах. (Время, в течение которого заказ блокирован для любых действий с ним, кроме как через шлюз)
- **result_code** - Код ошибки.
- **result_message -** описание ошибки

ReservDate и **ReservTime** - дата и время "выкупа" заказа. По истечение данного времени заказ считается "просроченным" и может быть снят сотрудниками театра/агентства либо автоматическим сервисом. На данный момент времени, заказы, оформленные на доставку автоматически не снимаются.
После оплаты (метод SetSoldEventsSet) заказ не может быть снят и его дата и время "выкупа" становятся равной максимальной дате и времени мероприятий входящих в его состав.

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow EventsSetID="ID комплекта" cod_sec ="ID сектора" Row
="ряд" Seat ="место" NameSpektator="ФИО зрителя"
TelSpektator="Телефон зрителя" EmailSpektator="email зрителя"
session="ID сессии"/>
      ……
 </ReqBody>
```

```
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в возвращаемом
"RecordSet-e" -->
 </AnswerResult>
 <AnswerBody>
   <row NomBilKn="ID события" cod_sec ="ID сектора" row ="ряд" seat
="место" price="Цена билета" reservid="№ заказа" reservdate="дата
заказа" NameSpektator="ФИО зрителя" TelSpektator="Телефон зрителя"
EmailSpektator="Email зрителя" result_code="код ошибки"/>
```

2.2.10 Получить статус заказа

Получить статус заказа

Function **GetReservationStatus** (strInp: string): string;

Метод возвращает список текущий статус заказа.

Описание полей в выходных данных

- **ReservID -** Номер заказа

Описание полей в выходных данных

- **ReservStatus -** ID статуса заказа
- **ReservStatusDescription -** Описание статуса заказа
- **Qty -** Кол-во мест заказа
- **Amount -** Сумма заказа
- **ReservDate -** Дата аннулирования заказа
- **ReservTime -** Время аннулирования заказа

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow ReservID="номер заказа"/>
    ……
 </ReqBody>
</GateReq>
```

Пример выходного XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>5</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row ReservStatus="ID статуса заказа"
```

ReservStatusDescription="*описание статуса заказа*"/>
……
</AnswerBody>
</GateAnswer>

2.2.11 Получить статус заказа 2

Получить статус заказа 2

Function **GetReservationStatus** (strInp: string): string;

Метод возвращает список текущий статус заказа с учетов снятого заказа

Описание полей в выходных данных
- **ReservID -** Номер заказа

Описание полей в выходных данных
- **ReservStatus -** ID статуса заказа
- **ReservStatusDescription -** Описание статуса заказа
- **Qty -** Кол-во мест заказа
- **Amount -** Сумма заказа
- **ReservDate -** Дата истечения срока заказа
- **ReservTime -** Время истечения срока заказа

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow ReservID="номер заказа"/>
    ……
 </ReqBody>
</GateReq>
```

Пример выходного XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>5</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
```

```
   <row ReservStatus="ID статуса заказа"
ReservStatusDescription="описание статуса заказа"/>
      ……
 </AnswerBody>
</GateAnswer>
```

2.2.12 Снятие бронирования мест

Снятие бронирования мест

Function **FreeReservation** (strInp: string): string;

Метод осуществляет снятие бронирования указанных мест в БД при условии, что они находятся в состоянии бронирования с теми же реквизитами заказа, которые указаны при вызове.

Описание входных параметров.

- **NomBilKn** - ID события
- **cod_sec** ID сектора (int).
- **row** ряд (string[4])
- **seat** место (string[4])
- **session** - ID сессии
- **reservID** - номер заказа
- **RemoteAddress** - IP адрес зрителя *(необязательное поле)*

Описание полей в выходных данных

- **cod_sec** ID сектора (int).
- **row** ряд (string[4]).
- **seat** место (string[4]).
- **price** Цена билета. (Номинал)* (В случае ошибки равна 0 или пустая)
- **priceSell** - Цена билета (продажи)* (В случае ошибки равна 0 или пустая)
- **reservID** - номер заказа (в случае ошибки может быть равен 0)
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
   <InputRow NomBilKn="ID события" cod_sec ="ID сектора" row
="ряд" seat ="место" session="ID сессии" reservid="№ заказа"/>
      ……
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row NomBilKn="ID события" cod_sec ="ID сектора" row ="ряд"
seat ="место" price="Цена билета" reservid="№ заказа"
reservDate="Дата заказа" result_code="код ошибки"/>
       ……
 </AnswerBody>
</GateAnswer>
```

2.2.13 Снятие бронирования мест комплекта

Снятие бронирования мест комплекта

Function **FreeReservationEventsSet** (strInp: string): string;

Метод осуществляет снятие бронирования указанных мест в БД при условии, что они находятся в состоянии бронирования с теми же реквизитами заказа, которые указаны при вызове.

Описание входных параметров.

