

Шлюз ПО "Профтикет".

Описание методов

© 2025

ОГЛАВЛЕНИЕ

ОГЛАВЛЕНИЕ

Введение	5
1. Алгоритм вызова методов	6
2. Основные термины и понятия	8
3. Форматы данных	9
4. Коды возврата.	9
5. Операции шлюза	16
6. Статусы заказа	17
7. Распространенные ошибки	19
8. Абонементы	19
9. Комплекты	20
10. Промокоды	21
11. Штрихкоды	22
12. Абонементная кампания	22
13. Статусы событий и мероприятий	23
14. Сертификаты оплаты	24
15. Сегменты	24
16. Алгоритм заказа в связи с коронавирусом	24
Функциональные методы	25
17. Списки	29
17.1. Площадки/залы	31
17.2. Мероприятия	38
17.3. События	45
17.4. Изменения	79
17.5. Комплекты	85
17.6. Вспомогательные методы	97
17.7. Администрирование	103

17.8.	Скидки/Лояльность/Дисконт	107
17.9.	КЛАДР.....	113
17.10.	Доставка.....	116
18.	Бронирование.....	119
18.1.	Комплекты	120
18.2.	Предварительное резервирование мест	130
18.3.	Снятие предварительного резервирования мест	135
18.4.	Бронирование мест на событие	136
18.5.	Снятие бронирования мест	143
18.6.	Информация о заказе/местах	145
18.7.	Проверка.....	157
18.8.	Холдирование	161
19.	Продажа и возврат	164
19.1.	Продажа мест	165
19.2.	Проверка мест перед продажей.....	170
19.3.	Продажа мест комплекта	172
19.4.	Возврат мест	175
19.5.	Проверка мест перед возвратом	178
19.6.	Дополнительные данные о продаже	180
19.7.	Дополнительные данные о возврате	182
19.8.	Проверка БИНа карты оплаты для получения скидки	183
20.	Журналирование	185
20.1.	Фрагмент журнала операций.....	185
20.2.	Сверка состояния проданных и забронированных мест	187
21.	Сертификаты	190
21.1.	Список свободных номиналов сертификатов	190
21.2.	Предварительное резервирование сертификатов.....	191
21.3.	Бронирование сертификатов	194
21.4.	Снятие бронирования сертификатов	196
21.5.	Продажа сертификатов	197
21.6.	Возврат сертификатов	199

21.7.	Проверка сертификатов	200
22.	Регистрация посетителей	202
22.1.	Список типов документов посетителя	202
22.2.	COVID 19.....	203
22.3.	ДИТ - Мос.ру	210
23.	Зрители.....	215
23.1.	Изменение данных зрителя	215
23.2.	Получить данные по зрителю	217
23.3.	Список карт зрителя.....	219
23.4.	Валидация карты.....	221
24.	Приглашения	222
24.1.	Выдать приглашения (служебные места)	223
24.2.	Отмена приглашений (снять служебные места)	224
24.3.	Список сетов служебных мест.....	226
24.4.	Список мест сета служебных мест	227
Примеры		229
25.	Работа с SOAP на языке PHP	229
26.	Пример запроса списка мест проведения	232
27.	Пример запроса списка залов места проведения	233

ВВЕДЕНИЕ

1. Все методы шлюза **ПО "Профтикет"** (далее также - **шлюз**) реализованы в виде метода интерфейса SOAP. Каждый метод реализуется в виде функции стандартного формата:

`function MethodName(strInp: string): string; stdcall;` (*Pascal синтаксис*)

`string __stdcall MethodName(string strInp);` (*C++ синтаксис*)

2. Входной параметр метода **strInp** – строка в формате XML-формата, отличающаяся для каждого метода набором параметров.
3. 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.
4. Если какой-либо из входных параметров тега InputRow(см. далее) не задается, то он может быть указан как пустая строка ("") или не указан вовсе.
5. Удаленный пользователь авторизуется по указанному во входящем запросе логину и паролю.
6. Порядок атрибутов тегов **<InputRow>** (входные параметры) и **<Row>** (выходные параметры) – произвольный. Атрибуты должны обрабатываться по имени. Регистр входных параметров **имеет значение**. На практике, порядок выходных параметров (атрибутов тегов) может незначительно отличаться от приведенного в данном руководстве.
7. Параметры, указанные в квадратных скобках (например **[cod_show]**) являются необязательными.
8. Если метод использует [пагинацию](#), в ответе также присутствует тег **<AllRecords>**.
9. Во входящих параметрах всех методов можно передать IP адрес зрителя в теге **<IP>**:

```
<GateReq>
  <ReqLogin>
    <UserName>....</UserName>
    <UserPass>xxxxxx</UserPass>
  </ReqLogin>
  <IP>xxx.xxx.xxx.xxx</IP>
</ReqBody>...
```

10. 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!
11. В нынешней версии шлюз является только связующим звеном между сайтом и БД программного обеспечения «Профтикет». 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 и возврата билетов. Обработка, внешний вывод данных, а также подключение к платежной системе, целиком относится к подключаемому сайту.
12. 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.
13. 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. АЛГОРИТМ ВЫЗОВА МЕТОДОВ

- Для получения информации о мероприятиях, событиях и схемах залов используются методы [GetShowList](#), [GetEventList](#), [GetSchemaHallList](#) соответственно. **Т.к. данные, возвращаемые этими методами, содержат большой объем и изменяются реже, чем наличие свободных мест, настоятельно рекомендуется осуществлять вызов этих методов не чаще раз в час или вручную в случае необходимости.**
- Для осуществления бронирования и продажи необходимо получить информацию о свободных местах на мероприятия. Для этого воспользуйтесь методом [GetEventPlaceList](#).
- 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, методом FreePreReservation можно снять отметку о предварительном бронировании.
- Далее места необходимо забронировать, даже если затем они будут проданы. Для этого используется метод SetReservation.
- **Использование CheckSoldTickets, не является обязательным, но крайне желательным** и позволит избежать ошибок при оплате. Этот метод проверяет наличие доступных для оплаты мест. Соответственно он должен быть вызван непосредственно перед отправкой данных в платежную систему.
- При отказе от дальнейшей работы с местом можно снять отметку о бронировании места с помощью метода FreeReservation.
- Срок для автоматического снятия заказов указан в параметрах ReservDate и ReservTime метода SetReservation.
- 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.
- Если метод SetSold не смог установить отметку о продаже, рекомендуем вызвать этот метод несколько раз через 2-3 минуты. Также можно воспользоваться методом GetReservationStatus2 для проверки статуса заказа. Если он уже оплачен (это может быть, если SetSold послал запрос, но ответ не получит (например, в следствие падения сети)), то можно проставить оплату на стороне сайта (агента).
- Проверить состояние места и историю по данному месту можно следующими методами GetLog, и GetCurrentState.
- Проверить статус заказа можно с помощью метода GetReservationStatus2

Обратите внимание!

Указанный 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).

2. 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 Идентификатор 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. Строка 50 символов.
- **Ряд** - ряд передается строкой (макс. длина 4 символа), т.к. встречаются литерные ряды (пример: А, В, С)
- **Место** - место передается строкой (макс. длина 4 символа), т.к. встречаются литерные места (пример: 1А, 2В, 3С)
- **Идентификатор 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 зрелищное мероприятие, (подарки, программы, парковка и т.д.), так и обыкновенное мероприятие. События могут быть привязаны только к событиям, у которых нет признака "услуга" (is-Service). Событие-услуга может продаваться как отдельно, так и вместе с родительским событием.
- **Динамическая переоценка мест** - По достижении заданного кол-ва свободных мест цены мест меняются автоматически. Если у события выставлен соответствующий признак, следует обратить внимание, что цена этих мест может часто меняться.

3. ФОРМАТЫ ДАННЫХ

1. **Дата.** Даты в строках формата XML представляются строкой в формате "**дд.мм.гггг**" вне зависимости от настроек локализации сервера или клиента.
2. **Время.** Формат - "**чч:мм:сс**"
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.
5. Ряд и Место - строка 4 символа.
6. Идентификатор - как правило целое положительное число. Идентификатор сессии - строка 50 символов.

4. КОДЫ ВОЗВРАТА.

- 1** - Частично успешное действие. Используется в качестве результата всей операции. Означает, что часть запроса выполнена успешно, 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 соответствует имеющейся.
- 21** – Неправильные или пустые 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 или сняты или возвращены места. Использование указанной сессии для повторной операции запрещено.

37 - Часть выбранных 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.
- 46** - Превышена сумма контракта агента
- 47** - Текущая дата не входит в диапазон дат накладных, указанных в контракте агента
- 48** - Минимальная дата события выбранных мест не входит в диапазон дат, указанных в контракте агента
- 49** - Указанный сертификат не является валидным
- 50** - Сумма всех мест не равна сумме указанных сертификатов плюс размер доплаты
- 51** - Нет достаточного кол-ва сертификатов (или суммы) для продажи
- 52** - Кол-во мест определенного события в заказе превышает допустимое
- 53** - Сумма скидки по промокоду больше цены билета
- 54** - На место (места) события из абонементной кампании примен промокод
- 55** - Не указана цена продано
- 56** - Не указана цена возврата
- 57** - Цена возврата больше цены продажи
- 58** - Флаг DisableDiscount указан в одном запросе с разными значениями
- 59** - Флаг DisableDiscount в запросе не равен флагу, установленному для текущей сессии.
- 60** - Невозможно заблокировать места рядом с выбранными (введен в связи с пандемией коронавируса).
- 61** - Снятие мест из предрезерва/резерва возможно только полностью. (введен в связи с пандемией коронавируса).
- 62** - Переданы не все места сегмента.
- 63** - Посетителю больше 65 лет на момент начала события
- 64** - ФИО посетителя не указано
- 65** - Не указан тип документа посетителя
- 66** - Не указан документ посетителя

- 67** - Дата рождения посетителя не указана или некорректная
- 68** - Один или несколько параметров не подтверждены
- 69** - Дата регистрации на мероприятие больше текущей
- 70** - По данному билету/месту посетитель уже зарегистрирован
- 71** - Превышен максимальный процент заполнения зала
- 72** - Билет с указанным штрихкодом не найден или является непроданным.
- 73** - Билет был куплен у стороннего продавца, Для регистрации обратитесь к нему.
- 74** - Регистрация не нужна.
- 75** - Превышено кол-во проданных мест по абонементной кампании.
- 76** - Кол-во мест по Пушкинской карте не должно быть более одного.
- 77** - Требуется подтверждение, что все посетители по билетам данного заказа имеют QR код о вакцинации.
- 78** - В данном групповом заказе можно оплатить только 1 место за 1 операцию.
- 79** - Не указан RRN операции (продажи или возврата).
- 80** - Место заказа уже занято плательщиком.
- 81** - Не указано имя или контактные данные плательщика.
- 82** - Место уже свободно.
- 83** - Указанный плательщик еще не зарегистрирован.
- 84** - Место занято другим плательщиком.
- 85** - Для проверки промокода не указан Email.
- 86** - Указанный Email уже использован с данных промокодом.
- 87** - Указанный промокод связан с другой сессией.
- 88** - Указанный сертификат не является проданным.
- 89** - Указанная карта не валидна.
- 90** - Кол-во или сумма сертификатов в заказе не совпадает с переданными.
- 91** - Указанный заказ (с сертификатами) не может быть продан, так как отменен.

92 - Указанный заказ (с сертификатами) не может быть отменен, так как уже отменен или продан.

93 - Указанный заказ (с сертификатами) для данной сессии не найден.

94 - По указанной сессии уже был сформирован заказ. Использование указанной сессии для повторной операции запрещено.

95 - Указанный заказ уже оплачен.

96 - В одном заказе есть места из обычной и эксклюзивной квоты.

97 - Не разрешено применять одновременно скидку по дисконтррой программе и другие виды скидок.

98 - Запрещено добавление одновременно мест из событий, которые можно оплатить сертификатом, и которые нельзя.

99 - Запрещено совместное использование сертификатов и Пушкинской карты.

100 - Указанные места нельзя оплачивать сертификатом.

101 - Указанная скидка не существует или не валидна.

102 - Данный вид скидки не допускает множественную продажу.

103 - Скидка не суммируется с промокодом или абонементной кампанией

104 - Возврат билетов, купленных по промокоду запрещен.

105 - Возврат билетов, купленных по сертификату запрещен.

106 - С указанным Email на данное событие уже куплен билет по Пушкинской карте.

107 - Повторный вызов метода по одной и той же сессии запрещен.

108 - Сертификат просрочен.

109 - Продажа по Пушкинской карте отменена. Повторите платеж. Обратите внимание, ФИО должно\нбыть введено как в паспорте. **

1091 - Продажа по Пушкинской карте отменена. По указаной карте билет на выбранное вами событие уже куплен!**

1092 - Продажа по Пушкинской карте отменена. Максимальная цена в Прокульта меньше цены билета**

110 - Продажа по Пушкинской карте на событие запрещена или нет достаточно данных для проверки.

111 - Продажа по Пушкинской карте отменена. Повторите платеж.

1111 - Продажа по Пушкинской карте отменена. Повторите платеж.

1112 - Проверка не пройдена.

1113 - Проверка Пушкинской карты. Ошибка валидации.

1114 - Проверка Пушкинской карты. Токен не найден.

1115 - Проверка Пушкинской карты. Транзакция еще не получена.

1116 - Проверка Пушкинской карты. Неизвестная ошибка

1117 - Проверка Пушкинской карты. Не удалось подключиться к сервису

1118 - Проверка Пушкинской карты. Неустановленная ошибка

1119 - Проверка Пушкинской карты. Дата и время начала сеанса не попадают в общий период существования мероприятия

112 - Цена продано и цена продажи не равны.

113 - Переданы места из разных заказов

114 - Оплата на данное событие возможна только Пушкинской карте

1151 - Указано более 1 билета для поиска.

1152 - Билет для перерегистрации посетителя не найден.

1153 - Время события прошло.

1154 - Зарегистрирован проход посетителя в зал.

1155 - Событие отменено.

1156 - Регистрация на событие не нужна.

1157 - Билет куплен по Пушкинской карте. Перерегистрация запрещена.

1158 - Достигнуто максимальное кол-во перерегистраций билета.

1159 - Перерегистрация запрещена за x часов до начала события.

11591 - Указанный посетитель уже зарегистрирован на данное событие.

116 - Оплата на событие возможна только Пушкинской карте. Множественный выкуп запрещен.

117 - Пользователю запрещен вызов данного метода.

118 - На данном событии запрещена работа со служебными местами (приглашениями).

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

**** При появлении данного кода ошибки необходимо вернуть средства зрителю и снять бронь.**

Если зафиксирована ошибка, связанная с общей структурой входного XML-документа, с логином и паролем, правами на вызов того или иного метода, 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 спецификацией.

5. 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 отметки 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.

6. СТАТУСЫ ЗАКАЗА

Код	Статус	Описание
0	В обработке	<i>Заказ принят</i>
1	На доставке	<i>Заказ распечатан и передан в службу доставки</i>
2	Передан на доставку	<i>Заказ передан курьеру для доставки</i>
3	Заказ доставлен	<i>Заказ доставлен зрителю</i>
4	Заказ снят	<i>Заказ аннулирован</i>
5	Оплачен по б/н. Билеты не переданы клиенту	<i>Заказ оплачен по безналичному расчету. Бланки билетов еще не распечатаны.</i>
6	Оплачен. Билеты переданы клиенту	<i>Заказ оплачен за наличный расчет. Бланки билетов распечатаны</i>
7	Билеты распечатаны. Заказ ожидает передачу на доставку.	<i>Заказ распечатан полностью. Ожидается передача курьеру на доставку</i>
8	Заказ снят с доставки. Билеты не распечатаны.	<i>Заказ снят с доставки (= установлен на самовывоз), но не аннулирован.</i>
9	Заказ снят с доставки. Билеты распечатаны.	<i>Заказ снят с доставки (= установлен на самовывоз), но не аннулирован. Билеты распечатаны.</i>
10	Заказ ожидает оплату.	<i>Заказ принят и блокирован на время ожидания оплаты с сайта.</i>
11	Заказ оплачен по б/н. Билеты распечатаны	<i>Заказ оплачен по б/н. Билеты распечатаны и переданы клиенту</i>
12	Билеты распечатаны. За-	<i>Заказ распечатан полностью и передан на достав-</i>

	каз передан на доставку.	<i>ку курьеру</i>
13	Заказ был оплачен по б/н расчету. Билеты не распечатаны. Один или несколько билетов заказа были возвращены.	<i>Заказ оплачен по безналичному расчету. Бланки билетов не распечатаны. Часть (или все) мест из были возвращены.</i>
14	Заказ был оплачен по б/н расчету. Билеты распечатаны. Один или несколько билетов заказа были возвращены	<i>Заказ оплачен по безналичному расчету. Бланки билетов распечатаны. Часть (или все) мест из были возвращены.</i>

7. 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 удаления мест из заказа вместо FreeReservation
- Метод FreeReservation вызывается для возврата мест вместо SetReturn

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.

8. АБОНЕМЕНТЫ

Для работы с абонементом 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.

9. КОМПЛЕКТЫ

Комплекты мероприятий очень похожи на абонементы. Есть только несколько принципиальных 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 комплекта, 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 комплекта по отдельности, например, на 1 вернуть, а на два нет.

10. ПРОМОКОДЫ

Промокод позволяет получить скидку на билеты при создании заказа или его оплате.

Скидка по промокоду может быть как в процентном отношении (PromoPercent), так и фиксированной суммой (PromoSum).

Промокоды применяются к отдельным событиям, а также к секторам событий.

Также может существовать ограничение по цене билета.

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 выбирается промокод с наибольшим процентом (суммой) 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.

В метод предварительного резервирования (PreSetReservation) Идентификатор промокода передается только для ознакомления зрителя с возможной скидкой.

Окончательное формирование цены происходит в момент резервирования и (или) продажи.

При создании заказа (SetReservation), если необходимо передать промокод, передается его ID в поле PromoCodeID.

Такой же параметр используются при продаже (SetSold) по промокоду.

Промокод применяется только к цене продажи.

11. ШТРИХКОДЫ

Типы штрих-кодов, которые необходимо использовать в электронном билете. По-умолчанию используется Interleaved 2 of 5.

Штрих-код	Тип штрих-кода	Код штрих-кода
 12345678	Interleaved 2 of 5	bcCode_2_5_interleaved
 12345678	Code39	bcCode39
 12345678	Code128A	bcCode128A
 12345678	Code128B	bcCode128B
 12345678	Code128C	bcCode128C
 0000012345678	EAN13	bcEAN13
 12345678	EAN128A	bcEAN128A
 12345678	EAN128B	bcEAN128B
 12345678	EAN128C	bcEAN128C

12. АБОНЕМЕНТНАЯ КАМПАНИЯ

Если событие участвует в абонементной кампании то к билетам на это событие применяется скидка по этой кампании. Размер скидки устанавливается площадкой исходя из кол-ва билетов на событие при покупке.

Обратите внимание, что цена, передаваемая в методах предварительного резервирования (PreSetReservation), бронирования (SetReservation) и проверки мест перед продажей (CheckSoldTickets) является предварительной. Окончательная сумма будет рассчитана только при продаже билетов (SetSold)

В списке событий, имеющих в п продаже, передается признак принадлежности к абонементной кампании (isSeasonTicketCompany).

Промкоды на события, участвующие в абонементной кампании не распространяются.

13. СТАТУСЫ СОБЫТИЙ И МЕРОПРИЯТИЙ

События могут иметь следующие статусы:

- 0** - Событие отменено.
- 1** - Событие (не абонемент) доступно. Разрешена продажа и бронь.
- 2** - Событие (не абонемент) доступно. Разрешена только бронь.
- 3** - Событие (не абонемент) доступно. Разрешена только продажа.
- 4** - На событие относившееся к абонементу разрешены и бронь и продажа.
- 5** - На событие относившееся к абонементу разрешена только бронь.
- 6** - На событие относившееся к абонементу разрешена только продажа.
- 7** - Событие относится к абонементу и не продается/бронируется отдельно.
- 8** - Событие абонемент доступно. Разрешена продажа и бронь.
- 9** - Событие абонемент доступно. Разрешена только бронь.
- 10** - Событие абонемент доступно. Разрешена только продажа.

Мероприятия могут иметь следующие статусы:

- 0** - Нет доступных для продажи событий.
- 1** - Все или часть событий доступны только для бронирования.
- 2** - Все или часть событий доступны только для продажи.
- 3** - Все события доступны и для бронирования и для продажи.

4 - Только часть событий доступны и для бронирования и для продажи.

14. СЕРТИФИКАТЫ ОПЛАТЫ

Сертификаты оплаты используются для оплаты заказа.

Сертификаты могут быть куплены как на сайте, так и в кассе.

Каждый сертификат имеет срок действия, после истечения которого он аннулируется и не может быть использован при оплате.

Для оплаты заказа можно использовать несколько сертификатов. Номиналы суммируются.

Если сумма заказа превышает сумму номиналов сертификатов, то разница, которую необходимо доплатить, разбивается между всеми позициями заказа поровну.

Если сумма заказа меньше суммы номиналов, то разница сгорает и не возвращается.

Если при продаже указано несколько сертификатов и если сумма покупки покрывается только частью сертификатов, остальные не отмечаются как использованные и могут быть задействованы в дальнейшем.

Сертификаты используются для оплаты только через "собственный" сайт.

Методы для работы с сертификатами описаны здесь. Для продажи по сертификатам используются стандартные методы предварительного резервирования, бронирования, продажи.

15. СЕГМЕНТЫ

Сегмент это объединение нескольких мест в один условный блок (сегмент) для бронирования или продажи целиком.

При этом кол-во билетов равно количеству мест сегмента.

Возврат мест, входящих в сегмент может осуществляться по-одному. В этом случае сегмент расформировывается.

16. АЛГОРИТМ ЗАКАЗА В СВЯЗИ С КОРОНОВИРУСОМ

1. Создается предрезерв.
2. Создается резерв (заказ).

3. **Передаются данные по каждому посетителю с привязкой к местам.**

4. Осуществляется продажа (оплата) заказа.

ФУНКЦИОНАЛЬНЫЕ МЕТОДЫ

- [Списки](#)
 - [Площадки/залы](#)
 - [Список городов](#)
 - [Список мест проведения \(театров\)](#)
 - [Список залов места проведения \(сцен театра\)](#)
 - [Список секторов](#)
 - [Мероприятия](#)
 - [Список мероприятий \(спектаклей\)](#)
 - [Список категорий мероприятия](#)
 - [Список жанров мероприятий](#)
 - [Список жанров мероприятия](#)
 - [Список категорий мероприятий](#)
 - [События](#)
 - [Список организаторов событий](#)
 - [Список событий, имеющихся в продаже](#)
 - [Краткий список событий, имеющихся в продаже](#)
 - [Список мест сектора для события](#)
 - [Список объектов на событие](#)
 - [Список свободных мест на событие](#)
 - [Список дат доступных событий](#)
 - [Список связанных событий](#)

- [Список связанных событий по мероприятию](#)
- [Список событий связанного события](#)
- [Список свободных мест эксклюзивной квоты](#)
- [Информация о переоценке](#)
- [Изменения](#)
 - [Список изменившихся событий](#)
 - [Список изменений по событиям](#)
 - [Список событий с изменением цены](#)
 - [Список изменений по событиям для фильтра](#)
- [Комплекты](#)
 - [Список комплектов](#)
 - [Список объектов комплекта](#)
 - [Список свободных мест комплекта](#)
- [Вспомогательные методы](#)
 - [Список поставщиков](#)
 - [Список НДС](#)
 - [Схема возвратов](#)
 - [Настройки](#)
- [Администрирование](#)
 - [Список мест с указанием прохода](#)
 - [Список мест, на которые осуществлен проход](#)
- [Скидки/Лояльность/Дисконт](#)
 - [Схема начисления баллов по программе лояльности](#)
 - [Схема начисления скидок по дисконтной программе](#)
 - [Список эмоций](#)
 - [Список видов скидок \(скидочных карт\)](#)

- [Схема скидок по абонементной кампании](#)
- [КЛАДР](#)
 - [Список регионов](#)
 - [Список городов](#)
- [Доставка](#)
 - [Список станций метро](#)
 - [Список офисов](#)
 - [Список услуг доставки](#)
- [Бронирование](#)
 - [Комплекты](#)
 - [Предварительное резервирование мест комплекта](#)
 - [Снятие предварительного резервирования мест комплекта](#)
 - [Бронирование мест комплекта на событие](#)
 - [Снятие бронирования мест комплекта](#)
 - [Предварительное резервирование мест](#)
 - [Снятие предварительного резервирования мест](#)
 - [Бронирование мест на событие](#)
 - [Снятие бронирования мест](#)
 - [Информация о заказе/местах](#)
 - [Получить статус заказа](#)
 - [Получить статус заказа 2](#)
 - [Информация о заказе по его номеру](#)
 - [Получить статус мест сессии](#)
 - [Информация о текущем состоянии места заказа](#)
 - [Проверка](#)
 - [Проверка промокодов](#)

- [Проверка скидочной карты](#)
- [Холдирование](#)
- [Удержание мест за плательщиком](#)
- [Снятие удержания мест](#)
- [Продажа и возврат](#)
 - [Продажа мест](#)
 - [Проверка мест перед продажей](#)
 - [Продажа мест комплекта](#)
 - [Возврат мест](#)
 - [Проверка мест перед возвратом](#)
 - [Дополнительные данные о продаже](#)
 - [Дополнительные данные о возврате](#)
 - [Проверка БИНа карты оплаты для получения скидки](#)
- [Журналирование](#)
 - [Фрагмент журнала операций](#)
 - [Сверка состояния проданных и забронированных мест](#)
- [Сертификаты](#)
 - [Список свободных номиналов сертификатов](#)
 - [Предварительное резервирование сертификатов](#)
 - [Бронирование сертификатов](#)
 - [Снятие бронирования сертификатов](#)
 - [Продажа сертификатов](#)
 - [Возврат сертификатов](#)
 - [Проверка сертификатов](#)
- [Регистрация посетителей](#)
 - [Список типов документов посетителя](#)

- [COVID 19](#)
 - [Обновить данные заказа](#)
 - [Информация о билете](#)
 - [Регистрация посетителя](#)
- [ДИТ - Мос.ру](#)
 - [Поиск места по штрихкоду и номеру заказа/чек-коду](#)
 - [Обновить данные посетителя](#)
- [Зрители](#)
 - [Изменение данных зрителя](#)
 - [Получить данные по зрителю](#)
 - [Список карт зрителя](#)
 - [Валидация карты](#)
- [Приглашения](#)
 - [Выдать приглашения \(служебные места\)](#)
 - [Отмена приглашений \(снять служебные места\)](#)
 - [Список сетов служебных мест](#)
 - [Список мест сета служебных мест](#)

17. СПИСКИ

- [Площадки/залы](#)
 - [Список городов](#)
 - [Список мест проведения \(театров\)](#)
 - [Список залов места проведения \(сцен театра\)](#)
 - [Список секторов](#)
- [Мероприятия](#)

- [Список мероприятий \(спектаклей\)](#)
- [Список категорий мероприятия](#)
- [Список жанров мероприятий](#)
- [Список жанров мероприятия](#)
- [Список категорий мероприятий](#)
- [События](#)
 - [Список организаторов событий](#)
 - [Список событий, имеющих в продаже](#)
 - [Краткий список событий, имеющих в продаже](#)
 - [Список мест сектора для события](#)
 - [Список объектов на событие](#)
 - [Список свободных мест на событие](#)
 - [Список дат доступных событий](#)
 - [Список связанных событий](#)
 - [Список связанных событий по мероприятию](#)
 - [Список событий связанного события](#)
 - [Список свободных мест эксклюзивной квоты](#)
 - [Информация о переоценке](#)
- [Изменения](#)
 - [Список изменившихся событий](#)
 - [Список изменений по событиям](#)
 - [Список событий с изменением цены](#)
 - [Список изменений по событиям для фильтра](#)
- [Комплекты](#)
 - [Список комплектов](#)
 - [Список объектов комплекта](#)

- [Список свободных мест комплекта](#)
- [Вспомогательные методы](#)
 - [Список поставщиков](#)
 - [Список НДС](#)
 - [Схема возвратов](#)
 - [Настройки](#)
- [Администрирование](#)
 - [Список мест с указанием прохода](#)
 - [Список мест, на которые осуществлен проход](#)
- [Скидки/Лояльность/Дисконт](#)
 - [Схема начисления баллов по программе лояльности](#)
 - [Схема начисления скидок по дисконтной программе](#)
 - [Список эмоций](#)
 - [Список видов скидок \(скидочных карт\)](#)
 - [Схема скидок по абонементной кампании](#)
- [КЛАДР](#)
 - [Список регионов](#)
 - [Список городов](#)
- [Доставка](#)
 - [Список станций метро](#)
 - [Список офисов](#)
 - [Список услуг доставки](#)

17.1. ПЛОЩАДКИ/ЗАЛЫ

- [Список городов](#)
- [Список мест проведения \(театров\)](#)

- [Список залов места проведения \(сцен театра\)](#)
- [Список секторов](#)

17.1.1. СПИСОК ГОРОДОВ

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

Метод возвращает список имеющихся городов

Описание полей в выходных данных

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

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
```

```
<Row CityId="Id города" CityName="Наименование города"
result_code="0" result_message="OK" />
</AnswerBody>
</GateAnswer>
```

17.1.2. СПИСОК МЕСТ ПРОВЕДЕНИЯ (ТЕАТРОВ)

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

Метод возвращает список имеющихся мест проведения (театров)

Описание полей в выходных данных

- **cod_t** - Идентификатор театра
- **name_t** Наименование театра
- **Address_t** Адрес театра
- **Tel_t** Телефон театра
- **Fax_t** Факс театра
- **Mail_t** Email театра
- **Web_t** Адрес сайта театра
- **cod_city** Идентификатор города, в котором расположен театр
- **name_city** Наименование города, в котором расположен театр
- **CityId** Идентификатор города, в котором расположен театр
- **CityName** Наименование города, в котором расположен театр
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
  <ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
```

```
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</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_t="http://адрес театра сайта"
cod_city="Код города" name_city="Наименование города" re-
sult_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.1.3. СПИСОК ЗАЛОВ МЕСТА ПРОВЕДЕНИЯ (СЦЕН ТЕАТРА)

Function GetHallList(strInp: string): string;

Метод возвращает список сцен для указанного театра

Описание входных параметров.

- **[cod_t]** - Идентификатор театра

Описание полей в выходных данных

- **cod_t** Идентификатор театра
- **cod_th** - Идентификатор сцены театра

- **name_h** Наименование сцены театра
- **name_h2** Краткое наименование сцены театра
- **Address_h** адрес сцены театра
- **Tel_h** Телефон сцены театра
- **Email_h** Email сцены театра
- **Fax_h** Факс сцены театра
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Обратите внимание!

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

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow cod_t="2"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row cod_th="15" name_h="Наименование сцены театра" re-
```

```
sult_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.1.4. СПИСОК СЕКТОРОВ

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

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

Описание входных параметров.

- **[NomBilKn]** - Идентификатор события
- **[FreeOnly]** - признак все места/только выделенные в квоту.

Описание полей в выходных данных

- **cod_sec** - Идентификатор сектора
- **name_sec** - Наименование сектора
- **FreeOfferSeat** - Признак свободной рассадки (1 - да, 0 - нет)
- **PrintRows** - Печатать номер ряда на билете (1 - да, 0 - нет)
- **VisibleRows** - Отображать номер ряда на схеме зала (1 - да, 0 - нет)
- **PrintSeats** - Печатать номер места на билете (1 - да, 0 - нет)
- **VisibleSeats** - Показывать номер места на схеме (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

- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Если указан Идентификатор события, то метод возвращает все сектора, в которых есть доступные для бронирования / продажи места.

Если Идентификатор события не указан, метод возвращает полный список секторов.

Параметр FreeOnly актуален только для случая, когда передается Идентификатор 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.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow [NomBilKn="12"]/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row cod_sec="1" name_sec="Партеп" MinPrice="100" Max-
Price="1520" PlacesCount="143" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.2. МЕРОПРИЯТИЯ

- [Список мероприятий \(спектаклей\)](#)
- [Список категорий мероприятия](#)
- [Список жанров мероприятий](#)
- [Список жанров мероприятия](#)
- [Список категорий мероприятий](#)

17.2.1. 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]** Идентификатор мероприятия
- **[FreeOnly]** - Признак показывать только свободные. **Значение по-умолчанию: 1.**
- **[Offset]** - При выводе списка записей смещение от начала списка
- **[Limit]** - При выводе списка записей кол-во записей, которые необходимо вернуть

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** - Идентификатор мероприятия
- **name_show** - Наименование мероприятия
- **name_show2** - Краткое наименование мероприятия
- **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,
- **tag** - Тег(и).
- **age** - Возрастное ограничение, -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
- **MinPrice** - Минимальная Цена номинал событий мероприятия (только на места, доступные для текущего шлюза)
- **MinPriceSell** - Минимальная Цена продажи событий мероприятия (только на места, доступные для текущего шлюза)
- **MaxPrice** - Максимальная Цена номинал событий мероприятия (только на места, доступные для текущего шлюза)
- **MaxPriceSell** - Максимальная Цена продажи событий мероприятия (только на места, доступные для текущего шлюза)
- **cod_t** - Необязательная привязка мероприятия к месту проведения (театру). Если cod_t равен 0, то мероприятие не привязано к театру.
- **ProCultureID** - ID мероприятия в ПроКультура
- **PushkinCardPermitted** - Флаг разрешения продажи по программе "Пушкинская карта".
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
  <AllRecords>1</AllRecords>
</AnswerResult>
<AnswerBody>
  <Row result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.2.2. 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 мероприятиям

Описание входных параметров.

- **[code_show]** - Идентификатор мероприятия

Описание полей в выходных данных

- **code_show** - Идентификатор мероприятия,
- **code_categ** - Идентификатор категории,
- **name_categ** - Наименование категории
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow cod_show="24" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
    <AnswerResult>
        <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий
для всего запроса-->
        <RecordCount>3</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-e" -->
    </AnswerResult>
    <AnswerBody>
        <Row code_show="24" name_show="Синяя птица" code_categ="9"
name_categ="Детский спектакль" result_code="0" re-
sult_message="OK"/>
        <Row code_show="24" name_show="Синяя птица"
code_categ="15" name_categ="Мюзикл" result_code="0" re-
sult_message="OK"/>
    </AnswerBody>
</GateAnswer>
```

17.2.3. СПИСОК ЖАНРОВ МЕРОПРИЯТИЙ

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

Метод возвращает список жанров мероприятий

Описание полей в выходных данных

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

- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row code_genre="1" name_genre="Мьюзикл" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.2.4. СПИСОК ЖАНРОВ МЕРОПРИЯТИЯ

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

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

Описание входных параметров.

- **[code_show]** - Идентификатор мероприятия

Описание полей в выходных данных

- **code_genre** - Идентификатор жанра
- **name_genre** - Наименование жанра
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow cod_show="12" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row code_genre="1" name_genre="Мьюзикл" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.2.5. СПИСОК КАТЕГОРИЙ МЕРОПРИЯТИЙ

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

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

Описание полей в выходных данных

- **code_categ** - Идентификатор категории,
- **name_categ** - Наименование категории
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
  <AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий
для всего запроса-->
    <RecordCount>3</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-e" -->
  </AnswerResult>
  <AnswerBody>
    <Row code_categ="1" name_categ="Концерт" result_code="0"
result_message="OK"/>
    <Row code_categ="2" name_categ="Балет" result_code="0"
result_message="OK"/>
    <Row code_categ="3" name_categ="Цирк" result_code="0" re-
sult_message="OK"/>
  </AnswerBody>
</GateAnswer>
```

- [Список организаторов событий](#)
- [Список событий, имеющих в продаже](#)
- [Краткий список событий, имеющих в продаже](#)
- [Список мест сектора для события](#)
- [Список объектов на событие](#)
- [Список свободных мест на событие](#)
- [Список дат доступных событий](#)
- [Список связанных событий](#)
- [Список связанных событий по мероприятию](#)
- [Список событий связанного события](#)
- [Список свободных мест эксклюзивной квоты](#)
- [Информация о переоценке](#)

17.3.1. СПИСОК ОРГАНИЗАТОРОВ СОБЫТИЙ

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

Метод возвращает список имеющихся организаторов событий

Описание полей в выходных данных

- **OrganizerId** Идентификатор организатора
- **OrganizerName** Наименование организатора
- **OrganizerTin** ИНН организатора
- **OrganizerPhone** Телефон организатора
- **OrganizerEmail** Email организатора
- **OrganizerAddress** Адрес организатора
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
  <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
  <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
  <AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза,
общий для всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в
возвращаемом "RecordSet-e" -->
  </AnswerResult>
  <AnswerBody>
    <Row OrganizerId="36" OrganizerName="Организатор события"
OrganizerTin="77125212582" OrganizerPhone="+74951234567" Organ-
izerEmail="organizator@email.ru" OrganizerAddress="г. Москва.
ул. Кирова 12." result_code="0" result_message="OK" />
  </AnswerBody>
</GateAnswer>
```

17.3.2. 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]** - Идентификатор места проведения (театра)
- **[cod_th]** - Идентификатор зала места проведения (сцены)
- **[NomBilKn]** - Идентификатор события.
- **[FreeOnly]** - Признак показывать только свободные. По-умолчанию 1.
- **[Date]** - Дата, на которую вернуть список мероприятий. Время не учитывается.
- **[cod_show]** - Идентификатор мероприятия (спектакля).
- **[DateFrom]** - Дата начала вывода списка событий. Если время не указано, время начала считается с 00:00
- **[DateTo]** - Дата окончания вывода списка событий. Если время не указано, время окончания считается до 23:59:59
- **[SeasonTicketId]** - Идентификатор абонеента.
- **[FreeOnly]** - Флаг только свободные. По-умолчанию - 1.
- **[OffSet]** - При выводе списка записей смещение от начала списка
- **[Limit]** - При выводе списка записей кол-во записей, которые необходимо вернуть
- **[isService]** - Флаг, передавать в списке события-услуги. (1 - передавать только услуги, 0 - все, кроме услуг) По-умолчанию 0.
- **[isCanceled]** - Флаг, передавать отмененные события. (0 - передавать только действующие события 1 - передвать отмененные события, 2 - все) По-умолчанию 2.
- **[isSeasonTicketCompany]** - Флаг, показать события, участвующие в абонементной компании. (0 - передавать все события, 1 - передавать только события абонементной компании). По-умолчанию 0.
- **[name_show]** - Наименование мероприятия (спектакля) для поиска по вхожде-

нию.

Метод принимает только одну запись входящих параметров.

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

Метод возвращает большой массив данных (список событий с их параметрами и описаниями), поэтому не рекомендуется частый вызов данного метода.

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** - Идентификатор 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.
- **StopReturnDate** - Дата, после которой возврат билетов на данное событие запрещен.
- **cod_show** - Идентификатор мероприятия
- **cod_t** - Идентификатор места проведения (театра)
- **cod_h** - Идентификатор зала места проведения (сцены)
- **EventDuration** Продолжительность события (может отличаться от продолжительности мероприятия)
- **is_Primera** - Флаг, премьеры 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,
- **Tag** - Тег(и).
- **Age** - Возрастное ограничение, -1 - не определен. (если передается, например, 16, это означает 16+)
- **Producer** - Продюссер,
- **Actors** - Актеры,
- **Annotation** - Анотация,
- **OrganizerID** - Идентификатор организатора мероприятия,
- **OrganizerName** - Наименование организатора мероприятия,
- **OrganizerTin** - ИНН организатора мероприятия,
- **OrganizerPhone** - Телефон организатора мероприятия,
- **OrganizerEmail** - Email организатора мероприятия,
- **OrganizerAddress** - Адрес организатора мероприятия,
- **OrganizationProCultureID** - Идентификатор организации в ПроКультура,
- **MinPrice** - Цена номинал минимальная (только на места, доступные для текущего шлюза)
- **MinPriceSell** - Цена продажи минимальная (только на места, доступные для текущего шлюза)
- **MaxPrice** - Цена номинал максимальная (только на места, доступные для текущего шлюза)
- **MaxPriceSell** - Цена продажи максимальная (только на места, доступные для текущего шлюза)
- **FreePlacesQty** - Кол-во доступных для шлюза мест обычной (не эксклюзивной) квоты.
- **FreePlacesQtyExclusive** - Кол-во доступных для шлюза мест эксклюзивной квоты.
- **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 объединены с местами других событий в одном заказе)

- **MaxPlacesQty** - Кол-во мест в одном заказе на событие. (0 - нет ограничений, любое число - кол-во мест в одном заказе на это событие)

- **DeleteReservPlacesPermitted** - Разрешено частичное удаление мест из заказа. (1 - разрешено, 0 - запрещено)

- **isService** - Признак, является ли событие услугой

- **isCanceled** - Признак отмены мероприятия

- **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 (0- не существуют, 1 существует хотя бы один не требующий ввода email, 2 - существует хотя бы один, требующий ввода Email),

- **SalesByQty** - Продажа мест по кол-ву,

- **isShowBySectors** - Признак отображения схемы зала по секторам или полностью (0 - полностью, 1 - по секторам)

- **isRevaluation** - Признак динамической переоценки мест.