- **EventsSetID** - ID комплекта
- **cod_sec** ID сектора (int).
- **row** ряд (string[4])
- **seat** место (string[4])
- **session** - ID сессии
- **reservID** - номер заказа
- **RemoteAddress -** IP адрес зрителя *(необязательное поле)*

Описание полей в выходных данных

- **NomBilKn** - ID события
- **cod_sec** ID сектора (int).
- **row** ряд (string[4]).
- **seat** место (string[4]).
- **price** Цена билета. (Номинал)* (В случае ошибки равна 0 или пустая)
- **priceSell -** Цена билета (продажи)* (В случае ошибки равна 0 или пустая)
- **reservID** - номер заказа (в случае ошибки может быть равен 0)
- **result_code** - Код ошибки.
- **result_message -** описание ошибки

Обратите внимание!

В случае ошибки или невозможности осуществить операцию, возвращаемые данные содержат только ID комплекта, ID сектора, ряд и место.

В случае удачного осуществеления операции возвращается полный список мест по всем событиям комплекта.

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
   <InputRow EventsSetID="ID комплекта" cod_sec ="ID сектора" row
="ряд" seat ="место" session="ID сессии" reservid="№ заказа"/>
      ……
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row NomBilKn="ID события" cod_sec ="ID сектора" row ="ряд"
seat ="место" price="Цена билета" reservid="№ заказа"
reservDate="Дата заказа" result_code="код ошибки"/>
      ……
 </AnswerBody>
</GateAnswer>
```

2.2.14 Проверка промокодов

Проверка промокодов

Function CheckPromoCode (strInp: string): string;

Проверка валидности промокодов

Описание входных параметров.

- **NomBilKn** - ID события к которому применяется промокод
- **PromoCode** - Промокод

Описание полей в выходных данных

- **NomBilKn** - ID события, к которому применяется промокод
- **PromoCode** - Указанный промокод,
- **PromoCodeID** - ID промокода (если промокод невалидный - 0),
- **Percent** - Процент скидки по промокоду,
- **isPermitted** - Разрешено ли применение данного промокода (1 - разрешено, 2 - нет),
- **result_code** код ошибки,
- **result_message** - описание ошибки

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow PromoCode="ABC" NomBilKn="17227" />
    ……
 </ReqBody>
</GateReq>
```

Пример выходного XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>5</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <Row NomBilKn="17227" PromoCode="ABC" PromoCodeID="8"
```

```
Percent="5" isPermitted="1" result_code="0" result_message="OK"/>
   <Row NomBilKn="17228" PromoCode="ABC" PromoCodeID="0"
Percent="0" isPermitted="0" result_code="0" result_message="OK"/>
 </AnswerBody>
</GateAnswer>
```

2.3 Продажа и возврат

❖**CheckSoldTickets** - Проверка мест перед продажей
Проверяет места перед продажей на возможность таковой.

❖**SetSold** - Продажа места
Устанавливает метку о продаже места.

❖**SetSoldEventsSet** - Продажа места комплекта
Устанавливает метку о продаже места.

❖**SetReturn** - Возврат места
Возращает проданные места.

2.3.1 Продажа мест

Продажа мест

Function **SetSold** (strInp: string): string;

Метод устанавливает отметку о продаже указанных мест в БД при условии, что они находятся в состоянии бронирования.

Описание входных параметров.

- **NomBilKn** - ID события
- **cod_sec** ID сектора (int).
- **row** ряд (string[4])
- **seat** место (string[4])
- **session** - ID сессии
- **TransactionID** ID транзакции (string[25])
- **PaymentDate** Дата оплаты
- **PaymentTime** Время оплаты
- **RemoteAddress** - IP адрес зрителя *(необязательное поле)*
- **PromoCodeID** - ID промокода *(необязательное поле)*
- **PriceSell** - Цена продажи с учетом скидки по промокоду. Применяется только с промокодом *(необязательное поле)*

Описание полей в выходных данных

- **cod_sec** ID сектора (int).
- **row** ряд (string[4]).
- **seat** место (string[4]).
- **price** Цена билета (номинал).* (в случае ошибки равна 0)
- **PriceSell** Цена билета (продажи).* (в случае ошибки равна 0)
- **reservID** - номер заказа (в случае ошибки равен 0)
- **reservDate** - дата заказа (в случае ошибки пустая)
- **barcode** - штрих-код места
- **BarCodeType** - тип штрих-кода,
- **TransactionID** ID транзакции платежной системы.
- **PaymentDate** Дата оплаты
- **PaymentTime** Время оплаты
- **PromoCodeID** - ID промокода *(необязательное поле)*
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Обратите внимание!
Данный метод получает список мест заказа и устанавливает на них метку об оплате.

Все непереданные места будут удалены из заказа.

После оформления покупки, следует сменить номер сессии для зрителя!

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow NomBilKn="ID события" cod_sec ="ID сектора" row
="ряд" seat ="место" session="ID сессии" TransactionID="ID
Транзакции" PaymentDate="20.12.2014" PaymentTime="18:01:19" />
    ……
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row NomBilKn="ID события" cod_sec ="ID сектора"  row ="ряд"
seat ="место" price="Цена билета" reservID="№ заказа"
reservDate="Дата заказа" result_code="код ошибки"/>
    ……
 </AnswerBody>
</GateAnswer>
```

2.3.2 Проверка мест перед продажей

Проверка мест

Function **CheckSoldTickets** (strInp: string): string;

Метод осуществляет проверку уже забронированных мест, доступны ли они для продажи.