- **isReservPermitted** - Разрешена бронь на событие.
- **isSalesPermitted** - Разрешена продажа на событие.
- **isSeasonTicketCompany** - Признак принадлежности к абонементной кампании.
- **SeasonTicketCompanyMin** - Минимальное кол-во билетов для получения скидки по абонементной кампании.
- **VatId** - Идентификатор НДС
- **Status** - Статус события.
- **isSoldCertificates** - Разрешена оплата сертификатами.
- **BlockPlacesAround** - Признак блокировки мест (социальная рассадка).
- **CheckPersonalData** - Признак - при бронировании на событие запрашивать данные, передаваемые в ДИТ.
- **ProCultureID** - ID мероприятия в ПроКультура
- **PushkinCardPermitted** - Разрешение продажи по программе "Пушкинская карта". (0 - не разрешено, 1 - разрешено, 2 - только по "Пушкинской карте")
- **QRCodeOnly** - Признак - доступ на событие разрешен только при предъявлении QR кода о вакцинации.
- **GroupOrderPermitted** - Признак - На событие разрешено (не разрешено) создавать групповые заказы,
- **DiscountProgram** - Событие участвует в дисконтной программе (только для собственного сайта),
- **LoyaltyProgram** - Событие участвует в программе лояльности (только для собственного сайта).
- **LoyaltyPoints** - Кол-во баллов по программе лояльности, начисляемых зрителю при покупке билетов на данное событие (только для собственного сайта).
- **ReturnDiscountPermit** - Разрешение возврата билетов, купленных по промокоду.
- **RequestVisitorData** - Запрашивать данные зрителя (0 - не запрашивать, 1 - запрашивать только ФИО, 2 - запрашивать ФИО + тип, серию и номер документа).
- **OfficialPlacesPermit** - Разрешение на выдачу служебных мест через шлюз на данном событии (только для собственного сайта)
- **result_code** - Код ошибки

- **result_message** - Описание ошибки

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.

Обратите внимание!

Если параметр isReservPermitted = 1 и isSalesPermitted = 0, на событие разрешена только бронь,

Если параметр isReservPermitted = 0 и isSalesPermitted = 1, на событие разрешена только продажа.

Если разрешена только продажа, порядок вызовов методов резерва и продажи остается неизменным.

Контроль за обязательным вызовом метода продажи лежит на стороне сайта.

Входной XML.

```
<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"]
    [DateFrom="20.03.2019"] [DateTo="20.04.2019"]/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
    всего запроса-->
    <RecordCount>2</RecordCount><!-- Количество записей в воз-
    вращаемом "RecordSet-e" -->
    <AllRecords>2</AllRecords>
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="2" EventDate="20.03.2025" EventTime="19:00"
    cod_show="12" cod_t="1" cod_h="3" ... result_code="0" re-
    sult_message="OK"/>
    <Row NomBilKn="3" EventDate="21.03.2025" EventTime="19:00"
    cod_show="15" cod_t="1" cod_h="4" ... result_code="0" re-
    sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.3.3. КРАТКИЙ СПИСОК СОБЫТИЙ, ИМЕЮЩИХСЯ В ПРОДАЖЕ

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

Метод возвращает список актуальных на момент запроса мероприятий.

Описание входных параметров.

- **[DateFrom]** - Дата начала вывода списка событий. Если время не указано, время начала считается с 00:00
- **[DateTo]** - Дата окончания вывода списка событий. Если время не указано,

время окончания считается до 23:59:59

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **FreePlacesQty** - Кол-во доступных для шлюза мест.
- **EventStatus** - Статус события.
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Статусы события:

0 - Событие завершено или завершена бронь или продажа на него

1 - Событие активно, свободных мест нет

2 - Событие активно, свободные места есть.

3 - Событие отменено

4 - Событие неактивно, свободные места есть

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
```

```
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="123" FreePlacesQty="85" EventStatus="2" re-
sult_code="0" result_message="OK"/>
  <Row NomBilKn="134" FreePlacesQty="37" EventStatus="0" re-
sult_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.3.4. 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** - Идентификатор события
- **[cod_sec]** - Идентификатор сектора

Описание полей в выходных данных

- **cod_hs** Идентификатор места
- **NomBilKn** - Идентификатор события
- **cod_sec** - Идентификатор сектора
- **name_sec** - Наименование сектора
- **row** ряд
- **seat** место
- **FreeOfferSeat** - признак свободной рассадки (1 - да, 0 - нет)

- **PrintRows** - Печатать номер ряда на билете (1 - да, 0 - нет)
- **VisibleRows** - Отображать номер ряда на схеме зала (1 - да, 0 - нет)
- **PrintSeats** - Печатать номер места на билете (1 - да, 0 - нет)
- **VisibleSeats** - Показывать номер места на схеме (1 - да, 0 - нет)
- **entranceId** - Идентификатор подъезда
- **entranceName** - Наименование подъезда
- **Price** - Цена номинал.
- **PriceSell** - Цена продажи.
- **EventsSetID** - Идентификатор комплекта (0 если не комплект)
- **isEventSet** - признак, что место относится к комплекту (1-да, 0 - нет)
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="12" cod_sec="1" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
```

```
<Row cod_hs="1" cod_sec="1" name_sec="Паптер" row="1"
seat="1" result_code="0" result_message="OK"/>
<Row cod_hs="2" cod_sec="1" name_sec="Паптер" row="1"
seat="2" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.3.5. 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 местом в ряду.

"Метка" - это текстовая метка используемая для написания наименования секторов, обозначения цены и т.д.

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **[cod_sec]** - Идентификатор сектора
- **[ShowLabels]** - Показывать метки и все отличные от мест объекты
- **[Offset]** - При выводе списка записей смещение от начала списка
- **[Limit]** - При выводе списка записей кол-во записей, которые необходимо вернуть

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**

Обратите внимание!

Так как схема мест и прочих объектов в зале изменяется редко, рекомендуется вызывать данный метод не чаще одного раза в час для конкретного события.

Описание полей в выходных данных

- **cod_hs** - Идентификатор объекта (уникален в рамках типа объекта),
- **NomBilKn** - Идентификатор события,
- **ObjectType** - тип объекта (Place - место, Label - метка, FreeOfferSeatObject**, StageSign***)
- **ObjectName** - тип объекта (Place - место, Label - метка, Rect - прямоугольник, Ellipse - эллипс****, 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 или любого другого объекта,
- **Row** - Ряд (Используется только для кресел),
- **Seat** - Место (Используется только для кресел),

- **cod_sec** - Идентификатор сектора (Используется только для кресел),
- **Name_sec** - Наименование сектора (Для меток и прочих объектов - текст внутри фигуры),
- **PrintRows** - Печатать номер ряда на билете (1 - да, 0 - нет)
- **VisibleRows** - Отображать номер ряда на схеме зала (1 - да, 0 - нет)
- **PrintSeats** - Печатать номер места на билете (1 - да, 0 - нет)
- **VisibleSeats** - Показывать номер места на схеме (1 - да, 0 - нет)
- **entranceId** - Идентификатор подъезда (Используется только для кресел),
- **entranceName** - Наименование подъезда (Используется только для кресел),
- **FreeOfferSeat** - признак свободной рассадки (1 - да, 0 - нет) (Используется только для кресел),
- **BackColor** - Цвет ценовой зоны или объекта. (Используется только для кресел)
- **FontColor** - Цвет шрифта (Для места не используется)
- **FontSize** - размер шрифта (Для места не используется)
- **Label** - Для места - наличие и расположение метки ряда. Для остальных объектов - выравнивание текста по горизонтали и вертикали*
- **LineWidth** - Толщина рамки (Для места не используется)
- **LineColor** - Цвет рамки (Для места не используется)
- **BasePoints** - Координаты вершин объекта (Координаты X и Y разделены точкой с запятой (;), вершины знаком прямая черта (|).
- **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
- **SegmentID** - Идентификатор сегмента
- **CovidBlock** - Признак места, заблокированного из-за правил, действующих в период коронавирусной пандемии. (0 — не заблокировано. 1 — заблокировано.)
- **Data** - Картинка в формате Base64

- **result_code** - Код ошибки
- **result_message** - Описание ошибки

*Label

Для кресла: метка ряда, которая находится рядом с креслом.

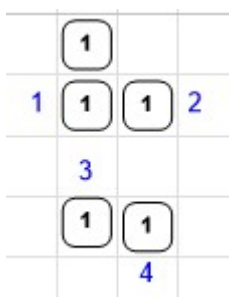
0 - нет метки ряда,

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

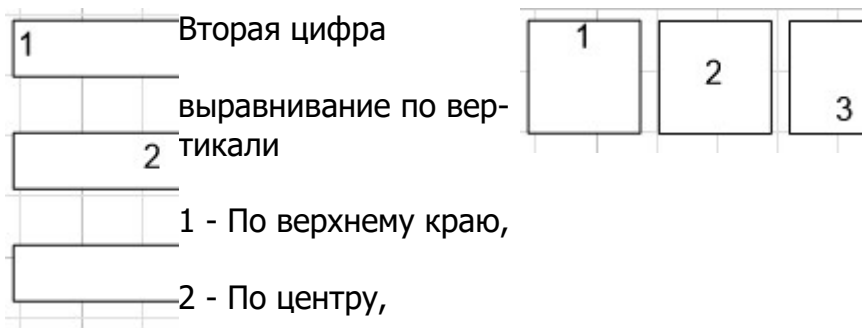
Вторая цифра

выравнивание по вертикали

1 - По верхнему краю,

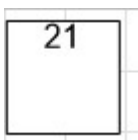
2 - По центру,

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 объект указывает на расположение сцены в макете.



******** Координаты, ширина и высота для эллипса передаются для прямоугольника, описанного вокруг эллипса.

Для отображения меток и объектов используются следующие атрибуты:

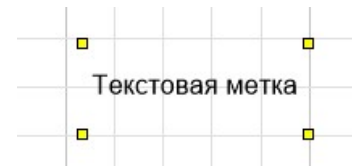
Место:

```
ObjectID="0" ObjectName="Place" ObjectType="Place"
```



Метка (надпись):

```
ObjectID="1" ObjectName="Label" ObjectType="Label"
```



Прямоугольник:

```
ObjectID="1" ObjectName="Rect" ObjectType="Label"
```



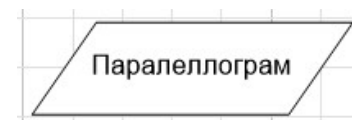
Эллипс:

```
ObjectID="1" ObjectName="Ellipse" ObjectType="Label"
```



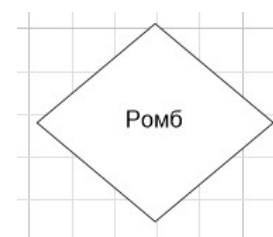
Параллелограмм:

```
ObjectID="1" ObjectName="Parallelogram" ObjectType="Parallelogram"
```



Ромб:

```
ObjectID="1" ObjectName="Rhombus" ObjectType="Rhombus"
```



Трапеция:

```
ObjectID="1" ObjectName="Trapezoid" Object
Type="Trapezoid"
```

**Линия:**

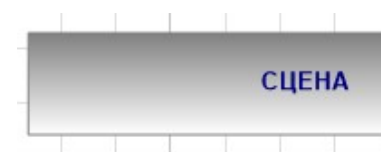
```
ObjectID="1" ObjectName="Line" Object
Type="Line"
```

**Полилиния:**

```
ObjectID="1" ObjectName="Polyline" Object
Type="Polyline"
```

**Указатель сцены:**

```
ObjectID="1" ObjectName="StageSign" Object
Type="StageSign"
```

**Указатель сектора со свободной рассадкой:**

```
ObjectID="2" ObjectName="FreeOfferSeatObject"
ObjectType="FreeOfferSeatObject"
```

**Картинка (Подложка):**

```
ObjectID="3" ObjectName="Image" Object
Type="SVG" | "PNG" | "JPG" | "GIF"
```

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="123" [cod_sec="1"] />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
```

```
всего запроса-->
    <RecordCount>9</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
    <AllRecords>9</AllRecords>
</AnswerResult>
<AnswerBody>
    <Row cod_hs="326607" NomBilKn="478" ObjectID="0" Object-
Name="Place" PlaceSize="22" Width="22" Height="22" PointIn-
dex="0" GroupPointIndex="0" CX="837" CY="155" Angle="0" Row="2"
Seat="3" cod_sec="1" Name_sec="ПАПТЕР" Label="" BackColor=""
FontColor="" FontSize="0" MinX="44" MinY="20" MaxX="1094"
MaxY="792" result_code="0" result_message="OK"/>
    <Row cod_hs="326608" NomBilKn="478" ObjectID="0" Object-
Name="Place" PlaceSize="22" Width="22" Height="22" PointIn-
dex="0" GroupPointIndex="0" CX="815" CY="155" Angle="0" Row="2"
Seat="4" cod_sec="1" Name_sec="ПАПТЕР" Label="" BackColor=""
FontColor="" FontSize="0" MinX="44" MinY="20" MaxX="1094"
MaxY="792" result_code="0" result_message="OK"/>
    <Row cod_hs="344773" NomBilKn="478" ObjectID="1" Object-
Name="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" FontCol-
or="000000" FontSize="0" LineWidth="1" LineColor="000000"
MinX="44" MinY="20" MaxX="1094" MaxY="792" result_code="0" re-
sult_message="OK"/>
    <Row cod_hs="344769" NomBilKn="478" ObjectID="1" Object-
Name="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" FontCol-
or="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094"
MaxY="792" result_code="0" result_message="OK"/>
    <Row cod_hs="344770" NomBilKn="478" ObjectID="1" Object-
Name="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" FontCol-
or="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094"
MaxY="792" result_code="0" result_message="OK"/>
    <Row cod_hs="344771" NomBilKn="478" ObjectID="1" Object-
Name="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" FontCol-
or="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094"
MaxY="792" result_code="0" result_message="OK"/>
    <Row cod_hs="344772" NomBilKn="478" ObjectID="1" Object-
Name="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" FontCol-
or="000000" FontSize="0" MinX="44" MinY="20" MaxX="1094"
MaxY="792" result_code="0" result_message="OK"/>
    <Row cod_hs="327415" NomBilKn="478" ObjectID="2" Object-
Name="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" result_code="0" result_message="OK"/>
    <Row cod_hs="326782" NomBilKn="478" ObjectID="2" Object-
Name="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" FontCol-
or="000000" FontSize="14" MinX="44" MinY="20" MaxX="1094"
MaxY="792" result_code="0" result_message="OK"/>

</AnswerBody>
</GateAnswer>
```

17.3.6. 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** - Идентификатор события
- **[cod_show]** - Идентификатор мероприятия. Может использоваться для проверки замены мероприятия у события.
- **[cod_sec]** - Идентификатор сектора
- **[Offset]** - При выводе списка записей смещение от начала списка
- **[Limit]** - При выводе списка записей кол-во записей, которые необходимо вернуть

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_hs** - Идентификатор места
- **cod_sec** - Идентификатор сектора
- **row** - ряд
- **seat** - место
- **FreeOfferSeat** - признак свободной рассадки (1 - да, 0 - нет)
- **NotShowNumbers** - Признак не показывать номера ряда и места (1 - не показывать, 0 - показывать)
- **PrintRows** - Печатать номер ряда на билете (1 - да, 0 - нет)
- **VisibleRows** - Отображать номер ряда на схеме зала (1 - да, 0 - нет)
- **PrintSeats** - Печатать номер места на билете (1 - да, 0 - нет)
- **VisibleSeats** - Показывать номер места на схеме (1 - да, 0 - нет)
- **Price** - Цена билета (номинал)**
- **PriceSell** - Цена билета (цена продажи)**
- **ServiceFee** - Сервисный сбор
- **PriceColor** - Цвет места
- **EventsSetID** - Идентификатор комплекта (0 если не комплект)
- **isEventSet** - признак, что место относится к комплекту (1-да, 0 - нет)
- **entranceId** - Идентификатор подъезда
- **entranceName** - Наименование подъезда,
- **SegmentID** - Идентификатор сегмента,
- **Exclusive** - Признак эксклюзивной квоты (0- простая, 1 - эксклюзивная)

- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="123"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
    <AllRecords>1</AllRecords>
</AnswerResult>
<AnswerBody>
    <Row cod_hs ="2352" cod_sec ="1" row ="1" seat ="2"
Price="1000" PriceSell="1000" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.3.7. СПИСОК ДАТ ДОСТУПНЫХ СОБЫТИЙ

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

Метод возвращает список дат актуальных на момент запроса событий

Описание входных параметров.

- **[FreeOnly]** - Признак показывать только свободные. По-умолчанию 1.

- **[DateFrom]** - Дата начала вывода списка событий. Время не учитывается.
- **[DateTo]** - Дата окончания вывода списка событий. Время не учитывается.
- **[cod_t]** - Идентификатор места проведения (театра)
- **[cod_th]** - Идентификатор зала места проведения (сцены)
- **[Offset]** - При выводе списка записей смещение от начала списка
- **[Limit]** - При выводе списка записей кол-во записей, которые необходимо вернуть
- **[isCanceled]** - Флаг, передавать отмененные события. (0 - передавать только действующие события 1 - передавать отмененные события, 2 - все). По-умолчанию 0.

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

- **EventDate** - Дата события(событий) в формате ГГГГ-ММ-ДД
- **cod_t** - Идентификатор места проведения (сцены)
- **name_t** - Наименование места проведения (сцены)
- **cod_h** - Идентификатор зала места проведения (сцены)

- **name_h** - Наименование зала места проведения (сцены)

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
    <AllRecords>1</AllRecords>
</AnswerResult>
<AnswerBody>
    <Row EventDate ="2025-07-09" cod_t="1" name_t="Городской
театр" cod_h="12" name_h="Малая сцена" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.3.8. СПИСОК СВЯЗАННЫХ СОБЫТИЙ

Function GetLinkedEventsList(strInp: string): string;

Метод возвращает связанные события

Описание входных параметров.

- **[ParentNomBilKn]** - Идентификатор "родительского" события. Если не указан, возвращается список всех связанных событий

Описание полей в выходных данных

- **NomBilKn** - Идентификатор 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 события с открытой датой,
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow ParentNomBilKn="1345"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="1404" EventDate="01.12.2025" EventTime="19:00"
BeginDate="01.12.2025" BeginTime="19:00" withOpenDate="0" re-
```

```
sult_code="0" result_message="OK" />
</AnswerBody>
</GateAnswer>
```

17.3.9. СПИСОК СВЯЗАННЫХ СОБЫТИЙ ПО МЕРОПРИЯТИЮ

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

Метод возвращает связанные события для указанного мероприятия.

Описание входных параметров.

- **cod_show** - Идентификатор мероприятия

Описание полей в выходных данных

- **NomBilKn** - Идентификатор 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 события с открытой датой,
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
```

```
<ReqBody>
    <InputRow cod_show="56"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="1404" EventDate="01.12.2025" EventTime="19:00"
BeginDate="01.12.2025" BeginTime="19:00" withOpenDate="0" re-
sult_code="0" result_message="OK" />
</AnswerBody>
</GateAnswer>
```

17.3.10. СПИСОК СОБЫТИЙ СВЯЗАННОГО СОБЫТИЯ

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

Метод возвращает список событий, которые используют связанное событие

Описание входных параметров.

- **LinkNomBilKn** - Идентификатор связанного события

Описание полей в выходных данных

- **NomBilKn** - Идентификатор связанного события,
- **LinkNomBilKn** - Идентификатор родительского 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),

- **cod_show** - Идентификатор мероприятия,

- **name_show** - Наименование мероприятия,

- **cod_t** - Идентификатор места проведения (театра),

- **name_t** - Наименование места проведения (театра),

- **cod_h** - Идентификатор зала места проведения (сцены),

- **name_h** - Наименование зала места проведения (сцены),

- **result_code** - Код ошибки.

- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow LinkNomBilKn="5"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="1404" LinkNomBilKn="5" EventDate="01.12.2025"
EventTime="19:00" BeginDate="01.12.2025" BeginTime="19:00"
cod_show="12" name_show="Спящая красавица" cod_t="1"
```

```
name_t="Городской театр" cod_h="2" name_h="Основная сцена" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.3.11. СПИСОК СВОБОДНЫХ МЕСТ ЭКСКЛЮЗИВНОЙ КВОТЫ

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

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

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **[cod_show]** - Идентификатор мероприятия. Может использоваться для проверки замены мероприятия у события.
- **[cod_sec]** - Идентификатор сектора
- **[Offset]** - При выводе списка записей смещение от начала списка
- **[Limit]** - При выводе списка записей кол-во записей, которые необходимо вернуть

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_hs** - Идентификатор места
- **cod_sec** - Идентификатор сектора
- **row** - ряд
- **seat** - место
- **FreeOfferSeat** - признак свободной рассадки (1 - да, 0 - нет)
- **NotShowNumbers** - Признак не показывать номера ряда и места (1 - не показывать, 0 - показывать)
- **PrintRows** - Печатать номер ряда на билете (1 - да, 0 - нет)
- **VisibleRows** - Отображать номер ряда на схеме зала (1 - да, 0 - нет)
- **PrintSeats** - Печатать номер места на билете (1 - да, 0 - нет)
- **VisibleSeats** - Показывать номер места на схеме (1 - да, 0 - нет)
- **Price** - Цена билета (номинал)**
- **PriceSell** - Цена билета (цена продажи)**
- **ServiceFee** - Сервисный сбор
- **PriceColor** - Цвет места
- **EventsSetID** - ID комплекта (0 если не комплект)
- **isEventSet** - признак, что место относится к комплекту (1-да, 0 - нет)
- **entranceId** - ID подъезда
- **entranceName** - Наименование подъезда,
- **SegmentID** - ID сегмента,
- **Exclusive** - Признак эксклюзивной квоты (0- простая, 1 - эксклюзивная)
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="123"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
    <AllRecords>1</AllRecords>
</AnswerResult>
<AnswerBody>
    <Row cod_hs ="2352" cod_sec ="1" row ="1" seat ="2"
Price="1000" PriceSell="1000" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.3.12. ИНФОРМАЦИЯ О ПЕРЕОЦЕНКЕ

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

Метод возвращает информацию о динамической переоценке.