Описание полей во входных данных

- **NomBilKn** - ID События (int).
- **cod_sec** - ID сектора (int).
- **row** - ряд (string[4]).
- **seat** -место (string[4]).
- **price** - Цена билета.
- **reservID** - номер заказа
- **session** - ID сессии

Описание полей в выходных данных

- **NomBilKn** - ID События (int).
- **cod_sec** - ID сектора (int).
- **row** - ряд (string[4]).
- **seat** -место (string[4]).
- **price** - Цена билета (номинал)*.
- **priceSell** - Цена билета (продажи)*.
- **reservID** - номер заказа
- **session** - ID сессии,
- **blocked** - Заказ заблокирован и ожидает оплаты (да - 1, нет - 2)
- **result_code** - код возврата записи. (0 - место доступно для продажи)
- **result_message** - описание кода возврата ("OK" - место доступно для продажи)

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow NomBilKn="ID события" cod_sec ="ID сектора" row
="ряд" seat ="место" sessionID="ID сессии" ReservID="Номер заказа"
```

Price="*Цена*" **TransactionID**="*ID транзакции*" **PaymentDate**="*Дата оплаты*" **PaymentTime**="*Время оплаты*" />

……

</ReqBody>

</GateReq>

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
        <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
      запроса-->
        <RecordCount>0</RecordCount><!-- Количество записей в
      возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row NomBilKn="ID события" cod_sec ="ID сектора"  row ="ряд" seat
="место" price="Цена билета" reservID="№ заказа" reservDate="Дата
заказа" TransactionID="ID транзакции" PaymentDate="Дата оплаты"
PaymentTime="Время оплаты" result_code="код ошибки"/>
      ……
 </AnswerBody>
</GateAnswer>
```

2.3.3 Продажа мест комплекта

Продажа мест комплекта

Function **SetSoldEventsSet** (strInp: string): string;

Метод устанавливает отметку о продаже указанных мест в БД при условии, что они находятся в состоянии бронирования.

Описание входных параметров.

- **EventsSetID** - ID комплекта (int)
- **cod_sec** ID сектора (int).
- **row** ряд (string[4])
- **seat** место (string[4])
- **session** - ID сессии
- **TransactionID** ID транзакции (string[25])
- **PaymentDate** Дата оплаты
- **PaymentTime** Время оплаты
- **RemoteAddress** - IP адрес зрителя *(необязательное поле)*

Описание полей в выходных данных

- **NomBilKn** - ID события
- **cod_sec** ID сектора (int).
- **row** ряд (string[4]).
- **seat** место (string[4]).
- **price** Цена билета (номинал).* (в случае ошибки равна 0)
- **priceSell** Цена билета (продажи).* (в случае ошибки равна 0)
- **reservID** - номер заказа (в случае ошибки равен 0)
- **reservDate** - дата заказа (в случае ошибки пустая)
- **barcode** - штрих-код места
- **BarCodeType** - тип штрих-кода,
- **TransactionID** ID транзакции платежной системы.
- **PaymentDate** Дата оплаты
- **PaymentTime** Время оплаты
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Обратите внимание!

В случае ошибки или невозможности осуществить операцию, возвращаемые данные содержат только ID комплекта, ID сектора, ряд и место.

В случае удачного осуществеления операции возвращается полный список мест по всем событиям комплекта.

Обратите внимание!
Данный метод получает список мест заказа и устанавливает на них метку об оплате.
Все непереданные места будут удалены из заказа.

После оформления покупки, следует сменить номер сессии для зрителя!

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow EventsSetID="ID комплекта" cod_sec ="ID сектора"
row ="ряд" seat ="место" session="ID сессии" TransactionID="ID
Транзакции" PaymentDate="20.12.2014" PaymentTime="18:01:19" />
      ……
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row NomBilKn="ID события" cod_sec ="ID сектора"  row ="ряд"
seat ="место" price="Цена билета" reservID="№ заказа"
reservDate="Дата заказа" result_code="код ошибки"/>
      ……
 </AnswerBody>
</GateAnswer>
```

2.3.4 Возврат мест

Возврат мест

Function **SetReturn** (strInp: string): string;

Метод возвращает проданные места при условии, что они находятся в состоянии проданных.

Описание входных параметров.

- **NomBilKn** - ID события
- **cod_sec** ID сектора (int).
- **row** ряд (string[4])
- **seat** место (string[4])
- **session** - ID сессии
- **RemoteAddress** - IP адрес зрителя *(необязательное поле)*

Описание полей в выходных данных

- **NomBilKn** - ID события
- **cod_sec** ID сектора (int).
- **name_sec** Наименование сектора
- **row** ряд (string[4]).
- **seat** место (string[4]).
- **price** Цена билета (номинал)*. (в случае ошибки равна 0)
- **priceSell** Цена билета (продажи)*. (в случае ошибки равна 0)
- **barcode** - штрих-код места
- **reservID** - номер заказа (в случае ошибки равен 0)
- **reservDate** - дата заказа (в случае ошибки пустая)
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Обратите внимание!
Данный метод осуществляет возврат только на непрошедшие мероприятия или на отмененные и замененные мероприятия в течение 10 дней от даты мероприятия.

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
```

```
        <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
        <InputRow NomBilKn="ID события" cod_sec ="ID сектора" row
="ряд" seat ="место" session="ID сессии" />
        ……
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row NomBilKn="ID события" cod_sec ="ID сектора"  row ="ряд"
seat ="место" price="Цена билета" reservID="№ заказа"
reservDate="Дата заказа" result_code="код ошибки"/>
        ……
 </AnswerBody>
</GateAnswer>
```

2.4 Журналирование

❖**GetLog** - Фрагмент журнала операций.
Возвращает список операций за указанный временной интервал.

❖**GetCurrentState** - Сверка состояния проданных и забронированных мест
Возвращает состояние в системе Профтикет® проданных/ забронированных через шлюз мест за указанный временной интервал.

❖**GetReservationPlaceStatus** - Информация о текущем состоянии места заказа.
Возвращает текущее состояние места в указанном заказе.

2.4.1 Фрагмент журнала операций

Фрагмент журнала операций

Function **GetLog** (strInp: string): string;

Метод возвращает список записей из журнала операций для сессии с указанным идентификатором либо для всех сессий, если идентификатор не указан. В коллекцию включаются записи, совершенные в заданном интервале времени. Каждый элемент коллекции содержит перечень полей их журнала операций, значение каждого поля представлено в виде текстовой строки переменной длины.

Описание входных параметров.

- **DateFrom** - Дата начала временного интервала*
- **TimeFrom** - Время начала временного интервала*
- **DateTo** - Дата окончания временного интервала*
- **TimeTo** - Время окончания временного интервала*
- **Session** - Сессия* *(Необязательный параметр)*
- **ReservID** - Номер заказа *(Необязательный параметр)*
- **NomBilKn** - ID события *(Необязательный параметр)*
- **cod_sec** - ID сектора *(Необязательный параметр)*
- **Row** - ряд *(Необязательный параметр)*
- **Seat** - место *(Необязательный параметр)*
- **SalesOnly** - Показать только операции продажа/возврат *(Необязательный параметр) 0 (по-умолчанию) - все, 1 - только продажа/возврат*

Описание полей в выходных данных

- **ActionDate** - дата совершенной операции
- **ActionTime** - Время совершенной операции
- **ActionDone** - описание совершенной операции
- **NomBilKn** - ID события
- **cod_sec** - ID сектора
- **Row** - ряд
- **Seat** - место
- **Session** - Сессия*
- **price** - Цена билета фактическая
- **priceSell** - Цена продажи билета
- **ReservID** - номер заказа
- **ReservDate** - дата заказа
- **result_code** - код ошибки
- **result_message** - описание ошибки