Описание входных параметров.

- **NomBilKn** - Идентификатор события

Описание полей в выходных данных

- **PriceSell** - Цена продажи

- **PriceColor** - Цвет "ценового пояса"

- **Start1** - 1 диапазон - начальный процент
- **End1** - 1 диапазон - конечный процент
- **Price1** - 1 диапазон - цена

- **[Start2]** - 2 диапазон - начальный процент
- **[End2]** - 2 диапазон - конечный процент
- **[Price2]** - 2 диапазон - цена

- **[Start3]** - 3 диапазон - начальный процент
- **[End3]** - 3 диапазон - конечный процент
- **[Price3]** - 3 диапазон - цена

- **[Start4]** - 4 диапазон - начальный процент
- **[End4]** - 4 диапазон - конечный процент
- **[Price4]** - 4 диапазон - цена

- **[Start5]** - 5 диапазон - начальный процент
- **[End5]** - 5 диапазон - конечный процент
- **[Price5]** - 5 диапазон - цена

- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row PriceSell="1000" PriceColor="#FF00FF" Start1="90"
End1="80" Price1="1100" Start2="89" End2="60" Price2="1200"
Start3="59" End3="10" Price3="1300" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.4. ИЗМЕНЕНИЯ

- [Список изменившихся событий](#)
- [Список изменений по событиям](#)
- [Список событий с изменением цены](#)
- [Список изменений по событиям для фильтра](#)

17.4.1. СПИСОК ИЗМЕНИВШИХСЯ СОБЫТИЙ

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

Метод возвращает список событий в которых произошли любые изменения с указанного времени

Описание входных параметров.

- **UpdatedFromDate** - Дата начала отсчета изменений
- **UpdatedFromTime** - Время начала отсчета изменений

Описание полей в выходных данных

- **NomBilKn** - Идентификатор 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),
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow UpdatedFromDate="08.05.<%YEAR>" Updat-
edFromTime="12:15"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>2</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
```

```
<Row NomBilKn="1404" EventDate="01.06.<%YEAR>"
EventTime="19:00:00" result_code="0" result_message="OK" />
<Row NomBilKn="1416" EventDate="08.06.<%YEAR>"
EventTime="19:00:00" result_code="0" result_message="OK" />
</AnswerBody>
</GateAnswer>
```

17.4.2. СПИСОК ИЗМЕНЕНИЙ ПО СОБЫТИЯМ

Function GetEventChanges(strInp: string): string;

Метод возвращает информацию об изменениях в событиях

Описание входных параметров.

- **[NomBilKn]** - Идентификатор события
- **[cod_show]** - Идентификатор мероприятия
- **[LastOnly]** - Флаг показывать все зменения/ только последнее. (1 - только последнее, 0 - все). **по-умолчанию 1**

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_show** - Текущий Идентификатор мероприятия
- **cod_show_old** - Старый Идентификатор мероприятия
- **name_show** - Наименование мероприятия
- **name_show_old** - Наименование "старого мероприятия"
- **EventDate** - Текущая дата события
- **EventTime** - Текущее время события
- **ChangeDate** - Дата изменения
- **ChangeTime** - Время изменения
- **Canceled** - Признак отмены
- **Status** - Статус изменения - (0 - отмена, 1 - замена, 2 - перенос)

- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow [NomBilKn="1481"] />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="1481" cod_show="900" cod_show_old="900"
name_show="Приключения Буратино" name_show_old="Приключения
Буратино" EventDate="30.09.2025" EventTime="19:00" Event-
DateOld="20.09.2025" EventTimeOld="19:00"
ChangeDate="12.09.2025" ChangeTime="11:19" Canceled="0" re-
sult_code="0" result_message="OK"/>
    <Row NomBilKn="1483" cod_show="901" cod_show_old="901"
name_show="Мастер и Маргарита" name_show_old="Мастер и
Маргарита" EventDate="31.10.2025" EventTime="19:00" Event-
DateOld="30.10.2025" EventTimeOld="19:00"
ChangeDate="03.10.2025" ChangeTime="14:03" Canceled="0" re-
sult_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.4.3. СПИСОК СОБЫТИЙ С ИЗМЕНЕНИЕМ ЦЕНЫ

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

Метод возвращает список событий в которых произошли изменения цены с указанного времени

Описание входных параметров.

- **[FromDate]** - Дата начала отсчета изменений (если не указан - текущая дата)
- **[FromTime]** - Время начала отсчета изменений (если не указан - 00:00)
- **[ToDate]** - Дата окончания отсчета изменений (если не указан - текущая дата)
- **[ToTime]** - Время окончания отсчета изменений (если не указан - текущее время)

Описание полей в выходных данных

- **NomBilKn** - Идентификатор 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),
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow FromDate="08.05.<%YEAR>" FromTime="12:15" ToDate="08.05.<%YEAR>" ToTime="12:30"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
```

```
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>2</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="1404" EventDate="01.06.<%YEAR>"
EventTime="19:00:00" result_code="0" result_message="OK" />
  <Row NomBilKn="1416" EventDate="08.06.<%YEAR>"
EventTime="19:00:00" result_code="0" result_message="OK" />
</AnswerBody>
</GateAnswer>
```

17.4.4. СПИСОК ИЗМЕНЕНИЙ ПО СОБЫТИЯМ ДЛЯ ФИЛЬТРА

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

Метод возвращает информацию об изменениях в событиях

Описание входных параметров.

- **[NomBilKn]** - Идентификатор события
- **[cod_show]** - Идентификатор мероприятия
- **[EventsSetID]** - Идентификатор комплекта

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_show** - Идентификатор мероприятия
- **name_show** - Наименование мероприятия
- **EventsSetID** - Идентификатор комплекта
- **EventsSetName** - Наименование комплекта
- **EventDate** - Дата события
- **EventTime** - Время события
- **ChangeDate** - Дата изменения

- **ChangeTime** - Время изменения
- **result_code** - Код ошибки.
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow [NomBilKn="1481"] />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="1481" cod_show="900" name_show="Приключения
Буратино" EventDate="30.09.2025" EventTime="19:00"
ChangeDate="12.09.2025" ChangeTime="11:19" result_code="0" re-
sult_message="OK"/>
    <Row NomBilKn="1483" cod_show="901" name_show="Мастер и
Мартышка" EventDate="31.10.2025" EventTime="19:00"
ChangeDate="03.10.2025" ChangeTime="14:03" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.5. КОМПЛЕКТЫ

- [Список комплектов](#)
- [Список объектов комплекта](#)

- [Список свободных мест комплекта](#)

17.5.1. СПИСОК КОМПЛЕКТОВ

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

Метод возвращает список комплектов

Описание входных параметров.

- **[EventsSetID]** - Идентификатор комплекта
- **[ListOnly]** - показывать только список комплектов

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** Идентификатор комплекта
- **EventsSetName** Наименование комплекта
- **CountEvents** - Кол-во мероприятий, которое должно входить в комплект
- **NomBilKn** - Идентификатор события (отображается только если ListOnly = 0 или опущен)
- **cod_show** - Идентификатор мероприятия (отображается только если ListOnly = 0 или опущен)
- **cod_t** Идентификатор места проведения (театра)
- **cod_h** Идентификатор 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** - Наименование мероприятия (отображается только если ListOnly = 0 или опущен)

- **EventNote** - Примечания к событию (отображается только если ListOnly = 0 или опущен)

- **Annotation** - Анотация мероприятия (отображается только если ListOnly = 0 или опущен)

- **MinPrice** - Цена номинал минимальная (только на места, доступные для текущего шлюза) (отображается только если ListOnly = 1)

- **MinPriceSell** - Цена продажи минимальная (только на места, доступные для текущего шлюза) (только на места, доступные для текущего шлюза) (отображается только если ListOnly = 1)

- **MaxPrice** - Цена номинал максимальная (только на места, доступные для текущего шлюза) (только на места, доступные для текущего шлюза) (отображается только если ListOnly = 1)

- **MaxPriceSell** - Цена продажи максимальная (только на места, доступные для текущего шлюза) (только на места, доступные для текущего шлюза) (отображается только если ListOnly = 1)

- **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)

- **NonEventSetCount** - Кол-во мест на данное событие, не входящих в комплект (если ListOnly = 1) или на событие комплекта (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.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
```

```
<UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
  <InputRow [EventsSetID="12"] [ListOnly="1"] />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>2</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row EventsSetID="12" EventsSetName="Классическая музыка"
CountEvents="2" result_code="0" result_message="OK"/>
  <Row EventsSetID="12" EventsSetName="Сказки Венского леса"
CountEvents="3" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.5.2. СПИСОК ОБЪЕКТОВ КОМПЛЕКТА

Function GetSchemaHallListEventsSet(strInp: string): string;

Метод возвращает список записей о местах зала и метках на комплект.

"Метка ряда" - это метка, которая отображает номер ряда и нераздельно связана с первым или последним местом в ряду.

"Метка" - это текстовая метка используемая для написания наименования секторов, обозначения сцены и т.д.

Описание входных параметров.

- **EventsSetID** - Идентификатор комплекта
- **[cod_sec]** - Идентификатор сектора
- **[ShowLabels]** - Показывать метки и все отличные от мест объекты
- **[Offset]** - При выводе списка записей смещение от начала списка
- **[Limit]** - При выводе списка записей кол-во записей, которые необходимо вер-

нуть

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**

Обратите внимание!

Так как схема мест и прочих объектов в зале изменяется редко, рекомендуется вызывать данный метод не чаще одного раза в час для конкретного события.

Описание полей в выходных данных

- **EventsSetID** - Идентификатор события,
- **ObjectType** - тип объекта (Place - место, Label - метка, FreeOfferSeatObject**, StageSign***)
- **ObjectName** - тип объекта (Place - место, Label - метка, Rect - прямоугольник, Ellipse - эллипс****, 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 или любого другого объекта,
- **Row** - Ряд (Используется только для кресел),
- **Seat** - Место (Используется только для кресел),
- **cod_sec** - Идентификатор сектора (Используется только для кресел),
- **Name_sec** - Наименование сектора (Для меток и прочих объектов - текст внутри фигуры),
- **PrintRows** - Печатать номер ряда на билете (1 - да, 0 - нет)
- **VisibleRows** - Отображать номер ряда на схеме зала (1 - да, 0 - нет)
- **PrintSeats** - Печатать номер места на билете (1 - да, 0 - нет)
- **VisibleSeats** - Показывать номер места на схеме (1 - да, 0 - нет)
- **entranceId** - Идентификатор подъезда (Используется только для кресел),
- **entranceName** - Наименование подъезда (Используется только для кресел),
- **FreeOfferSeat** - признак свободной рассадки (1 - да, 0 - нет) (Используется только для кресел),
- **BackColor** - Цвет ценовой зоны или объекта. (Используется только для кресел)
- **FontColor** - Цвет шрифта (Для места не используется)
- **FontSize** - размер шрифта (Для места не используется)
- **Label** - Для места - наличие и расположение метки ряда. Для остальных объектов - выравнивание текста по горизонтали и вертикали*
- **LineWidth** - Толщина рамки (Для места не используется)
- **LineColor** - Цвет рамки (Для места не используется)
- **BasePoints** - Координаты вершин объекта (Координаты X и Y разделены точкой)

с запятой (;), вершины знаком прямая черта (|).

- **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
- **SegmentID** - Идентификатор сегмента
- **CovidBlock** - Признак места, блокированного из-за правил, действующих в период коронавирусной пандемии. (0 — не блокировано. 1 — блокировано.)
- **Data** - Картинка в формате Base64
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

*Label

Для кресла: метка ряда, которая находится рядом с креслом.

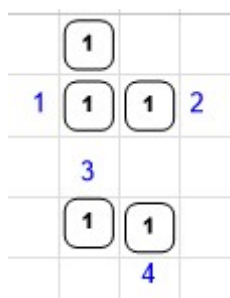
0 - нет метки ряда,

1 - метка ряда слева,

2 - метка ряда справа,

3 - метка ряда сверху,

4 - метка ряда снизу.



Для метки:

Первая цифра

1

Вторая цифра

1

2

3

выравнивание по горизонтали

2

выравнивание по вертикали

1 - По левому краю,

1 - По верхнему краю,

2 - По центру,

2 - По центру,

3 - По правому краю

3 - По нижнему краю

Например, 21

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 объект указывает на расположение сцены в макете.



******** Координаты, ширина и высота для эллипса передаются для прямоугольника, описанного вокруг эллипса.

Для отображения меток и объектов используются следующие атрибуты:

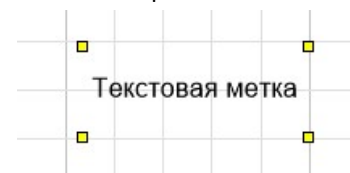
Место:

`ObjectID="0" ObjectName="Place" ObjectType="Place"`

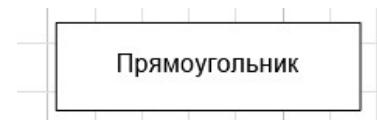


Метка (надпись):

```
ObjectID="1" ObjectName="Label" Object
Type="Label"
```

**Прямоугольник:**

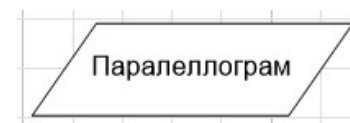
```
ObjectID="1" ObjectName="Rect" Object
Type="Label"
```

**Эллипс:**

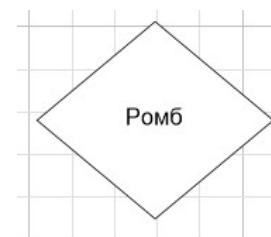
```
ObjectID="1" ObjectName="Ellipse" Object
Type="Label"
```

**Параллелограмм:**

```
ObjectID="1" ObjectName="Parallelogram" Object
Type="Parallelogram"
```

**Ромб:**

```
ObjectID="1" ObjectName="Rhombus" Object
Type="Rhombus"
```

**Трапеция:**

```
ObjectID="1" ObjectName="Trapezoid" Object
Type="Trapezoid"
```

**Линия:**

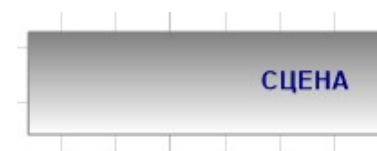
```
ObjectID="1" ObjectName="Line" Object
Type="Line"
```

**Полилиния:**

```
ObjectID="1" ObjectName="Polyline" Object
Type="Polyline"
```

**Указатель сцены:**

```
ObjectID="1" ObjectName="StageSign" Object
Type="StageSign"
```

**Указатель сектора со свободной рассадкой:**

```
ObjectID="2" ObjectName="FreeOfferSeatObject"
ObjectType="FreeOfferSeatObject"
```

**Картинка (Подложка):**

```
ObjectID="3" ObjectName="Image" Object
```

```
jectType="SVG"|"PNG"|"JPG"|"GIF"
```

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="123" [cod_sec="1"] />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>9</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
    <AllRecords>9</AllRecords>
</AnswerResult>
<AnswerBody>
    <Row EventsSetID="14" 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="ПАПТЕР" Label="" BackColor="" FontColor=""
FontSize="0" MinX="44" MinY="20" MaxX="1094" MaxY="792" re-
sult_code="0" result_message="OK"/>
    <Row EventsSetID="14" 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="ПАПТЕР" Label="" BackColor="" FontColor=""
FontSize="0" MinX="44" MinY="20" MaxX="1094" MaxY="792" re-
sult_code="0" result_message="OK"/>
    <Row EventsSetID="14" ObjectID="1" ObjectName="Point"
PlaceSize="0" Width="0" Height="0" PointIndex="1" GroupPointIn-
dex="1" CX="121" CY="452" Angle="0" Row="" Seat="" cod_sec="0"
Name_sec="" Label="" BackColor="000000" FontColor="000000" Font-
Size="0" LineWidth="1" LineColor="000000" MinX="44" MinY="20"
MaxX="1094" MaxY="792" result_code="0" result_message="OK"/>
    <Row EventsSetID="14" ObjectID="1" ObjectName="Point"
PlaceSize="0" Width="0" Height="0" PointIndex="2" GroupPointIn-
dex="1" CX="121" CY="473" Angle="0" Row="" Seat="" cod_sec="0"
```

```

Name_sec="" Label="" BackColor="000000" FontColor="000000" Font-
Size="0" MinX="44" MinY="20" MaxX="1094" MaxY="792" re-
sult_code="0" result_message="OK"/>
<Row EventsSetID="14" ObjectID="1" ObjectName="Point"
PlaceSize="0" Width="0" Height="0" PointIndex="3" GroupPointIn-
dex="1" CX="935" CY="473" Angle="0" Row="" Seat="" cod_sec="0"
Name_sec="" Label="" BackColor="000000" FontColor="000000" Font-
Size="0" MinX="44" MinY="20" MaxX="1094" MaxY="792" re-
sult_code="0" result_message="OK"/>
<Row EventsSetID="14" ObjectID="1" ObjectName="Point"
PlaceSize="0" Width="0" Height="0" PointIndex="4" GroupPointIn-
dex="1" CX="935" CY="297" Angle="0" Row="" Seat="" cod_sec="0"
Name_sec="" Label="" BackColor="000000" FontColor="000000" Font-
Size="0" MinX="44" MinY="20" MaxX="1094" MaxY="792" re-
sult_code="0" result_message="OK"/>
<Row EventsSetID="14" ObjectID="1" ObjectName="Point"
PlaceSize="0" Width="0" Height="0" PointIndex="5" GroupPointIn-
dex="1" CX="121" CY="297" Angle="0" Row="" Seat="" cod_sec="0"
Name_sec="" Label="" BackColor="000000" FontColor="000000" Font-
Size="0" MinX="44" MinY="20" MaxX="1094" MaxY="792" re-
sult_code="0" result_message="OK"/>
<Row EventsSetID="14" ObjectID="2" ObjectName="Label"
PlaceSize="0" Width="0" Height="0" PointIndex="0" GroupPointIn-
dex="0" CX="972" CY="616" Angle="0" Row="" Seat="" cod_sec="0"
Name_sec="" Label="ЛОЖА №1" BackColor="FFFFFF" FontCol-
or="000000" FontSize="11" MinX="44" MinY="20" MaxX="1094"
MaxY="792" result_code="0" result_message="OK"/>
<Row EventsSetID="14" ObjectID="2" ObjectName="Label"
PlaceSize="0" Width="0" Height="0" PointIndex="0" GroupPointIn-
dex="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" re-
sult_code="0" result_message="OK"/>

</AnswerBody>
</GateAnswer>

```

17.5.3. 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** - Идентификатор комплекта

- **[Offset]** - При выводе списка записей смещение от начала списка
- **[Limit]** - При выводе списка записей кол-во записей, которые необходимо вернуть

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

- **EventsSetID** - Идентификатор комплекта
- **cod_hs** - Идентификатор места
- **cod_sec** - Идентификатор сектора
- **row** - ряд
- **seat** - место
- **FreeOfferSeat** - признак свободной рассадки (1 - да, 0 - нет)
- **NotShowNumbers** - Признак не показывать номера ряда и места (1 - не показывать, 0 - показывать)
- **PrintRows** - Печатать номер ряда на билете (1 - да, 0 - нет)
- **VisibleRows** - Отображать номер ряда на схеме зала (1 - да, 0 - нет)
- **PrintSeats** - Печатать номер места на билете (1 - да, 0 - нет)

- **VisibleSeats** - Показывать номер места на схеме (1 - да, 0 - нет)
- **Price** - Цена билета (номинал)**
- **PriceSell** - Цена билета (цена продажи)**
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow EventsSetID="5"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
    <AllRecords>1</AllRecords>
</AnswerResult>
<AnswerBody>
    <Row EventsSetID ="5" cod_sec ="1" row ="1" seat ="5"
Price="500" PriceSell="500" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.6. ВСПОМОГАТЕЛЬНЫЕ МЕТОДЫ

- [Список поставщиков](#)
- [Список НДС](#)

- [Схема возвратов](#)
- [Настройки](#)

17.6.1. СПИСОК ПОСТАВЩИКОВ

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

Метод возвращает список поставщиков. Метод доступен только для собственного сайта.

Описание входных параметров.

- **SupplierID** - Идентификатор поставщика

Описание полей в выходных данных

- **SupplierName** - Наименование поставщика

- **SupplierID** - Идентификатор поставщика

- **SupplierAddress** - Адрес поставщика

- **SupplierPhone** - Телефон поставщика

- **SupplierTin** - ИНН поставщика

- **SupplierOGRN** - ОГРН поставщика

- **result_code** - Код ошибки.

- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
```

```
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row SupplierID="125" SupplierName="ФГБУК Театральный театр"
SupplierAddress="1211111 г. Москва. ул. Коекого 12." Suppli-
erPhone="+7 495-123-4567" SupplierTin="77123456789012" re-
sult_code="0" result_message="OK" />

</AnswerBody>
</GateAnswer>
```

17.6.2. СПИСОК НДС

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

Метод возвращает список НДС. Метод доступен только для собственного сайта.