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow DateFrom="dd.mm.yyyy" TimeFrom="hh:mm" DateTo
="dd.mm.yyyy" TimeTo="hh:mm"/>
      ……
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row ActionDate= "дата события" ActionTime="Время
события" ActionDone = "описание действия"
        NomBilKn= "ID события" cod_sec="ID сектора"
Row="ряд" Seat="место" Session="Сессия"
        price="цена" priceSell="цена продажи"  ReservID="№
заказа" ReservDate="дата заказа"
        result_code="код ошибки" result_message="описание
ошибки" />
      ……
 </AnswerBody>
</GateAnswer>
```

2.4.2 Сверка состояния проданных и забронированных мест

Сверка состояния проданных и забронированных мест

Function **GetCurrentState** (strInp: string): string;

Метод возвращает информацию о местах в соответствии с переданными параметрами. Для каждого места указывается состояние (свободно, забронировано, продано и т.д.), цена места, реквизиты заказа, дата продажи и т.п.

Описание входных параметров.
- **DateFrom** - Дата начала временного интервала*
- **TimeFrom** - Время начала временного интервала*
- **DateTo** - Дата окончания временного интервала*
- **TimeTo** - Время окончания временного интервала*
- **NomBilKn** - ID события (необязательный параметр)
- **cod_sec** - ID сектора (необязательный параметр)
- **Row** - ряд (необязательный параметр)
- **Seat** - место (необязательный параметр)

Поля DateFrom, TImeFrom, DateTo, TimeTo могут быть пустыми, но такой вариант передачи параметров является нежелательным, т.к. этом случае будет выбран весь диапазон мест, с которыми работал шлюз.

Описание полей в выходных данных
- **ActionDate** - дата совершенной операции
- **ActionTime** - Время совершенной операции
- **Status** - текущий статус места ("" - свободен | "SOL" продан | "RES" забронирован)
- **NomBilKn** - ID события
- **cod_sec** - ID сектора
- **Row** - ряд
- **Seat** - место
- **Price** - цена
- **ReservID** - текущий № заказа или номер заказа, по которому прошла продажа
- **ReservDate** - дата заказа

- **GateUser** - флаг того, что последний статус установлен текущим пользователем шлюза (0 | 1)

- **GateActionDate** - дата совершенной операции по данным шлюза
- **GateActionTime** - Время совершенной операции по данным шлюза
- **GateStatus** - текущий статус места по данным шлюза*
- **GateReservID** - номер заказа, место по которому бронировалось (снималась бронь) по данным шлюза.

- **result_code** - Код ошибки.
- **result_message -** краткое описание ошибки

***Статусы места по шлюзу:**
"*пусто*" - Место свободно
"**SEL**" - Место выделено (предварительное резервирование)
"**RES**" - Место забронировано
"**CRS**" - Место снято с брони
"**SOL**" - Место продано
"**RET**" - Место возвращено

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
            <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow DateFrom="dd.mm.yyyy" TimeFrom="hh:mm" DateTo
="dd.mm.yyyy" TimeTo="hh:mm"
                        [NomBilKn="ID события" cod_sec="ID сектора"
row="ряд" seat="место"/]>
      ……
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
        <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
      запроса-->
        <RecordCount>0</RecordCount><!-- Количество записей в
      возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row ActionDate= "дата события"
              ActionTime="Время события"
```

Status = "*текущий статус места*"
NomBilKn= "*ID события*"
Cod_sec="*ID сектора*"
Row="*ряд*"
Seat="*место*"
Price="*цена*"
ReservID="*№ заказа*"
ReservDate="*дата брони*"
GateUser="*флаг*"
GateStatus="*текущий статус места*"
GateReservID="*№ заказа*"
/>
……
</AnswerBody>
</GateAnswer>

2.4.3 Информация о текущем состоянии места заказа

Информация о текущем состоянии места

Function **GetReservationPlaceStatus** (strInp: string): string;

Метод возвращает текущее состояние места из конкретного заказа. Во входных параметрах можно передавать как одну запись, так и коллекцию записей для получения информации о нескольких местах одновременно.

Описание входных параметров.
- **ReservID** - Номер заказа
- **NomBilKn** - ID события
- **cod_sec** - ID сектора
- **Row** - ряд
- **Seat** - место

Описание полей в выходных данных
- **NomBilKn** - ID события
- **cod_sec** - ID сектора
- **Name_sec** - Наименование сектора
- **Row** - ряд
- **Seat** - место
- **ReservID** - номер заказа
- **ReservDate** - дата создания заказа*
- **ReservTime** - время создания заказа*
- **ChangeDate** - дата последнего изменения заказа*
- **ChangeTime** - время последнего изменения заказа*
- **Blocked** - признак блокировки заказа
- **PlacesStatus** - статус места (RES - забронировано, CRS - заказ или место из заказа сняты с брони, SOL - продано, RET - возвращено)
- **GateUser** - Признак, является ли пользователь, осуществивший операцию пользователем шлюза (1 - да, 2 - нет),
- **StatusComment** - Описание статуса
- **result_code** - Код ошибки.
- **result_message** - Описание ошибки

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
```

```
        <!-- Имя пользователя и пароль для авторизации -->
      <UserName>User</UserName>
      <UserPass>Password</UserPass>
 </ReqLogin>
 <ReqBody>
      <InputRow NomBilKn= "ID события" cod_sec="ID сектора"
Row="ряд" Seat="место"/>
      ……
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
   <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего
запроса-->
   <RecordCount>0</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-е" -->
 </AnswerResult>
 <AnswerBody>
   <row NomBilKn= "ID события" cod_sec="ID сектора"
Name_sec="Наименование сектора" Row="ряд" Seat="место"
        ReservID="№ заказа" ReservDate="дата заказа"
ChangeDate="дата изменения" ChangeTime="время изменения"

        Blocked="блокирован" PlacesStatus="статус места"
GateUser="пользователь шлюза"
StatusComment="комментарий"
        result_code="код ошибки" ResultMessage="Описание
ошибки" />
      ……
 </AnswerBody>
</GateAnswer>
```

3 Примеры

Примеры

- Работа с SOAP на языке PHP

В настоящее время многие разработчики сайтов используют PHP в качестве языка программирования для своего сайта. В данном руководстве мы приводим пример работы с SOAP на языке PHP.

- Пример запроса списка мест проведения (театров)

Пример простого запроса к шлюзу и ответ на этот запрос.

- Пример запроса списка залов места проведения (сцен театра)

Пример запроса с параметрами к шлюзу и ответ на этот запрос.

- Пример запроса списка объектов зала

Пример запроса на языке PHP к шлюзу для получения объектов зала.

3.1 Работа с SOAP на языке PHP

Работа с SOAP на языке PHP

Данный пример описывает работу с методом GetHallsList

```
/* Функция parse_xml нужна для обработки XML. */
function parse_xml($XMLTEXT) {
    $parser = xml_parser_create('');
    if(!$parser)
      return false;

    $xml_values = array();
    xml_parser_set_option($parser,
XML_OPTION_TARGET_ENCODING, 'UTF-8');
    xml_parser_set_option($parser,
XML_OPTION_CASE_FOLDING, 0);
    xml_parser_set_option($parser, XML_OPTION_SKIP_WHITE,
1);
    xml_parse_into_struct($parser, trim($XMLTEXT),
$xml_values);
    xml_parser_free($parser);
    $last_tag_ar = & $xml_array;
    $parents = array();
    $last_counter_in_tag = array(1=>0);

    foreach($xml_values as $data) {
    switch($data['type']) {
       case 'open':
         $last_counter_in_tag[$data['level']+1] = 0;
         $new_tag = array('name' => $data['tag']);
         if(isset($data['attributes']))
           $new_tag['attributes'] = $data['attributes'];

         if(isset($data['value']) && trim($data
['value']))
           $new_tag['value'] = trim($data['value']);

         $last_tag_ar[$last_counter_in_tag[$data
['level']]] = $new_tag;
         $parents[$data['level']] =& $last_tag_ar;
         $last_tag_ar =& $last_tag_ar
[$last_counter_in_tag[$data['level']]++];
       break;
       case 'complete':
         $new_tag = array('name' => $data['tag']);
         if(isset($data['attributes']))
           $new_tag['attributes'] = $data['attributes'];

         if(isset($data['value']) && trim($data
```

```
['value']))
            $new_tag['value'] = trim($data['value']);

          $last_count = count($last_tag_ar)-1;
          $last_tag_ar[$last_counter_in_tag[$data
['level']]++] = $new_tag;
        break;
        case 'close':
          $last_tag_ar =& $parents[$data['level']];
        break;
        default: break;
      }
}

return $xml_array;
 }

 /* Функция get_value_by_path  используется для обработки
дерева XML и заведению данных в массив */
function get_value_by_path($__xml_tree, $__tag_path) {
   $tmp_arr =& $__xml_tree;
   $tag_path = explode('/', $__tag_path);
   foreach($tag_path as $tag_name) {
        $res = false;
        foreach($tmp_arr as $key => $node)
        {
            if(is_int($key) && $node['name'] == $tag_name)
            {
                $tmp_arr = $node;
                $res = true;
                break;
            }
        }
        if(!$res)
            return false;
   }
   return $tmp_arr;
}
…

//Создание нового объекта и Установка соединения с
сервером SOAP.
$client = new SoapClient("http://192.168.1.2:8080/
scripts/STicketGate.exe/wsdl/ISTicket");
$login ='Gate'; //Логин
$pass = 'Gate'; //Пароль

$q = ' ?';
$quot = '"';
```

```
$cod_t = 8; //Код театра (получен ранее)

//Получаем массив со списков сцен театра с кодом «8».
$halls = get_value_by_path(parse_xml($client->GetHallList
('<?xml version="1.0" encoding="Windows-
1251"'.$q.'><GateReq><ReqLogin><UserName>'.$login.'</
UserName><UserPass>'.$pass.'</UserPass></
ReqLogin><ReqBody><InputRow cod_t="'.$cod_t.'"/></
ReqBody></GateReq>')),'GateAnswer/AnswerBody');