Описание полей в выходных данных

- **VatId** - Идентификатор НДС
- **VatName** - Наименование НДС
- **Vat** - Размер (процент) НДС
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Обратите внимание!

При введении новых ставок НДС справочник будет дополняться.

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>7</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row VatId="-1" VatName="Без НДС" Vat="-1.00"/>
    <Row VatId="1" VatName="НДС 0%" Vat="0.00"/>
    <Row VatId="2" VatName="НДС 10%" Vat="10.00"/>
    <Row VatId="3" VatName="НДС 18%" Vat="18.00"/>
    <Row VatId="4" VatName="НДС 20%" Vat="20.00"/>
    <Row VatId="5" VatName="НДС 5%" Vat="5.00"/>
    <Row VatId="6" VatName="НДС 7%" Vat="7.00"/>
</AnswerBody>
</GateAnswer>
```

17.6.3. СХЕМА ВОЗВРАТОВ

Function GetReturnSchema(strInp: string): string;

Метод возвращает схему возвратов

Описание входных параметров.

- **NomBilKn** - Идентификатор события

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **DaysFrom** - Диапазон кол-ва дней до мероприятия начало
- **DaysTo** - Диапазон кол-ва дней до мероприятия окончание
- **PercentReturn** - Процент возврата
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="25"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>4</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="25" DaysFrom="10000" DaysTo="11" Percen-
tReturn="100" result_code="0" result_message="OK"/>
    <Row NomBilKn="25" DaysFrom="10" DaysTo="5" PercentRe-
turn="50" result_code="0" result_message="OK"/>
    <Row NomBilKn="25" DaysFrom="4" DaysTo="3" PercentRe-
turn="20" result_code="0" result_message="OK"/>
    <Row NomBilKn="25" DaysFrom="2" DaysTo="0" PercentRe-
turn="0" result_code="0" result_message="OK"/>
</AnswerBody>
```

```
</GateAnswer>
```

17.6.4. НАСТРОЙКИ

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

Метод возвращает настройки шлюза

Описание полей в выходных данных

- **BackgroundColor1** - Цвет фона 1
- **BackgroundColor2** - Цвет фона 2 (равен цвету фона 1, если заливка не градиентная)
- **RowLabelColor** - Цвет метки ряда
- **BusyPlacesColor** - Цвет занятого места
- **CheckPersonalData** - Проверять персональные и иные данные при заказе на сайте.
- **VersionNumber** - Номер версии ПО «Профтикет».
- **DiscountProgram** - Флаг наличия дисконтной программы
- **LoyaltyProgram** - Флаг наличия программы лояльности
- **ConcateDiscount** - Разрешено совмещать дисконтную программу и иные скидки
- **CertificateSurchargePermitted** - Разрешена доплата при оплате сертификатом
- **RequestVisitorData** - Запрашивать данные зрителя (0 - не запрашивать, 1 - запрашивать только ФИО, 2 - запрашивать ФИО + тип, серию и номер)
- **GetOnly** - Разрешен только просмотр справочников (0 - все методы, 1 - только справочники)
- **OfficialPlacesPermitted** - Разрешена выдача служебных мест (0 - запрещено, 1 - разрешено)
- **VisitorDataDelayedInput** - Разрешен ввод персональных данных посетителя после покупки билета (0 - запрещено, 1 - разрешено)
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row BackgroundColor1="C0C0C0" BackgroundColor2="FFFFFF" Row-
LabelColor="000080" BusyPlacesColor="808080" CheckPersonalDa-
ta="0" VersionNumber="4.1.0" DiscountProgram="1" LoyaltyPro-
gram="1" ConcatDiscount="0" CertificateSurchargePermitted="0"
RequestVisitorData="2" MosIdIntegration="1" GetOnly="0" Offi-
cialPlacesPermitted="1" VisitorDataDelayedInput="1" re-
sult_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.7. АДМИНИСТРИРОВАНИЕ

- [Список мест с указанием прохода](#)
- [Список мест, на которые осуществлен проход](#)

17.7.1. СПИСОК МЕСТ С УКАЗАНИЕМ ПРОХОДА

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

Метод возвращает список мест с учетом информации, какие места были проданы и на какие был осуществлен проход. Необходим для отображения информации для администратора о проходе в зрительный зал

Обратите внимание!

Метод доступен только для собственного сайта и должен вызываться только из административной части сайта

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **[cod_sec]** - Идентификатор сектора

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_sec** - Идентификатор сектора
- **name_sec** - Наименование сектора
- **row** - Ряд
- **seat** - Место
- **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,
- **AccessDate** - Дата первого прохода. Если пусто - проход не осуществлялся.
- **AccessTime** - Время первого прохода. Если пусто - проход не осуществлялся.
- **isInside** - Признак - Зритель находится в зале (0 - нет, 1 - да)
- **isSold** - Признак - Место продано (0 - нет, 1 - да).
- **ContAgentName** - Имя зрителя или наименование агента купившего билет
- **Barcode** - Штрихкод

- **isPushkinCard** - Флаг продажи по карте Пушкина

- **VisitorName** - Маскированное ФИО посетителя (в виде Имя Отчество (при наличии) Ф.)

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="24" [cod_sec="1"] />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>3</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="1504" cod_sec="1" name_sec="Паптер" row="1"
seat="1" CX="88" CY="88" CX2="152" CY2="152" Angle="0" Ac-
cessDate="12.02.2025" AccessTime="18:45:13" isInside="1" is-
Sold="1" />

    <Row NomBilKn="1504" cod_sec="1" name_sec="Паптер" row="1"
seat="2" CX="168" CY="88" CX2="232" CY2="152" Angle="0" Ac-
cessDate="" AccessTime="" isInside="0" isSold="1" />

    <Row NomBilKn="1504" cod_sec="1" name_sec="Паптер" row="1"
seat="3" CX="248" CY="88" CX2="312" CY2="152" Angle="0" Ac-
cessDate="" AccessTime="" isInside="0" isSold="1" Visi-
torName="Иван Иванович И." />

</AnswerBody>
</GateAnswer>
```

```
Function GetAccessedPlacesList(strInp: string): string;
```

Метод возвращает список мест, на которые осуществлен проход зрителей.

Обратите внимание!

Метод доступен только для собственного сайта и должен вызываться только из административной части сайта

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **[cod_sec]** - Идентификатор сектора

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_sec** - Идентификатор сектора
- **name_sec** - Наименование сектора
- **row** - Ряд
- **seat** - Место
- **AccessDate** - Дата прохода
- **AccessTime** - Время прохода
- **ContAgentName** - Имя зрителя или наименование агента купившего билет
- **Barcode** - Штрихкод
- **isPushkinCard** - Флаг продажи по карте Пушкина
- **VisitorName** - Маскированное ФИО посетителя (в виде Имя Отчество (при наличии) Ф.)
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="24" [cod_sec="1"] />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="1504" cod_sec="1" name_sec="Партер" row="1"
seat="1" AccessDate="12.02.2025" AccessTime="18:45:13" Visi-
torName="Иван Иванович И." result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.8. СКИДКИ/ЛОЯЛЬНОСТЬ/ДИСКОНТ

- [Схема начисления баллов по программе лояльности](#)
- [Схема начисления скидок по дисконтной программе](#)
- [Список эмоций](#)
- [Список видов скидок \(скидочных карт\)](#)
- [Схема скидок по абонементной кампании](#)

17.8.1. СХЕМА НАЧИСЛЕНИЯ БАЛЛОВ ПО ПРОГРАММЕ ЛОЯЛЬНОСТИ

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

Метод возвращает схему начисления баллов по программе лояльности

Описание полей в выходных данных

- **PositionName** - Наименование позиции
- **Points** - Кол-во баллов
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>3</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row PositionName="День рождения" Points="500" re-
sult_code="0" result_message="OK"/>
    <Row PositionName="Заполнение анкеты" Points="500" re-
sult_code="0" result_message="OK"/>
    <Row PositionName="Регистрация в личном кабинете"
Points="1000" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.8.2. СХЕМА НАЧИСЛЕНИЯ СКИДОВ ПО ДИСКОНТНОЙ ПРОГРАММЕ

Function GetSchemaPercents(strInp: string): string;

Метод возвращает схему начисления процентов по дисконтной программе

Описание полей в выходных данных

- **AmountFrom** - Начало диапазона
- **AmountTo** - Конец диапазона
- **DiscountPercent** - Процент скидки
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>3</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row AmountFrom="1000.00" AmountTo="10000.00" Dis-
countPercent="5.00" result_code="0" result_message="OK"/>
    <Row AmountFrom="10001.00" AmountTo="30000.00" Dis-
countPercent="10.00" result_code="0" result_message="OK"/>
    <Row AmountFrom="30001.00" AmountTo="100000000.00" Dis-
countPercent="15.00" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.8.3. СПИСОК ЭМОЦИЙ

Function GetEmotionsList(strInp: string): string;

Метод возвращает список эмоций по программе лояльности

Описание полей в выходных данных

- **EmotionID** - Идентификатор эмоции
- **EmotionName** - Наименование эмоции
- **Points** - Стоимость эмоции в баллах
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>3</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row EmotionID="1" EmotionName="Покупка программки"
Points="20" result_code="0" result_message="OK"/>
    <Row EmotionID="2" EmotionName="Поздравление с днем
рождения" Points="150" result_code="0" result_message="OK"/>
```

```
<Row EmotionID="3" EmotionName="Фото с любимым
артистом" Points="100" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.8.4. СПИСОК ВИДОВ СКИДОК (СКИДОЧНЫХ КАРТ)

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

Метод возвращает список видов скидок (скидочных карт)

Описание полей в выходных данных

- BenefitId ID вида скидки (скидочной карты)
- BenefitName Наименование вида скидки (скидочной карты)

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>2</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row BenefitName="Единая Карта Петербуржца" BenefitId="24"
result_code="0" result_message="OK"/>
    <Row BenefitName="Скидочная карта Колокольчик" Bene-
fitId="29" result_code="0" result_message="OK"/>
</AnswerBody>
```

```
</GateAnswer>
```

17.8.5. СХЕМА СКИДОК ПО АБОНЕМЕНТНОЙ КАМПАНИИ

Function GetSTCompaniesPricesList(strInp: string): string;

Метод возвращает возвращает схему скидок по абонементной кампании (кампаниям)

Описание входных параметров.

- **[STCompanyId]** - Идентификатор аботментной кампании*
- **[NomBilKn]** - Идентификатор события**

*Если параметр опущен, будет возвращены данные по всем активным абонементным кампаниям.

**Если параметр присутствует, будет возвращены данные по абонементной кампании, связанной с конкретным событием.

Описание полей в выходных данных

- **STCompanyId** - Идентификатор аботментной кампании
- **NomBilKn** - Идентификатор события (0 если одноименный входной параметр опущен)
- **QtyFrom** - Минимальное кол-во билетов для получения соответствующей скидки
- **QtyTo** - Максимальное кол-во билетов для получения соответствующей скидки
- **DiscountPercent** - Процент скидки
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
```

```
<UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
  [<InputRow [STCompanyId="2"] />]
  [<InputRow [STCompanyId="6"] />] |
  [<InputRow [NomBilKn="26"] />]
  [<InputRow [NomBilKn="48"] />]
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>6</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row STCompanyId="2" NomBilKn="0" FromQty="2" ToQty="4"
DiscountPercent="20.00" result_code="0" result_message="OK"/>
  <Row STCompanyId="2" NomBilKn="0" FromQty="5" ToQty="6"
DiscountPercent="30.00" result_code="0" result_message="OK"/>
  <Row STCompanyId="2" NomBilKn="0" FromQty="7"
ToQty="1000" DiscountPercent="35.00" result_code="0" re-
sult_message="OK"/>
  <Row STCompanyId="6" NomBilKn="0" FromQty="3" ToQty="4"
DiscountPercent="7.00" result_code="0" result_message="OK"/>
  <Row STCompanyId="6" NomBilKn="0" FromQty="5" ToQty="6"
DiscountPercent="15.00" result_code="0" result_message="OK"/>
  <Row STCompanyId="6" NomBilKn="0" FromQty="7" ToQty="10"
DiscountPercent="25.00" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.9. КЛАДР

- [Список регионов](#)
- [Список городов](#)

17.9.1. СПИСОК РЕГИОНОВ

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

Метод возвращает список регионов

Описание полей в выходных данных

- **RegionID** - Идентификатор региона (строка)
- **RegionName** - Наименование региона
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row RegionID="7700000000000" RegionName="Москва" re-
sult_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.9.2. СПИСОК ГОРОДОВ

Function GetRegionsCitiesList(strInp: string): string;

Метод возвращает список городов по регионам

Описание входных параметров.

- **[RegionID]** - Идентификатор региона (строка)

Описание полей в выходных данных

- **RegionID** - Идентификатор региона (строка)

- **CityID** - Идентификатор города (строка)

- **RegionName** - Наименование города

- **result_code** - Код ошибки

- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow RegionID="7700000000000" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>7</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row RegionID="7700000000000" CityID="7700000000000"
CityName="Москва result_code="0" result_message="OK"/>
    <Row RegionID="7700000000000" CityID="7701100200000"
CityName="Город Московский" result_code="0" re-
sult_message="OK"/>
```

```
<Row RegionID="770000000000" CityID="7701200000100"
CityName="Поселок завода Мосрентген" result_code="0" re-
sult_message="OK"/>
<Row RegionID="770000000000" CityID="7701200000000"
CityName="Поселок Мосрентген" result_code="0" re-
sult_message="OK"/>
<Row RegionID="770000000000" CityID="7700100001300"
CityName="дск Московский Писатель" result_code="0" re-
sult_message="OK"/>
<Row RegionID="770000000000" CityID="7701600000800"
CityName="деревня Мостовское" result_code="0" re-
sult_message="OK"/>
<Row RegionID="770000000000" CityID="7701100000000"
CityName="Поселок Московский" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.10. ДОСТАВКА

- [Список станций метро](#)
- [Список офисов](#)
- [Список услуг доставки](#)

17.10.1. СПИСОК СТАНЦИЙ МЕТРО

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

Метод возвращает список записей о станция метро и линиях

Описание полей в выходных данных

- **ID** - Идентификатор станции,
- **Name** - Наименование станции,
- **LineName** - Наименование линии (ветки) метро, к которой относится станция
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>3</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row ID="25" Name="Парк Культуры" LineName = "Сокольническая"
result_code="0" result_message="OK"/>
    <Row ID="89" Name="Парк Культуры" LineName = "Кольцевая" re-
sult_code="0" result_message="OK"/>
    <Row ID="36" Name="Кантемировская" LineName =
"Замоскворецкая" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.10.2. СПИСОК ОФИСОВ

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

Метод возвращает список офисов и касс.

Описание полей в выходных данных

- **ID** - Идентификатор офиса,
- **Name** - Наименование офиса
- **result_code** - Код ошибки

- result_message - Описание ошибки**Входной XML.**

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>4</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row ID="1" Name="Центральный офис" result_code="0" re-
sult_message="OK"/>
    <Row ID="2" Name="Доп. офис" result_code="0" re-
sult_message="OK"/>
    <Row ID="3" Name="Касса №1 (Отрадное)" result_code="0" re-
sult_message="OK"/>
    <Row ID="4" Name="Касса №2 (Марьино)" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

17.10.3. СПИСОК УСЛУГ ДОСТАВКИ

Function GetDeliveriesList(strInp: string): string;

Метод возвращает список записей об услугах доставки и их стоимости.

Описание полей в выходных данных

- **ID** - Идентификатор услуги
- **Name** - Наименование услуги
- **Price** - Цена
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>2</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row ID="1" Name="Доставка в пределах МКАД" Price="100" re-
sult_code="0" result_message="OK"/>
    <Row ID="2" Name="Доставка по Московской области" Price="300"
result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

18. БРОНИРОВАНИЕ

- [Комплекты](#)

- [Предварительное резервирование мест комплекта](#)
- [Снятие предварительного резервирования мест комплекта](#)
- [Бронирование мест комплекта на событие](#)
- [Снятие бронирования мест комплекта](#)
- [Предварительное резервирование мест](#)
- [Снятие предварительного резервирования мест](#)
- [Бронирование мест на событие](#)
- [Снятие бронирования мест](#)
- [Информация о заказе/местах](#)
 - [Получить статус заказа](#)
 - [Получить статус заказа 2](#)
 - [Информация о заказе по его номеру](#)
 - [Получить статус мест сессии](#)
 - [Информация о текущем состоянии места заказа](#)
- [Проверка](#)
 - [Проверка промокодов](#)
 - [Проверка скидочной карты](#)
- [Холдирование](#)
 - [Удержание мест за плательщиком](#)
 - [Снятие удержания мест](#)

18.1. КОМПЛЕКТЫ

- [Предварительное резервирование мест комплекта](#)
- [Снятие предварительного резервирования мест комплекта](#)
- [Бронирование мест комплекта на событие](#)
- [Снятие бронирования мест комплекта](#)

18.1.1. 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 ПО «ПрофТикет» 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** - Идентификатор комплекта
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии
- **[RemoteAddress]** - IP адрес зрителя

Описание полей в выходных данных

- **NomBilKn** Идентификатор события.
- **EventsSetID** - Идентификатор комплекта
- **cod_sec** Идентификатор сектора.
- **name_sec** Наименование сектора.
- **row** ряд.
- **seat** место.

- **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 идентификатор комплекта, идентификатор сектора, ряд и место.

В случае удачного осуществления операции возвращается полный параметров.

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow EventsSetID="34" cod_sec ="2" row="1" seat="3"
session="SDSU42343DS343433"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
```

```
всего запроса-->
    <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="256" EventsSetID="34" cod_sec ="2" row="1"
seat="3" session="SDSU42343DS343433" price="1000"
priceSell="1000" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

18.1.2. 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** - Идентификатор комплекта
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии
- **[RemoteAddress]** - IP адрес зрителя

Описание полей в выходных данных

- **NomBilKn** Идентификатор события.
- **EventsSetID** - Идентификатор комплекта
- **cod_sec** Идентификатор сектора.
- **name_sec** Наименование сектора.
- **row** ряд.
- **seat** место.

- **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

Обратите внимание!

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

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

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
```

```
<AnswerBody>
  <Row result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

18.1.3. 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** - Идентификатор комплекта
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии
- **NameSpektator** - ФИО зрителя
- **TelSpektator** - Телефон зрителя
- **EmailSpektator** - Email зрителя
- **[SendAdvertising]** - Согласие на рассылку (рекламу). 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. 4 Цифры. 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) (обязательное поле при наличии доставки)
- **[DeliveryDate]** - Дата доставки (обязательное поле при наличии доставки)
- **[Address]** - Адрес доставки (обязательное поле при наличии доставки)

- **[MetroID]** - Идентификатор станции метро.
- **[OfficeID]** - Идентификатор офиса. Указывается при самовывозе (т.е. отсутствии доставки). Это предполагаемый офис выкупа билетов. Если забронирован один или несколько "живых" билетов, то указать идентификатор офиса обязательно.
- **[Notes]** - Комментарий
- **[OrderID]** - Идентификатор заказа в БД сайта или платежной системе
- **[DeliveryService]** - Идентификатор услуги доставки (обязательное поле при наличии доставки)
- **[RemoteAddress]** - IP адрес зрителя

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события.
- **cod_sec** Идентификатор сектора.
- **name_sec** Наименование сектора
- **row** ряд (string[4]).
- **seat** место (string[4]).
- **price** - Цена билета. (номинал)*
- **priceSell** - Цена билета (продажи)*
- **IDSpectator** - Идентификатор зрителя.
- **NameSpektator** ФИО зрителя
- **TelSpektator** Телефон зрителя
- **EmailSpektator** Email зрителя
- **Delivery** Наличие/отсутствие доставки (1 есть доставка, 0 - нет доставки)
- **DeliveryDate** - Предполагаемая дата доставки
- **Address** - Адрес доставки
- **MetroID** - Идентификатор станции метро.
- **Metro** - Наименование станции метро.