//Проходим по списку сцен
  for($i=0; $i< count($halls)-1; ++$i) {
   $hall=$halls[$i]['attributes'];
 echo utf8_win($hall['name_h']). '<BR>'; //Выводим на
печать наименование сцены
 echo utf8_win($hall['Address_h']). '<BR>'; //Выводим на
печать адрес сцены
 echo utf8_win($hall['Tel_h']) . '<BR>'; //Выводим на
печать тел. сцены
   }
```

3.2 Пример запроса списка мест проведения

Пример запроса списка мест проведения

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>Gate</UserName>
    <UserPass>dasjdfpojasdj</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow/>
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
  <ResultCode>0</ResultCode>
  <RecordCount>14</RecordCount>
 </AnswerResult>
 <AnswerBody>
   <Row cod_t="25" name_t="ET CETERA" result_code="0"/>
   <Row cod_t="14" name_t="БДТ" result_code="0"/>
   <Row cod_t="15" name_t="ДК МЭИ" result_code="0"/>
   <Row cod_t="20" name_t="Драмтеатр" result_code="0"/>
   <Row cod_t="27" name_t="Зал ГУУ" result_code="0"/>
   <Row cod_t="19" name_t="Музыкальный театр" result_code="0"/>
   <Row cod_t="12" name_t="МХТ им. А.П. Чехова" result_code="0"/>
   <Row cod_t="28" name_t="Саратовский драмтеатр" result_code="0"/>
   <Row cod_t="29" name_t="Театр им. Моссовета" result_code="0"/>
   <Row cod_t="16" name_t="Театр оперы и балета г.Воронеж"
result_code="0"/>
   <Row cod_t="21" name_t="Театр юного зрителя" result_code="0"/>
   <Row cod_t="24" name_t="Театр Юного зрителя г.Екатеринбург"
result_code="0"/>
   <Row cod_t="23" name_t="Театриум на Серпуховке" result_code="0"/>
   <Row cod_t="26" name_t="ЦДКЖ" result_code="0"/>
  </AnswerBody>
</GateAnswer>
```

3.3 Пример запроса списка залов места проведения

Пример запроса списка залов места проведения

Входной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateReq>
 <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
    <UserName>Gate</UserName>
    <UserPass>dasjdfpojasdj</UserPass>
 </ReqLogin>
 <ReqBody>
    <InputRow  cod_t="12"/>
 </ReqBody>
</GateReq>
```

Выходной XML.

```
<?xml version="1.0" encoding="Windows-1251"?>
<GateAnswer>
 <AnswerResult>
  <ResultCode>0</ResultCode>
  <RecordCount>4</RecordCount>
 </AnswerResult>
 <AnswerBody>
  <Row cod_th="31" name_h="МАЛАЯ  СЦЕНА" result_code="0"/>
  <Row cod_th="32" name_h="НОВАЯ  СЦЕНА" result_code="0"/>
  <Row cod_th="30" name_h="ОСНОВНАЯ  СЦЕНА" result_code="0"/>
  <Row cod_th="52" name_h="Саратовский драмтеатр" result_code="0"/>
 </AnswerBody>
</GateAnswer>
```

3.4 Пример запроса списка объектов зала

Пример запроса списка объектов зала

Данный пример описывает работу с методом GetSchemaHallList

```
header("Content-type: image/gif");

function parse_xml($XMLTEXT) {
  $parser = xml_parser_create('');
  if(!$parser)
    return false;

  $xml_values = array();
  xml_parser_set_option($parser,
XML_OPTION_TARGET_ENCODING, 'UTF-8');
  xml_parser_set_option($parser, XML_OPTION_CASE_FOLDING,
0);
  xml_parser_set_option($parser, XML_OPTION_SKIP_WHITE,
1);
  xml_parse_into_struct($parser, trim($XMLTEXT),
$xml_values);
  xml_parser_free($parser);
  $last_tag_ar = & $xml_array;
  $parents = array();
  $last_counter_in_tag = array(1=>0);

  foreach($xml_values as $data) {
    switch($data['type']) {
      case 'open':
        $last_counter_in_tag[$data['level']+1] = 0;
        $new_tag = array('name' => $data['tag']);
        if(isset($data['attributes']))
          $new_tag['attributes'] = $data['attributes'];

         if(isset($data['value']) && trim($data
['value']))
           $new_tag['value'] = trim($data['value']);

       $last_tag_ar[$last_counter_in_tag[$data
['level']]] = $new_tag;
        $parents[$data['level']] =& $last_tag_ar;
        $last_tag_ar =& $last_tag_ar[$last_counter_in_tag
[$data['level']]++];
      break;
      case 'complete':
        $new_tag = array('name' => $data['tag']);
        if(isset($data['attributes']))
          $new_tag['attributes'] = $data['attributes'];
```

```
        if(isset($data['value']) && trim($data
['value']))
          $new_tag['value'] = trim($data['value']);

        $last_count = count($last_tag_ar)-1;
        $last_tag_ar[$last_counter_in_tag[$data['level']]
++] = $new_tag;
      break;
      case 'close':
        $last_tag_ar =& $parents[$data['level']];
      break;
      default: break;
    }
  }

  return $xml_array;
}

function get_value_by_path($__xml_tree, $__tag_path) {
  $tmp_arr =& $__xml_tree;
  $tag_path = explode('/', $__tag_path);
  foreach($tag_path as $tag_name) {
    $res = false;
    foreach($tmp_arr as $key => $node) {
      if(is_int($key) && $node['name'] == $tag_name) {
        $tmp_arr = $node;
        $res = true;
        break;
      }
    }
    if(!$res)
      return false;
  }
  return $tmp_arr;
}

function utf8_win($s) {
  $r='';
  $state=1;
  for ($i=0;$i<strlen($s);$i++) {
    $c=ord($s[$i]);
    switch($state) {
      case 1: //not a special symbol
        if($c<=127) {
          $r.=$s[$i];
        }
        else {
          if(($c>>5)==6) {
            $c1=$c;
```