- **OfficeID** - Идентификатор офиса.
- **OfficeName** - Наименование офиса.
- **Notes** - Комментарий
- **OrderID** - Идентификатор заказа в БД сайта или платежной системе
- **DeliveryService** - Идентификатор услуги доставки
- **session** - Идентификатор сессии
- **ReservID** - номер заказа (в случае ошибки равен 0)
- **Barcode** - штрих-код места
- **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 заблокирован для любых действий с ним, кроме как через шлюз)
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

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 идентификатор комплекта, идентификатор сектора, ряд и место.

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

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow EventsSetID="34" cod_sec="2" row="1" seat="3"
session="SDSU42343DS343433" NameSpektator="Иванов Иван" Tel-
Spektator="+7(111)111-1144" EmailSpektator="ivanov@mail.ru" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>3</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="256" EventsSetID="34" cod_sec="2" row="1"
seat="3" session="SDSU42343DS343433" price="1000"
priceSell="1000" NameSpektator="Иванов Иван" TelSpekta-
tor="+7(111)111-1144" EmailSpektator="ivanov@mail.ru" Reserv-
ID="2575" ReservDate="16.06.2025"
ReservTime="19:00"result_code="0" result_message="OK"/>
    <Row NomBilKn="257" EventsSetID="34" cod_sec="2" row="1"
seat="3" session="SDSU42343DS343433" price="1000"
priceSell="1000" NameSpektator="Иванов Иван" TelSpekta-
tor="+7(111)111-1144" EmailSpektator="ivanov@mail.ru" Reserv-
ID="2575" ReservDate="16.06.2025"
ReservTime="19:00"result_code="0" result_message="OK"/>
    <Row NomBilKn="258" EventsSetID="34" cod_sec="2" row="1"
seat="3" session="SDSU42343DS343433" price="1000"
priceSell="1000" NameSpektator="Иванов Иван" TelSpekta-
tor="+7(111)111-1144" EmailSpektator="ivanov@mail.ru" Reserv-
ID="2575" ReservDate="16.06.2025"
ReservTime="19:00"result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

18.1.4. 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** - Идентификатор комплекта
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии
- **[RemoteAddress]** - IP адрес зрителя

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора
- **row** ряд
- **seat** место
- **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 идентификатор комплекта, идентификатор сектора, ряд и место.

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

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow EventsSetID="34" cod_sec="2" row="1" seat="3"
session="SDSU42343DS343433" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="256" EventsSetID="34" cod_sec="2" row="1"
seat="3" session="SDSU42343DS343433" price="1000"
priceSell="1000" ReservID="2575" ReservDate="16.06.2025"
ReservTime="19:00" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

18.2. ПРЕДВАРИТЕЛЬНОЕ РЕЗЕРВИРОВАНИЕ МЕСТ

Function SetPreReservation(strInp: string): string;

Метод осуществляет резервирование указанных мест в БД таким образом, чтобы эти места на заданное время (глобальный параметр системы, от 3 до 30 мин.) стали недоступны для резервирования или бронирования с рабочего места ПО «ПрофТикет» и вызовом 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. Метод можно вызывать сколько угодно раз в рамках одной сессии, пока места не будут забронированы и оплачены.

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии
- **[RemoteAddress]** - IP адрес зрителя
- **[PromoCodeID]** - Идентификатор промокода
- **[Price]** - Цена билета
- **[DisableDiscount]** - Флаг не использовать скидку абонементной кампании. 1 - не использовать скидку, 0 или не указано использовать скидку при ее наличии.
- **[PushkinCard]**- Признак оплаты по Пушкинской карте.
- **[EmailSpektator]** - Email зрителя для проверки промокода, привязанного к Email.
- **[CardNumber]** - Номер или штрих-код дисконтной карты.

[URL площадки продажи билетов (необязательный раздел)]

- **SourceURL** - URL площадки продажи билетов

[Подарочные сертификаты]

- **Number** Номер сертификата

[Скидки]

- **BenefitId** - Идентификатор вида скидочной карты
- **BenefitCardId** - Номер скидочной карты

Описание полей в выходных данных

- **cod_sec** Идентификатор сектора.
- **name_sec** Наименование сектора.
- **row** ряд
- **seat** место
- **price** Цена билета (номинал).*
- **priceSell** Цена билета (продажи)* В случае, если указан промокод, цена продажи учитывает скидку по промокоду.
- **priceForPay** - Сумма к оплате. Если сертификаты не используются, то равна цене продажи билета + сервисный сбор.
- **ServiceFee** - Сервисный сбор.
- **priceSellFull** - Сумма с учетом первоначальных наценок и скидок, без сервисного сбора, промокода и абонементной кампании,
- **priceReturn** - Максимальная сумма возврата
- **CertificatePermitted** - Место можно оплачивать сертификатом.
- **TTL** - Время в секундах, после которого места, находящиеся в предварительном резервировании могут быть сняты автоматически, если они не попали в заказ.
- **PromoCodeID** - Идентификатор промокода (если не указан во входных параметрах, то 0)
- **PromoPercent** - Процент по промокоду (если указан Идентификатор промокода и промокод актуален, в противном случае - 0)
- **STCompanyDiscount** - Процент скидки по абонементной кампании.
- **DisableDiscount** - Флаг не использовать скидку абонементной кампании

- **DiscountProgramPercent** - Процент скидки по дисконтной программе
- **PushkinCard** - Признак оплаты по Пушкинской карте.
- **EmailSpectator** - Email зрителя для проверки промокода, привязанного к Email.
- **TicketAsGift** - Возможность купить билет в подарок.
- **StopRegistrationDate** - Дата и время, до которых разрешено оформление ПДн
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

[Подарочные сертификаты]

- **Number** Номер (штрих-код) сертификата
- **Amount** Номинал
- **StartDate** Дата начала действия сертификата.
- **StopDate** Дата окончания действия сертификата.

[Скидки]

- **BenefitId** - Идентификатор вида скидочной карты
- **BenefitCardId** - Номер скидочной карты
- **BenefitDiscountPercent** - Процент по скидочной карте
- **BenefitDiscountSum** - Сумма скидки по скидочной карте (в случае скидки в твердой сумме)

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!

Для создания заказа с промокодом, его необходимо указать в вызове

метода бронирования SetReservation!

Для создания продажи с промокодом, его необходимо указать в вызове метода продажи SetSold.

Если указан входящий параметр Price (цена), будет произведена проверка цены. В случае несовпадения цены у любого из мест в передаваемом наборе

Результатом будет код ошибки 20, а также поле GatePrice, возвращающее переданную цену.

Структуру данных при передаче списка сертификатов для оплаты!

Структуру данных при передаче URL площадки продажи билетов!

Структуру данных при передаче номера скидочной карты!

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="234" cod_sec="1" row="1" seat="2"
session="KFKDI23202828SBW4DAH8"/>
    [<Certificates Number="1234567890" Amount="500.00"
StartDate="01.01.2025" StopDate="31.12.2025">
    <Certificates Number="9876543210" Amount="500.00"
StartDate="01.01.2025" StopDate="31.12.2025">
    ]
    [<Discount BenefitId="24" BenefitCard-
Number="01234567890" />]
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
```

```
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="234" cod_sec="1" row="1" seat="2" ses-
sion="KFKDI23202828SBW4DAH8" price="1000" priceSell="900"
priceForPay="900" priceFull="1000" result_code="0" re-
sult_message="OK"/>

  <Certificates Number="1234567890" Amount="500.00"
StartDate="01.08.2020" StopDate="31.07.2021">
  <Certificates Number="9876543210" Amount="500.00"
StartDate="01.08.2020" StopDate="31.07.2021">
  <Discount BenefitId="24" BenefitCardNumber="01234567890" Dis-
countPercent="10" DiscountSum="0.00" />
</AnswerBody>
</GateAnswer>
```

18.3. 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** - Идентификатор события
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии
- **[RemoteAddress]** - IP адрес зрителя

Описание полей в выходных данных

- **cod_sec** ID сектора.
- **row** ряд.

- **seat** место.
- **price** Цена билета (номинал).*
- **priceSell** Цена билета (продажи)*
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="234" cod_sec="1" row="1" seat="2"
session="KFKDI23202828SBW4DAH8"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="234" cod_sec ="1" row ="1" seat ="2"
price="1000" priceSell="1000" result_code="0" re-
sult_message=""OK/>
</AnswerBody>
</GateAnswer>
```

18.4. 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** - Идентификатор события
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии
- **NameSpektator** - ФИО зрителя
- **TelSpektator** - Телефон зрителя
- **EmailSpektator** - Email зрителя
- **[CityName]** - город зрителя
- **[Gender]** - пол зрителя (необязательное поле. 0 - не указан, 1 - женский пол, 2 мужской пол, 3 - неопределен)
- **[BirthDate]** - Дата рождения в формате дд.мм.гггг
- **[SendAdvertising]** - Согласие на рассылку (рекламу).. 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. 4 цифры. 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]** - Идентификатор станции метро.
- **[OfficeID]** - Идентификатор офиса. Указывается при самовывозе (т.е. отсутствии доставки). Это предполагаемый офис выкупа билетов. Если забронирован один или несколько "живых" билетов, то указать Идентификатор офиса обязательно.
- **[Notes]** - Комментарий
- **[OrderID]** - Идентификатор заказа в БД сайта или платежной системе
- **[DeliveryService]** - Идентификатор услуги доставки (необязательное поле при отсутствии доставки)
- **[RemoteAddress]** - IP адрес зрителя
- **[PromoCodeID]** - Идентификатор промокода
- **[isPaidOrder]** - Признак заведомо оплачиваемого заказа.
- **[DisableDiscount]** - Флаг не использовать скидку абонементной кампании. 1 - не использовать скидку, 0 или не указано использовать скидку при ее наличии.
- **[PushkinCard]** - Признак оплаты по Пушкинской карте.
- **[withQRCode]** - Флаг - покупатель подтверждает наличие QR кода о вакцинации*.
- **[GroupOrder]** - Признак группового заказа.. 0 - не групповой заказ (по-умолчанию). 1 - групповой заказ
- **[MultiPayGroup]** - Для группового заказа возможность оплатить несколько билетов одним плательщиком.. 0 - нельзя одному плательщику выкупить несколько мест (по-умолчанию). 1 - можно.
- **[CardNumber]** - Номер или штрих-код дисконтной карты.
- **[VisitorDataDelayedInput]** - Флаг - данные посетителя будут переданы после покупки.

[Подарочные сертификаты]

- **Number** Номер сертификата

[Скидки]

- **BenefitИдентификатор** - идентификатор вида скидочной карты

- **BenefitCardИдентификатор** - Номер скидочной карты

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события.
- **cod_sec** Идентификатор сектора.
- **name_sec** Наименование сектора
- **row** ряд.
- **seat** место.
- **price** - Цена билета. (номинал)*
- **priceSell** - Цена билета (продажи)* с учетом введенного промокода и прочих возможных скидков
- **priceForPay** - Сумма к оплате. Если сертификаты не используются, то равна цене продажи билета + сервисный сбор.
- **ServiceFee** - сервисный сбор.
- **priceSellFull** - сумма с учетом первоначальных наценок и скидков, без сервисного сбора, промокода и абонементной кампании,
- **priceReturn** - максимальная сумма возврата
- **CertificatePermitted** - Место можно оплачивать сертификатом.

- **IDSpectator** Идентификатор зрителя.
- **NameSpektator** ФИО зрителя
- **TelSpektator** Телефон зрителя
- **EmailSpektator** Email зрителя
- **CityName** - Город зрителя,
- **Gender** - Пол зрителя (0 - не указан, 1 - женский пол, 2 мужской пол, 3 - неопределен)
- **BirthDate** - Дата рождения

- **Delivery** Наличие/отсутствие доставки (1 есть доставка, 0 - нет доставки)
- **DeliveryDate** - Предполагаемая дата доставки
- **Address** - Адрес доставки
- **MetroID** - Идентификатор станции метро.
- **Metro** - Наименование станции метро.
- **OfficeID** - Идентификатор офиса.
- **OfficeName** - Наименование офиса.
- **Notes** - Комментарий
- **OrderID** - Идентификатор заказа в БД сайта или платежной системе
- **DeliveryService** - Идентификатор услуги доставки
- **session** - Идентификатор сессии
- **ReservID** - номер заказа (в случае ошибки равен 0)
- **Barcode** - штрих-код места
- **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 заблокирован для любых действий с ним, кроме как через шлюз)
- **PromoCodeID** - Идентификатор промокода (если не указан во входных параметрах, то 0)
- **PromoPercent** - Процент по промокоду (если указан Идентификатор промокода и промокод актуален, в противном случае - 0)
- **STCompanyDiscount** - Процент скидки по абонементной кампании.
- **SupplierID** - Идентификатор поставщика (Только для собственного сайта. Для стороннего сайта - поле пустое).
- **PushkinCard** - Признак оплаты по Пушкинской карте
- **withQRCode** - Флаг - покупатель подтверждает наличие QR кода о вакцинации,
- **DiscountProgramPercent** - Процент скидки по дисконтной программе
- **VisitorDataDelayedInput** - Флаг - данные посетителя будут переданы после

покупки.

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

ReservDate и ReservTime - дата и время "выкупа" заказа. По истечении 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.

[Подарочные сертификаты]

- **Number** Номер (штрих-код) сертификата
- **Amount** Номинал
- **StartDate** Дата начала действия сертификата.
- **StopDate** Дата окончания действия сертификата.

[Скидки]

- **BenefitId** - идентификатор вида скидочной карты
- **BenefitCardId** - Номер скидочной карты
- **BenefitDiscountPercent** - Процент по скидочной карте
- **BenefitDiscountSum** - Сумма скидки по скидочной карте (в случае скидки в твердой сумме)

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!

Признак заведомо оплачиваемого заказа устанавливается для того,

чтобы отличить заказы которые могут быть проданы в кассе площадки, от тех которые будут оплачиваться электронно. У таких заказов дата выкупа заказа может быть меньше простых заказов и кроме разницы в дате выкупа заказы не отличаются.

По-умолчанию заказ не имеет признак заведомо оплачиваемого.

Если к событиям, входящим в заказ, может быть применена скидка по абонементной компании, она будет рассчитана исходя из кол-ва мест, находящихся в заказе в текущий момент. При этом сумма скидки может измениться в последующем, если в заказ будут добавлены или удалены места.

Если событие имеет флаг QRCodeOnly - на событие допускаются только зрители с наличием QR кода о вакцинации, то поле withQRCode обязательно и должно быть равным 1.

Обратите внимание!

на структуру данных при передаче списка сертификатов для оплаты!

на структуру передаваемых данных по сертификатам и скидкам!

Входной XML.

```
<GateReq>
<ReqLogin>
  <!-- Имя пользователя и пароль для авторизации -->
  <UserName>User</UserName>
  <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
  <InputRow NomBilKn="125" cod_sec="1" Row="1" Seat="1"
NameSpektator="Иванов Иван" TelSpektator="+7(111)111-1144"
EmailSpektator="ivanov@mail.ru" session="JF83325NKEDFBSAL829"/>

  [<Certificates Number="124396531808071091"/>

  <Certificates Number="113058751407531493"/>

  <Discount BenefitId="24" BenefitCardNumber="01234567890" />]
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="125" cod_sec="1" row="1" seat="1" price="1000"
priceSell="1000" ReservID="2565" ReservDate="15.06.2025"
ReservTime="19:00" NameSpektator="Иванов Иван" TelSpekta-
tor="+7(111)111-1144" EmailSpektator="ivanov@mail.ru" re-
sult_code="0" result_message="OK"/>

  [
    <Certificates Number="1234567890" Amount="500.00"
StartDate="01.01.2025" StopDate="31.12.2025"/>
    <Certificates Number="9876543210" Amount="500.00"
StartDate="01.01.2025" StopDate="31.12.2025"/>

    <Discount BenefitId="24" BenefitCardNumber="01234567890" Dis-
countPercent="5" DiscountSum="0.00" />
  ]
</AnswerBody>
</GateAnswer>
```

18.5. СНЯТИЕ БРОНИРОВАНИЯ МЕСТ

Function FreeReservation(strInp: string): string;

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

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место

- **session** - Идентификатор сессии
- **[RemoteAddress]** - IP адрес зрителя

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора
- **row** ряд.
- **seat** место.
- **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)
- **ReservDate** - Дата заказа
- **ReservTime** - Время заказа
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="125" cod_sec="1" Row="1" Seat="1"
session="JF83325NKEDFBSAL829"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
```

```
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="125" cod_sec="1" row="1" seat="1" ses-
sion="JF83325NKEDFB829" price="1000" priceSell="1000" Reserv-
ID="2565" ReservDate="15.06.2025" ReservTime="19:00" re-
sult_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

18.6. ИНФОРМАЦИЯ О ЗАКАЗЕ/МЕСТАХ

- [Получить статус заказа](#)
- [Получить статус заказа 2](#)
- [Информация о заказе по его номеру](#)
- [Получить статус мест сессии](#)
- [Информация о текущем состоянии места заказа](#)

18.6.1. ПОЛУЧИТЬ СТАТУС ЗАКАЗА

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

Метод возвращает список текущий статус заказа.

[Список статусов заказа](#)

Описание входных параметров.

- **ReservID** - Номер заказа

Описание полей в выходных данных

- **ReservStatus** - Идентификатор [статуса заказа](#)
- **ReservStatusDescription** - Описание статуса заказа
- **Qty** - Кол-во мест заказа
- **Amount** - Сумма заказа

- **ReservDate** - Дата аннулирования заказа
- **ReservTime** - Время аннулирования заказа
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow ReservID="1805"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row ReservID="1805" ReservStatus="5" ReservStatusDescrip-
tion="Оплачен по б/н. Билеты не переданы клиенту." Qty="2"
Amount="4000.00" ReservDate="22.11.2025" ReservTime="19:00:00"
result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

18.6.2. ПОЛУЧИТЬ СТАТУС ЗАКАЗА 2

Function **GetReservationStatus2**(strInp: string): string;

Метод возвращает список текущий статус заказа.

Описание входных параметров.

- **ReservID** - Номер заказа

Описание полей в выходных данных

- **ReservID** - Номер заказа

- **ReservStatus** - Идентификатор [статуса заказа](#)

- **ReservStatusDescription** - Описание статуса заказа

- **Qty** - Кол-во мест заказа

- **Amount** - Сумма заказа

- **CanceledByGate** - Признак снятия заказа (1 - заказ снят пользователем шлюза, 0 - заказ снят не пользователем шлюза, -1 - заказ не снят или иное)

- **result_code** - Код ошибки

- **result_message** - Описание ошибки

Статусы заказов:

"**RES**" - Заказ создан (даже если часть место снята)

"**CRS**" - Заказ снят полностью

"**CRA**" - Заказ снят автоматически

"**SOL**" - Заказ продан (даже если часть мест возвращена)

"**RET**" - Все места заказа возвращены

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow ReservID="1805"/>
</ReqBody>
```

```
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row ReservID="1805" ReservStatus="SOL" Qty="2"
Amount="3600.00" ReserveDate="22.11.2025" ReserveTime="19:00:00"
CanceledByGate="-1"/>
</AnswerBody>
</GateAnswer>
```

18.6.3. ИНФОРМАЦИЯ О ЗАКАЗЕ ПО ЕГО НОМЕРУ

Function GetReservationData(strInp: string): string;

Возвращает данные по заказу. Необходима для продажи на сайте по номеру заказа. Используется только для собственного сайта. Для сторонних агентов метод закрыт.

Описание входных параметров.

- **ReservID** - Номер заказа
- **TelSpektator** - Телефон зрителя

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события.
- **ReservID** - Номер заказа
- **session** Идентификатор сессии
- **cod_sec** Идентификатор сектора.
- **name_sec** Наименование сектора

- **FreeOfferSeat** Признак свободной рассадки
- **PrintRows** - Печатать номер ряда на билете (1 - да, 0 - нет)
- **VisibleRows** - Отображать номер ряда на схеме зала (1 - да, 0 - нет)
- **PrintSeats** - Печатать номер места на билете (1 - да, 0 - нет)
- **VisibleSeats** - Показывать номер места на схеме (1 - да, 0 - нет)
- **row** ряд
- **seat** место
- **price** - Цена билета. (номинал)
- **priceSell** - Цена билета (продажи) с учетом введенного промокода и прочих возможных скидков
- **ServiceFee** - сервисный сбор.
- **isBenefit** - Признак льготного билета
- **NameSpektator** ФИО зрителя
- **TelSpektator** Телефон зрителя
- **EmailSpektator** Email зрителя
- **Barcode** - штрих-код места (на некоторые события штрихкод возвращается только после продажи билета)
- **ReservDate** - дата создания заказа
- **ReservTime** - время создания заказа
- **CancelDate** - дата снятия заказа
- **CancelTime** - время снятия заказа

- **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
- **isService** - Событие является услугой.
- **cod_show** - Идентификатор мероприятия
- **cod_t** - Идентификатор места проведения (театра)
- **cod_h** - Идентификатор зала места проведения (сцены)
- **name_t** - Наименование места проведения (театра)
- **name_h** - Наименование 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** - Идентификатор организатора мероприятия,

- **OrganizerName** - Наименование организатора мероприятия,
- **OrganizerTin** - ИНН организатора мероприятия,
- **OrganizerPhone** - Телефон организатора мероприятия,
- **OrganizerEmail** - Email организатора мероприятия,
- **OrganizerAddress** - Адрес организатора мероприятия,
- **Status** Статус места заказа. 0 - Место недоступно для продажи. 1 - Место доступно для продажи, 2 - Место оплачено. 3 - Место в заказе и удержано за плательщиком
- **SupplierID** - Идентификатор поставщика.
- **PushkinCard** - Признак оплаты по Пушкинской карте.
- **ProCultureID** - Идентификатор мероприятия в ПроКультура
- **PushkinCardPermitted** - Разрешение продажи по программе "Пушкинская карта". (0 - не разрешено, 1 - разрешено, 2 - только по "Пушкинской карте")
- **QRCodeOnly** - Признак - доступ на событие разрешен только при предъявлении QR кода о вакцинации.
- **RequestVisitorData** - Запрашивать данные зрителя (0 - не запрашивать, 1 - запрашивать только ФИО, 2 - запрашивать ФИО + тип, серию и номер документа).
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow ReservID="1805" TelSpektator="+71234567890"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>2</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row ReservID="1805" ses-
sion="5dd3ac5626be9aac3fb39eb84d02b5b9" NomBilKn="211" Event-
Date="22.11.2025" EventTime="19:00" BeginDate="22.11.2025"
BeginTime="19:00" withOpenDate="0" cod_show="79" cod_t="1"
cod_h="1" EventDuration="01:00" is_Primer="1" WithIntermis-
sion="1" author="" Note1="" Note2="" Note3="" Note4="" Produc-
er="" OrganizerID="1" isService="0" OrganizerName="ФГУП
Академический театр" OrganizerTin="7711111111" Organiz-
erPhone="84952150000" OrganizerEmail="organizator@yandex.ru" Or-
ganizerAddress="121108, Улица Кое-кого, 4" name_show="Ревизор"
Age="0" name_t="Академический театр" name_h="Основная сцена" Ac-
tors="" Annotation="" cod_sec="1" name_sec="Партр" FreeOffer-
Seat="0" NotShowNumbers="0" PrintRows="1" VisibleRows="1"
PrintSeats="1" VisibleSeats="1" row="9" seat="1" price="2000.00"
priceSell="1800.00" priceSellFull="2000.00" ServiceFee="0.00"
priceForPay="1800.00" priceReturn="1800.00" NameSpektator="Петр
Петров" TelSpektator="+7 (122) 111-11-11" EmailSpekta-
tor="petrov@yandex.ru" Barcode="522820870453412294" Supplie-
rID="1" ReservDate="06.08.2025" ReservTime="14:17" Can-
celDate="22.11.2025" CancelTime="19:00" Status="2" PromoCode=""
PromoPercent="0" PromoSum="0.00" isBenefit="0" STCompanyDis-
count="10.00" GroupOrder="0" MultiPayGroup="0" ProCultureID=""
PushkinCardPermitted="0" PayerEmail="" RequestVisitorData="0"
result_code="0" result_message="OK"/>
  <Row ReservID="1805" ses-
sion="5dd3ac5626be9aac3fb39eb84d02b5b9" NomBilKn="211" Event-
Date="22.11.2025" EventTime="19:00" BeginDate="22.11.2025"
BeginTime="19:00" withOpenDate="0" cod_show="79" cod_t="1"
cod_h="1" EventDuration="01:00" is_Primer="1" WithIntermis-
sion="1" author="" Note1="" Note2="" Note3="" Note4="" Produc-
er="" OrganizerID="1" isService="0" OrganizerName="ФГУП
Академический театр" OrganizerTin="7711111111" Organiz-
erPhone="84952150000" OrganizerEmail="organizator@yandex.ru" Or-
ganizerAddress="121108, Улица Кое-кого, 4" name_show="Ревизор"
Age="0" name_t="Академический театр" name_h="Основная сцена" Ac-
tors="" Annotation="" cod_sec="1" name_sec="Партр" FreeOffer-
Seat="0" NotShowNumbers="0" PrintRows="1" VisibleRows="1"
PrintSeats="1" VisibleSeats="1" row="9" seat="2" price="2000.00"
priceSell="1800.00" priceSellFull="2000.00" ServiceFee="0.00"
priceForPay="1800.00" priceReturn="1800.00" NameSpektator="Петр
Петров" TelSpektator="+7 (122) 111-11-11" EmailSpekta-
tor="petrov@yandex.ru" Barcode="522820870453412294" Supplie-
rID="1" ReservDate="06.08.2025" ReservTime="14:17" Can-
```

```
celDate="22.11.2025" CancelTime="19:00" Status="2" PromoCode=""  
PromoPercent="0" PromoSum="0.00" isBenefit="0" STCompanyDis-  
count="10.00" GroupOrder="0" MultiPayGroup="0" ProCultureID=""  
PushkinCardPermitted="0" PayerEmail="0" RequestVisitorData="0"  
result_code="0" result_message="OK"/>  
</AnswerBody>  
</GateAnswer>
```

18.6.4. ПОЛУЧИТЬ СТАТУС МЕСТ СЕССИИ

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

Возвращает данные о местах сессии: статус места, цены, размер скидки и т.д.

Описание входных параметров.

- **session** - Идентификатор сессии

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события.
- **ReservID** - Номер заказа
- **session** Идентификатор сессии
- **cod_sec** Идентификатор сектора.
- **name_sec** Наименование сектора
- **row** ряд
- **seat** место
- **price** - Цена билета. (номинал)
- **priceSell** - Цена билета (продажи) с учетом введенного промокода и прочих возможных скидок
- **ServiceFee** - Сервисный сбор
- **PlaceStatus** - Статус места*
- **PromoCodeID** - Идентификатор промокода (если не указан во входных параметрах, то 0)

- **PromoPercent** - Процент по промокоду (если указан Идентификатор промокода и промокод актуален, в противном случае - 0)
- **PromoSum** - Сумма скидки по промокоду (если указан Идентификатор промокода и промокод актуален, в противном случае - 0)
- **STCompanyID** - Идентификатор абонементной кампании
- **STCompanyDiscount** - размер скидки по абонементной кампании
- **DisableDiscount** - Флаг неиспользовать скидку абонементной кампании
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

***Статусы мест:**

SEL - место в предварительном резерве,

RES - место забронировано,

USO - место удалено из заказа,

SOL - место продано

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow session="5dd3ac5626be9aac3fb39eb84d02b5b9"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
```

```
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="211" cod_sec="1" Row="9" Seat="1"
name_sec="Партер" PlaceStatus="SOL" price="2000.00"
priceSell="1800.00" ServiceFree="0.00" ses-
sion="5dd3ac5626be9aac3fb39eb84d02b5b9" STCompanyID="1" STCompa-
nyDiscount="10.00" DisableDiscount="0" PromoPercent="0.00" Pro-
mocodeID="0" PromoSum="0.00" result_code="0" re-
sult_message="OK"/>
  <Row NomBilKn="211" cod_sec="1" Row="9" Seat="2"
name_sec="Партер" PlaceStatus="SOL" price="2000.00"
priceSell="1800.00" ServiceFree="0.00" ses-
sion="5dd3ac5626be9aac3fb39eb84d02b5b9" STCompanyID="1" STCompa-
nyDiscount="10.00" DisableDiscount="0" PromoPercent="0.00" Pro-
mocodeID="0" PromoSum="0.00" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

18.6.5. ИНФОРМАЦИЯ О ТЕКУЩЕМ СОСТОЯНИИ МЕСТА ЗАКАЗА

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** - Идентификатор события
- **cod_sec** - Идентификатор сектора
- **row** - ряд
- **seat** - место

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_sec** - Идентификатор 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, HLD - место в заказе и удержано за плательщиком 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),
- **Barcode** - штрих-код (только если заказ в статусе RES или SOL. Для остальных статусов возвращается пустая строка)
- **price** - Цена места (номинал).
- **priceSell** - Цена места (продажа)
- **STCompanyDiscount** - Процент скидки по абонементной кампании
- **StatusComment** - Описание статуса
- **CheckCode** - Код для проверки места заказа (при самостоятельной регистрации посетителя)*
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

** если на событие не нужно собирать данные зрителя - код передается пустой.*

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
```

```
<UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
  <InputRow ReservID="1805" NomBilKn="211" cod_sec="1"
row="9" seat="1"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="211" cod_sec="1" Name_sec="Партер" row="9"
seat="1" ReservID="1805" ReservDate="06.08.2025"
ReservTime="14:17:58" ChangeDate="06.08.2025"
ChangeTime="14:18:20" Blocked="0" PlaceStatus="SOL" GateUser="1"
StatusComment="Sold by Gate" Barcode="522820870453412294"
price="1800.00" priceSell="1800.00" STCompanyDiscount="0.00"
CheckCode="" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

18.7. ПРОВЕРКА

- [Проверка промокодов](#)
- [Проверка скидочной карты](#)

18.7.1. ПРОВЕРКА ПРОМОКОДОВ

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

Проверка валидности промокодов

Описание входных параметров.

- **NomBilKn** - Идентификатор события, к которому применяется промокод
- **cod_sec** - Идентификатор сектора

- **row** - Ряд
- **seat** - Место
- **PromoCode** - Промокод
- **[EmailSpektator]** - Email зрителя для проверки промокода, привязанного к Email.

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события, к которому применяется промокод
- **cod_sec** - Идентификатор сектора
- **row** - Ряд
- **seat** - Место
- **PromoCode** - Указанный промокод,
- **PromoCodeID** - Идентификатор промокода (если промокод невалидный - 0),
- **Percent** - Процент скидки по промокоду (0 если скидка по сумме),
- **PromoSum** - Сумма скидки по промокоду (0 если скидка в %),
- **price** - цена номинала
- **PriceSell** - цена продажи
- **EmailSpektator** - Email зрителя (если был указан, если нет - пустая строка).
- **isPermitted** - Разрешено ли применение данного промокода (1 - разрешено, 0 - нет),
- **result_code** код ошибки,
- **result_message** - описание ошибки

Обратите внимание!

Промокод может давать скидку как в процентном отношении так и фиксированную. Если поле Percent больше нуля, а Summa равна нулю, то скидка - в процентах, если наоборот, то фиксированная

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow PromoCode="BECHA2025" NomBilKn="17227"
cod_sec="1" Row="1" Seat="1" [EmailSpekta-
tor="ivanov@email.ru"]/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="17227" cod_sec="1" Row="1" Seat="1" Promo-
Code="BECHA2025" PromoCodeID="8" Percent="5" PromoSum="0.00"
price="1000" PriceSell="950" isPermitted="1" EmailSpekta-
tor="[ivanov@email.ru]" result_code="0" result_message="OK"/>

</AnswerBody>
</GateAnswer>
```

18.7.2. ПРОВЕРКА СКИДОЧНОЙ КАРТЫ

Function CheckDiscountCard(strInp: string): string;

Метод проверяет валидность сторонней скидочной карты и возвращает процент скидки.

Описание входных параметров.

- **BenefitID** Идентификатор вида скидочной карты.
- **BenefitCardNumber** Номер скидочной карты.

Описание полей в выходных данных

- **BenefitId** Идентификатор вида скидочной карты.
- **BenefitCardNumber** Номер скидочной карты.
- **DiscountPercent** Процент скидки (если скидка в процентах).
- **DiscountSum** Сумма скидки (если скидка в фиксированной сумме).
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow BenefitId="24" BenefitCard-
Number="01234567890"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row BenefitId="24" CardBenefitCardNumber="01234567890" Dis-
countPercent="5" DiscountSum="0.00" result_code="0" re-
sult_message="OK"/>

</AnswerBody>
</GateAnswer>
```

18.8. ХОЛДИРОВАНИЕ

- [Удержание мест за плательщиком](#)
- [Снятие удержания мест](#)

18.8.1. УДЕРЖАНИЕ МЕСТ ЗА ПЛАТЕЛЬЩИКОМ

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

Метод занимает указанные места за плательщиком

Метод необходим для удержания мест за плательщиком в групповом заказе. Также может быть использован и не в групповом заказе.

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии
- **ReservID** - Номер заказа
- **PayerName** - Имя плательщика
- **PayerPhone** - Телефон плательщика
- **PayerEmail** - Email плательщика

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии

- **ReservID** - Номер заказа
- **PayerName** - Имя плательщика
- **PayerPhone** - Телефон плательщика
- **PayerEmail** - Email плательщика
- **Holded** - Статус места (1 - занято, 0 - свободно)

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="5625" cod_sec="1" row="1" seat="1" reservID="95265" session="SDSU42343DS3SG433" PayerName="Иванов Иван Иванович" PayerPhone="+7 (123) 456-7890" PayerEmail="ivanov@email.ru" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в возвращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="5625" cod_sec="1" row="1" seat="1" reservID="95265" session="SDSU42343DS3SG433" PayerName="Иванов Иван Иванович" PayerPhone="+7 (123) 456-7890" PayerEmail="ivanov@email.ru" Helded="1" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

18.8.2. СНЯТИЕ УДЕРЖАНИЯ МЕСТ

```
Function UnHoldReservPlaces(strInp: string): string;
```

Метод занимает указанные места за плательщиком

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии
- **ReservID** - Номер заказа
- **PayerName** - Имя плательщика
- **PayerPhone** - Телефон плательщика
- **PayerEmail** - Email плательщика

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии
- **ReservID** - Номер заказа
- **PayerName** - Имя плательщика
- **PayerPhone** - Телефон плательщика
- **PayerEmail** - Email плательщика
- **Holded** - Статус места (1 - занято, 0 - свободно)

Входной XML.

```
<GateReq>
<ReqLogin>
  <!-- Имя пользователя и пароль для авторизации -->
  <UserName>User</UserName>
  <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
  <InputRow NomBilKn="5625" cod_sec="1" row="1" seat="1" reservID="95265" session="SDSU42343DS3SG433" PayerName="Иванов Иван Иванович" PayerPhone="+7 (123) 456-7890" PayerEmail="ivanov@email.ru" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в возвращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="5625" cod_sec="1" row="1" seat="1" reservID="95265" session="SDSU42343DS3SG433" PayerName="Иванов Иван Иванович" PayerPhone="+7 (123) 456-7890" PayerEmail="ivanov@email.ru" Holded="0" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

19. ПРОДАЖА И ВОЗВРАТ

- [Продажа мест](#)
- [Проверка мест перед продажей](#)
- [Продажа мест комплекта](#)
- [Возврат мест](#)
- [Проверка мест перед возвратом](#)
- [Дополнительные данные о продаже](#)

- [Дополнительные данные о возврате](#)
- [Проверка БИНа карты оплаты для получения скидки](#)

19.1. 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** - Идентификатор события
- **cod_sec** Идентификатор сектора
- **row** - ряд
- **seat** - место
- **session** - Идентификатор сессии
- **TransactionID** - Идентификатор транзакции
- **RRN** - RRN платежа
- **PaymentDate** - Дата оплаты
- **PaymentTime** - Время оплаты
- **[RemoteAddress]** - IP адрес зрителя (необязательное поле)
- **[PromoCodeID]** - Идентификатор промокода
- **[ExtraPay]** - Дополнительная оплата к сумме сертификатов
- **[BIN]** - Первые 6 цифр карты оплаты (Банковский идентификационный номер).
- **[TicketSeries]** - Серия электронного билета
- **[TicketNumber]** - Номер электронного билета
- **[DisableDiscount]** - Флаг не использовать скидку абонементной кампании. 1 - не использовать скидку, 0 или не указано использовать скидку при ее наличии.
- **[PushkinCard]** - Признак оплаты по Пушкинской карте (0 - обычная карта, 1 -

Пушкинская карта)

- **[PayerName]** - Имя плательщика (обязательно для групповых заказов)
- **[PayerPhone]** - Телефон плательщика (обязательно для групповых заказов)
- **[PayerEmail]** - Email плательщика (обязательно для групповых заказов)
- **[CardNumber]** - Номер или штрих-код дисконтной карты.
- **[DonatorPayment]** - Флаг - билет оплачен меценатом (0 - нет, 1 - да)
- **[PaymentMethodID]** - Идентификатор способа оплаты (см. список)

[Подарочные сертификаты]

- **Number** Номер (штрих-код) сертификата

[Скидки]

- **BenefitId** - Идентификатор вида скидочной карты
- **BenefitCardId** - Номер скидочной карты

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора
- **row** - ряд
- **seat** - место
- **price** - Цена билета (номинал).* (в случае ошибки равна 0)
- **priceSell** - Цена билета (продажи).* (в случае ошибки равна 0) В случае, если указан промокод или применяются иные скидки, цена продажи учитывает скидку по промокоду.
- **priceForPay** - Сумма к оплате. Если сертификаты не используются, то равна цене продажи билета + сервисный сбор.
- **ServiceFee** - сервисный сбор.

- **priceSellFull** - сумма с учетом первоначальных наценок и скидок, без сервисного сбора, промокода и абонементной кампании,
- **priceReturn** - максимальная сумма возврата
- **isBenefit** - Признак льготного билета
- **CertificatePermitted** - Место можно оплачивать сертификатом.
- **reservID** - номер заказа (в случае ошибки равен 0)
- **reservDate** - дата заказа (в случае ошибки пустая)
- **barcode** - штрих-код места
- **BarCodeType** - код типа штрих-кода,
- **TransactionID** - Идентификатор транзакции платежной системы.
- **PaymentDate** Дата оплаты
- **PaymentTime** Время оплаты
- **PromoCodeID** - Идентификатор промокода
- **PromoPercent** - Процент по промокоду (если указан идентификатор промокода и промокод актуален и скидка в %, в противном случае - 0)
- **PromoSum** - Сумма скидки по промокоду (если указан идентификатор промокода и промокод актуален и скидка не в %, в противном случае - 0)
- **STCompanyDiscount** - Процент скидки по абонементной кампании
- **BIN** - первые 6 цифр карты оплаты (Банковский идентификационный номер)
- **PaymentCardDiscount** - скидка по карте оплаты (если предоставляется скидка)
- **TicketSeries** - Серия электронного билета.
- **TicketNumber** - Номер электронного билета.
- **PushkinCard** - Признак оплаты по Пушкинской карте
- **DonatorPayment** - Флаг - билет оплачен меценатом (0 - нет, 1 - да)
- **LoyaltyPoints** - Кол-во начисленных баллов по программе лояльности за данную операцию.
- **DiscountProgramPercent** - Процент скидки по дисконтной программе

- **PaymentMethodID** - Идентификатор способа оплаты (см. список)
- **CheckCode** - Код для проверки места заказа (при самостоятельной регистрации посетителя)*
- **result_code** - Код ошибки.
- **result_message** - описание ошибки

** если на событие не нужно собирать данные зрителя - код передается пустой.*

[Подарочные сертификаты]

- **Number** Номер (штрих-код) сертификата
- **Amount** Номинал
- **StartDate** Дата начала действия сертификата.
- **StopDate** Дата окончания действия сертификата.
- **Balance** Остаток средств на сертификате
- **SoldAmount** - Сумма, списанная с сертификата при продаже мест. Равна или меньше номинала сертификата. Если SoldAmount равно = 0, то сертификат не использован при покупке. Разница между Amount и SoldAmount не возвращается (сгорает).

[Скидки]

- **BenefitId** - Идентификатор вида скидочной карты
- **BenefitCardId** - Номер скидочной карты
- **BenefitDiscountPercent** - Процент по скидочной карте
- **BenefitDiscountSum** - Сумма скидки по скидочной карте (в случае скидки в твердой сумме)

Обратите внимание!

Данный метод получает список мест заказа и устанавливает на них мет-

ку об оплате.

Все непереданные места будут удалены из заказа.

После оформления покупки, следует сменить номер сессии для зрителя!

Обратите внимание на структуру данных при передаче списка сертификатов для оплаты и структуру данных про скидочным картам!

Идентификаторы способов оплаты

- 1** - Кредитная карта
- 2** - Пушкинская карта
- 3** - Система Быстрых Платежей
- 4** - Плати частями
- 5** - Оплата сертификатом
- 6** - Оплата сертификатом/Кредитная карта
- 7** - Оплата сертификатом/СБП

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="5689" cod_sec="1" row="1" seat="1"
session="UEIIDAJH7567LHFSFJOWHY432" Transac-
tionID="4544445842414" RRN="4544445842414" Payment-
Date="20.05.2025" PaymentTime="18:01:19" />

    [
        <Certificates Number="1234567890"/>
        <Certificates Number="9876543210"/>
        <Discount BenefitId="24" BenefitCardNumber="01234567890"
/>
    ]
</ReqBody>
```

```
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="5689" cod_sec="1" row="1" seat="1" ses-
sion="UEIIDA JH7567LHFSFJOWHY432" price="1000.00"
priceSell="950.00" priceForPay="950.00" ServiceFee="0"
priceSellFull="950.00" priceReturn="950.00" isBenefit="0" Cer-
tificatePermitted="1" reservID="25685" reservDate="20.05.2025"
barcode="922267969684897857" BarCodeType="" Transac-
tionID="4544445842414" RRN="4544445842414" Payment-
Date="20.05.2025" PaymentTime="18:01:19" PromoCodeID="" Promo-
Percent="0" PromoSum="0" STCompanyDiscount="0" Payment-
CardDiscount="0" TicketSeries="" TicketNumber="" PushkinCard="0"
DonatorPayment="0" LoyaltyPoints="0" DiscountProgramPercent=""
PaymentMethodID="5" CheckCode="123456" result_code="0" re-
sult_message="OK"/>

  [
    <Certificates Number="1234567890" Amount="500.00" SoldA-
mount="300.00" Balance="200.00" StartDate="01.08.2020"
StopDate="31.07.2021"/>
    <Certificates Number="9876543210" Amount="500.00" SoldA-
mount="500.00" Balance="0.00" StartDate="01.08.2020"
StopDate="31.07.2021"/>
    <Discount BenefitId="24" BenefitCardNumber="01234567890" Dis-
countPercent="5" DiscountSum="0.00" />
  ]

</AnswerBody>
</GateAnswer>
```