```
            $state=2;
          }
          else
            $r.=chr(128);
        }
        break;
      case 2: //an utf-8 encoded symbol has been meet
        $new_c2=($c1&3)*64+($c&63);
        $new_c1=($c1>>2)&5;
        $new_i=$new_c1*256+$new_c2;
        switch($new_i)
        {
          case   1025: $out_c='¨'; break;
          case   1105: $out_c='¸'; break;
          case 0x00ab: $out_c='«'; break;
          case 0x00bb: $out_c='»'; break;
          default: $out_c=chr($new_i-848);
        }
        $r.=$out_c;
        $state=1;
        break;
    }
  }
  return $r;
}

function win_utf8($in_text) {
   $output = "";
   $other[1025] = "¨";
   $other[1105] = "¸";
   $other[1028] = "ª";
   $other[1108] = "º";
   $other[1030] = "I";
   $other[1110] = "i";
   $other[1031] = "¯";
   $other[1111] = "¿";
   for ($i = 0; $i < strlen($in_text); $i++){
      if (ord($in_text{$i}) > 191) {
         $output.="&#".(ord($in_text{$i})+848).";";
      }else {
         if (array_search($in_text{$i}, $other)===false){
            $output.=$in_text{$i};
         }else {
            $output.="&#".array_search($in_text{$i},
$other).";";
         }
      }
   }
   return $output;
}
```

```
  $login='Gate';
  $pass = '123';


  $q = ' ?';
  $quot = '"';

  $client = new SoapClient("http://192.168.1.2:8080/40/
STicketGate.exe/wsdl/ISTicket");


  //Вызов метода...
  $HallShema = get_value_by_path(parse_xml($client-
>GetSchemaHallList('<?xml version="1.0"
encoding="windows-
1251"'.$q.'><GateReq><ReqLogin><UserName>'.$login.'</
UserName><UserPass>'.$pass.'</UserPass></
ReqLogin><ReqBody><InputRow NomBilKn="478" /></ReqBody></
GateReq>')),'GateAnswer/AnswerBody');

  //Считываем максимальные координаты в зале по X и Y
  $MaxX = $HallShema[0]['attributes']['MaxX']+22;
  $MaxY = $HallShema[0]['attributes']['MaxY']+22;


  //Рисуем картинку для зала
  $image = imagecreate($MaxX, $MaxY);
  $color = imagecolorallocate($image,192,192,192);

  $placecolor = imagecolorallocate($image,255,255,0);
  $labelcolor = imagecolorallocate($image,255,255,255);
  $linecolor = imagecolorallocate($image,00,00,00);

  $CurrentGroupIndex = 0;

  //Проходим по списку объектов зала
  for($i=0; $i< count($HallShema)-1; ++$i) {
    $object = $HallShema[$i]['attributes'];
    //Если объект - кресло
    if ($object['ObjectID'] == 0) {
      $place = imagecreatefromgif('place.gif');
      //Отрисовываем кресло
      imagecopymerge($image, $place, $object['CX']-floor
($object['PlaceSize'] / 2), $object['CY']-floor($object
['PlaceSize'] / 2), 0, 0, $object['PlaceSize'], $object
['PlaceSize'], 100);
      $delta = 3;
      switch (strlen($object['Seat'])) {
        case 1: $delta = 4; break;
```

```
        case 2: $delta = 6; break;
        case 3: $delta = 8; break;
      }
      //Пишем номер места на картике кресла
      imagettftext($image,8, 0,$object['CX']-$delta,
$object['CY']+3, $labelcolor, 'arial.ttf', $object
['Seat']);
    }
    //Если объект - метка
    if ($object['ObjectID'] == 2) {
      $labelcolor = imagecolorallocate($image,substr
($object['FontColor'],0,2) ,substr($object
['FontColor'],2,2) ,substr($object['FontColor'],4,2));
      imagettftext($image,$object['FontSize'], 0,$object
['CX'], $object['CY']+floor($object['PlaceSize'] / 4) ,
$labelcolor, 'arial.ttf', $object['Label']);

    }
    //Если объект - линия
    if ($object['ObjectID'] == 1) {
      if ($HallShema[$i+1]['attributes']['PointIndex'] !=
1  &&  $HallShema[$i+1]['attributes']['PointIndex'] != 0)
{
        $x = $HallShema[$i+1]['attributes']['CX'];
        $y = $HallShema[$i+1]['attributes']['CY'];
        imageline($image, $object['CX'], $object['CY'],
$x, $y, $linecolor);
      }
    }
  }
  imagegif($image);
  imagedestroy($image);
?>
```