19.2. ПРОВЕРКА МЕСТ ПЕРЕД ПРОДАЖЕЙ

Function CheckSoldTickets(strInp: string): string;

*Метод осуществляет проверку доступности для продажи уже заброниро-
ванных мест*

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **cod_sec** - Идентификатор сектора
- **row** - ряд
- **seat** - место
- **price** - Цена билета
- **reservID** - номер заказа
- **session** - Идентификатор сессии
- **PushkinCard** - Признак оплаты по Пушкинской карте

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события.
- **cod_sec** - Идентификатор сектора.
- **row** - ряд.
- **seat** - место.
- **price** - Цена билета (номинал)*.
- **priceSell** - Цена билета (продажи)*.
- **reservID** - номер заказа
- **session** - Идентификатор сессии,
- **blocked** - Заказ заблокирован и ожидает оплаты (да - 1, нет - 2)
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>  
<ReqLogin>
```

```
<!-- Имя пользователя и пароль для авторизации -->
<UserName>User</UserName>
<UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
  <InputRow NomBilKn="5689" cod_sec="1" row="1" seat="1"
session="UEIIDAJH7567LHFSFJOWHY432" reservID="25685" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="5689" cod_sec="1" row="1" seat="1" ses-
sion="UEIIDAJH7567LHFSFJOWHY432" reservID="25685"
price="1000.00" priceSell="1000.00" blocked="1" result_code="0"
result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

19.3. 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** - Идентификатор комплекта
- **cod_sec** Идентификатор сектора
- **row** - ряд
- **seat** - место
- **session** - Идентификатор сессии

- **TransactionID** - Идентификатор транзакции
- **PaymentDate** Дата оплаты
- **PaymentTime** Время оплаты
- **[RemoteAddress]** - IP адрес зрителя
- **[PaymentMethodID]** - Идентификатор способа оплаты (см. список)

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора
- **row** ряд
- **seat** место
- **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** Идентификатор транзакции платежной системы.
- **PaymentDate** Дата оплаты
- **PaymentTime** Время оплаты
- **PaymentMethodID** - Идентификатор способа оплаты (см. список)
- **CheckCode** - Код для проверки места заказа (при самостоятельной регистрации посетителя)*
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Обратите внимание!

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

В случае удачного осуществления 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.

Идентификаторы способов оплаты

- 1** - Кредитная карта
- 2** - Пушкинская карта
- 3** - Система Быстрых Платежей
- 4** - Плати частями
- 5** - Оплата сертификатом
- 6** - Оплата сертификатом/Кредитная карта
- 7** - Оплата сертификатом/СБП

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow EventsSetID="34" cod_sec="2" row="1" seat="3"
session="SDSU42343DS343433" session="UEIIDAJH7567LHFSFJOWHY432"
TransactionID="4544445842414" RRN="4544445842414" Payment-
```

```
Date="20.05.2025" PaymentTime="18:01:19" />

</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>3</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="256" EventsSetID="34" cod_sec="2" row="1"
seat="3" session="SDSU42343DS343433" price="1000"
priceSell="1000" NameSpektator="Иванов Иван" TelSpekta-
tor="+7(111)111-1144" EmailSpektator="ivanov@mail.ru" Reserv-
ID="2575" TransactionID="4544445842414" RRN="4544445842414" Pay-
mentDate="20.05.2025" PaymentTime="18:01:19" result_code="0" re-
sult_message="OK"/>
  <Row NomBilKn="257" EventsSetID="34" cod_sec="2" row="1"
seat="3" session="SDSU42343DS343433" price="1000"
priceSell="1000" NameSpektator="Иванов Иван" TelSpekta-
tor="+7(111)111-1144" EmailSpektator="ivanov@mail.ru" Reserv-
ID="2575" TransactionID="4544445842414" RRN="4544445842414" Pay-
mentDate="20.05.2025" PaymentTime="18:01:19" result_code="0" re-
sult_message="OK"/>
  <Row NomBilKn="258" EventsSetID="34" cod_sec="2" row="1"
seat="3" session="SDSU42343DS343433" price="1000"
priceSell="1000" NameSpektator="Иванов Иван" TelSpekta-
tor="+7(111)111-1144" EmailSpektator="ivanov@mail.ru" Reserv-
ID="2575" TransactionID="4544445842414" RRN="4544445842414" Pay-
mentDate="20.05.2025" PaymentTime="18:01:19" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

19.4. ВОЗВРАТ МЕСТ

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

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

Описание входных параметров.

- **NomBilKn** - Идентификатор события
 - **cod_sec** Идентификатор сектора.
 - **row** - ряд
 - **seat** - место
 - **session** - Идентификатор сессии
 - **PriceSold** - Цена продажи.
 - **PriceReturn** - Цена возврата
 - **PriceReturnActual** - сумма, возвращенная на карту клиента (с учетом всех удержаний).
 - **[RRN]** - RRN возврата
 - **[ReturnReason]** - Причина возврата
 - **[RemoteAddress]** - IP адрес зрителя
- [Params]** - дополнительные параметры
- **AdminReturn** - возврат осуществляется администратором (только для собственного сайта)

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора
- **name_sec** Наименование сектора
- **row** - ряд
- **seat** - место
- **price** Цена билета (номинал)*. (в случае ошибки равна 0).
- **priceSell** Цена билета (продажи)*. (в случае ошибки равна 0). В случае, если заказ/продажа были с промокодом, цена продажи учитывает скидку по промоко-

ду.

- **PriceSold** - Цена продажи.
- **PriceReturn** - Цена возврата
- **barcode** - штрих-код места
- **reservID** - номер заказа (в случае ошибки равен 0)
- **reservDate** - дата заказа (в случае ошибки пустая)
- **RRN** - RRN возврата
- **ReturnReason** - Причина возврата
- **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.

Также, если был распечатан бланк вместо электронного билета, то возврат возможен только в кассе.

Цена продажи (PriceSold) необходима для контроля цены возврата.

Цена возврата может отличаться от цены продажи и цены билета в меньшую сторону.

До 1 октября 2019г. параметр PriceSold и PriceReturn являются необязательными.

После этой даты будет включена проверка на их наличие.

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
```

```
<UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
  <InputRow NomBilKn="5689" cod_sec="1" row="1" seat="1"
  session="UEIIDA JH7567LHFSFJOWHY432" RRN="5544445842414" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
  всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
  вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="5689" cod_sec="1" row="1" seat="1" ses-
  sion="UEIIDA JH7567LHFSFJOWHY432" price="1000.00"
  priceSell="950.00" priceReturn="950.00" bar-
  code="922267969684897857" reservID="25685"
  reservDate="20.05.2025" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

19.5. ПРОВЕРКА МЕСТ ПЕРЕД ВОЗВРАТОМ

Function CheckReturn(strInp: string): string;

Метод проверяет проданные места перед возвратом, можно ли их вернуть.

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора.
- **row** - ряд
- **seat** - место
- **session** - Идентификатор сессии

- **PriceSold** - Цена продажи.
- **PriceReturn** - Цена возврата
- **PriceReturnActual** - сумма, возвращенная на карту клиента (с учетом всех удержаний).
- **[RRN]** - RRN возврата
- **[ReturnReason]** - Причина возврата
- **[RemoteAddress]** - IP адрес зрителя

- [Params]** - дополнительные параметры

- **AdminReturn** - возврат осуществляется администратором (только для собственного сайта)

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора
- **name_sec** Наименование сектора
- **row** - ряд
- **seat** - место
- **PriceSold** - Цена продажи.
- **PriceReturn** - Цена возврата
- **barcode** - штрих-код места
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>  
<ReqLogin>
```

```
<!-- Имя пользователя и пароль для авторизации -->
<UserName>User</UserName>
<UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
  <InputRow NomBilKn="5689" cod_sec="1" row="1" seat="1"
session="UEIIDAJH7567LHFSFJOWHY432" RRN="5544445842414" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="5689" cod_sec="1" row="1" seat="1" ses-
sion="UEIIDAJH7567LHFSFJOWHY432" priceSell="950.00" priceRe-
turn="950.00" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

19.6. ДОПОЛНИТЕЛЬНЫЕ ДАННЫЕ О ПРОДАЖЕ

Function UpdateSale(strInp: string): string;

Метод передает дополнительные данные о продаже, которые отсутствуют в сам момент продажи

Описание входных параметров.

NomBilKn - Идентификатор события

- **cod_sec** Идентификатор сектора

- **row** - ряд

- **seat** - место

- **session** - Идентификатор сессии

- **RRN** - RRN платежа

Описание полей в выходных данных

NomBilKn - Идентификатор события

- **cod_sec** Идентификатор сектора

- **row** - ряд

- **seat** - место

- **session** - Идентификатор сессии

- **RRN** - RRN платежа

- **result_code** - Код ошибки

- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="5689" cod_sec="1" row="1" seat="1"
session="UEIIDAjh7567LHFSFJOWHY432" RRN="5544445842414" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
```

```
<Row NomBilKn="5689" cod_sec="1" row="1" seat="1" session="UEIIDAJH7567LHFSFJOWHY432" RRN="5544445842414" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

19.7. ДОПОЛНИТЕЛЬНЫЕ ДАННЫЕ О ВОЗВРАТЕ

Function UpdateReturn(strInp: string): string;

Метод передает дополнительные данные о возврате, которые отсутствовали в момент возврата

Описание входных параметров.

NomBilKn - Идентификатор события

- **cod_sec** Идентификатор сектора

- **row** - ряд

- **seat** - место

- **session** - Идентификатор сессии

- **RRN** - RRN платежа

- **priceReturnActual** - Актуальная цена возврата

Описание полей в выходных данных

NomBilKn - Идентификатор события

- **cod_sec** Идентификатор сектора

- **row** - ряд

- **seat** - место

- **session** - Идентификатор сессии

- **RRN** - RRN платежа

- **priceReturnActual** - Актуальная цена возврата

- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
ateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="5689" cod_sec="1" row="1" seat="1"
session="UEIIDA JH7567LHFSFJOWHY432" RRN="5544445842414" priceRe-
turnActual="950.00" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="5689" cod_sec="1" row="1" seat="1" ses-
sion="UEIIDA JH7567LHFSFJOWHY432" RRN="5544445842414" priceRe-
turnActual="950.00" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

19.8. ПРОВЕРКА БИНА КАРТЫ ОПЛАТЫ ДЛЯ ПОЛУЧЕНИЯ СКИДКИ

Function CheckPaymentCardBIN(strInp: string): string;

Метод проверяет по BIN первые 6 цифр карты оплаты и при наличии скидки по карте сообщает об этом

Описание входных параметров.

- **BIN** - Первые 6 цифр карты

Описание полей в выходных данных

- **BIN** - Первые 6 цифр карты

- **CardType** Тип карты (MasterCard, VISA и т.д), если не определяется, то "--".

- **Discount** - Процент скидки по карте

- **result_code** - Код ошибки

- **result_message** - Описание ошибки

Обратите внимание!

В параметре BIN необходимо передавать только первые 6 цифр карты.

Остальные цифры будут проигнорированы.

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow BIN="544223"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>0</RecordCount><!-- Количество записей в воз-
```

```
        вращаемом "RecordSet-e" -->
    </AnswerResult>
    <AnswerBody>
        <Row BIN="544223" CardType="Mastercard" Discount="10.00" re-
        sult_code="0" result_message="OK"/>
    </AnswerBody>
</GateAnswer>
```

20. ЖУРНАЛИРОВАНИЕ

- [Фрагмент журнала операций](#)
- [Сверка состояния проданных и забронированных мест](#)

20.1. 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*
- **[ReservID]** - Номер заказа
- **[NomBilKn]** - Идентификатор события
- **[cod_sec]** - Идентификатор сектора
- **[Row]** - ряд

- **[Seat]** - место
- **[SalesOnly]** - Показать только операции продажа/возврат **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
- **IDSpektator** - идентификатор зрителя/агента,
- **NameSpektator** - имя зрителя/агента ,
- **TelSpektator** телефон зрителя/агента,
- **EmailSpektator** - Email зрителя/агента
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
```

```

<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow DateFrom="12.03.2025" TimeFrom="08:00" Date-
To="12.03.2025" TimeTo="09:00"/>
</ReqBody>
</GateReq>

```

Выходной XML.

```

<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>3</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row ActionDate= "12.03.2025" ActionTime="08:00:01" Ac-
tionDone = "PreRes" NomBilKn="123" cod_sec="1" Row="1" Seat="1"
session="FNKEWHRISNVN394732T" price="1000.00"
priceSell="1000.00" ReservID="" ReservDate="" result_code="0"
result_message="OK" />
    <Row ActionDate= "12.03.2025" ActionTime="08:00:01" Ac-
tionDone = "SetRes" NomBilKn="123" cod_sec="1" Row="1" Seat="1"
session="FNKEWHRISNVN394732T" price="1000.00"
priceSell="1000.00" ReservID="544" ReservDate="12.03.2025" re-
sult_code="0" result_message="OK" />
    <Row ActionDate= "12.03.2025" ActionTime="08:00:01" Ac-
tionDone = "SetSold" NomBilKn="123" cod_sec="1" Row="1" Seat="1"
session="FNKEWHRISNVN394732T" price="1000.00"
priceSell="1000.00" ReservID="544" ReservDate="12.03.2025" re-
sult_code="0" result_message="OK" />
</AnswerBody>
</GateAnswer>

```

20.2. 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]** - Идентификатор события
- **[cod_sec]** - Идентификатор сектора
- **[Row]** - ряд
- **[Seat]** - место

Описание полей в выходных данных

- **ActionDate** - дата совершенной операции
- **ActionTime** - Время совершенной операции
- **Status** - текущий статус места*
- **NomBilKn** - Идентификатор события
- **cod_sec** - Идентификатор 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** - Описание ошибки

Обратите внимание!

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.

*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.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow DateFrom="12.03.2025" TimeFrom="08:00" DateTo=
="12.03.2025" TimeTo="09:00" [NomBilKn="123" cod_sec="1" Row="1"
Seat="1"] />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row ActionDate= "12.03.2025" ActionTime="08:00:01" Status =
"SOL" NomBilKn="123" cod_sec="1" Row="1" Seat="1"
price="1000.00" ReservID="524" ReservDate="12.03.2025" Ga-
teUser="1" GateStatus="SOL" GateReservID="524" result_code="0"
result_message="OK" />
</AnswerBody>
</GateAnswer>
```

21. СЕРТИФИКАТЫ

- [Список свободных номиналов сертификатов](#)
- [Предварительное резервирование сертификатов](#)
- [Бронирование сертификатов](#)
- [Снятие бронирования сертификатов](#)
- [Продажа сертификатов](#)
- [Возврат сертификатов](#)
- [Проверка сертификатов](#)

21.1. СПИСОК СВОБОДНЫХ НОМИНАЛОВ СЕРТИФИКАТОВ

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

Метод возвращает список свободных номиналов сертификатов

Описание полей в выходных данных

- **Qty** - Кол-во доступных сертификатов номинала
- **Amount** - Номинал

- **StartDate** Дата начала действия сертификата.
- **StopDate** Дата окончания действия сертификата.
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>2</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row Qty="1205" Amount="1000" StartDate="10.08.2025"
StopDate="31.12.2025" result_code="0" result_message="OK" />
    <Row Qty="205" Amount="5000" StartDate="10.08.2025"
StopDate="31.12.2025" result_code="0" result_message="OK" />
</AnswerBody>
</GateAnswer>
```

21.2. ПРЕДВАРИТЕЛЬНОЕ РЕЗЕРВИРОВАНИЕ СЕРТИФИКАТОВ

Function PreSetReservationCertificates(strInp: string): string;

Метод осуществляет предварительное резервирование сертификатов

Описание входных параметров.

- **session** - Идентификатор сессии по сертификатам
- **Amount** - Сумма номиналов (в рамках срока действия)
- **Qty**- Кол-во (в рамках срока действия и номинала)
- **StartDate** - Дата начала действия сертификата.
- **StopDate** - Дата окончания действия сертификата.

Описание полей в выходных данных

- **session** - Идентификатор сессии по сертификатам
- **Amount** - Сумма номиналов (в рамках срока действия)
- **Qty**- Кол-во (в рамках срока действия и номинала)
- **StartDate** - Дата начала действия сертификата.
- **StopDate** - Дата окончания действия сертификата.
- **TTL** - Время жизни сессии (в секундах) до ее истечения
- **Number** - номер сертификата
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Обратите внимание!

Обратите внимание на изменение структуры XML в методах работы с сертификатами по сравнению с другими методами!

Если раздел Positions будет пустым или отсутствовать - все сертификаты будут освобождены из сессии

Для изменения кол-ва сертификатов по номиналу - необходимо передавать просто новое количество.

Например:

было: `<Position Amount="1000" Qty="3" ... />` - три сертификата по 1000р.

далее передали `<Position Amount="1000" Qty="5" ... />` - стало 5 (добавилось еще 2) по 1000р.

далее передали `<Position Amount="1000" Qty="2" ... />` - стало 2 (удалили 3) по 1000р.

Если в сессии более одного номинала и кол-во сертификатов в 1 номинале меняется, а в другом нет - нужно передавать оба номинала

В возвращаемых данных передается кол-во сертификатов, относящихся к сессии, на данный момент с учетом переданного добавления/удаления.

Повторный запрос без именованного кол-ва сертификатов только вернет информацию по текущей сессии

Входной XML.

```
<GateReq>
<ReqLogin>
  <!-- Имя пользователя и пароль для авторизации -->
  <UserName>User</UserName>
  <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
  <InputRow session="FNKEW35GESNVN39473GJ" / >
  <Positions>
    <Position Amount="1000" Qty="2"
StartDate="01.01.2025" StopDate="31.12.2025"/>
  </Positions>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>2</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row session="FNKEW35GESNVN39473GJ" TTL="300" />
```

```
<Positions>
  <Position Amount="1000" Qty="2"
StartDate="01.01.2025" StopDate="31.12.2025" result_code="0" re-
sult_message="OK" />
</Positions>
<Items>
  <Item Number="8427" Amount="1000"/>
  <Item Number="8428" Amount="1000"/>
</Items>
</AnswerBody>
</GateAnswer>
```

21.3. БРОНИРОВАНИЕ СЕРТИФИКАТОВ

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

Метод осуществляет бронирование сертификатов

Описание входных параметров.

- **session** - Идентификатор сессии по сертификатам
- **NameSpektator** - ФИО зрителя
- **TelSpektator** - Телефон зрителя
- **EmailSpektator** - Email зрителя

Описание полей в выходных данных

- **Amount** - Сумма номиналов (в рамках срока действия)
- **Qty** - Кол-во (в рамках срока действия и номинала)
- **StartDate** - Дата начала действия сертификата.
- **StopDate** - Дата окончания действия сертификата.
- **session** - Идентификатор сессии по сертификатам
- **ReservID** - Номер заказа. (в случае ошибки отсутствует)
- **ReservDate** - дата истечения заказа (в случае ошибки отсутствует)

- **ReservTime** - время истечения заказа (в случае ошибки отсутствует)
- **Number** - номер сертификата
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow session="FNKEW35GESNVN39473GJ" NameSpekta-
tor="Иванов Иван" TelSpektator="+79990000000" EmailSpekta-
tor="mail@provider.ru" / >
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row session="FNKEW35GESNVN39473GJ" ReservID="52135"
ReservDate="26.04.2025" ReservTime="13:50" NameSpektator="Name"
TelSpektator="+79990000000" EmailSpektator="mail@provider.ru" />

    <Positions>
        <Position Amount="1000" Qty="2"
StartDate="01.01.2025" StopDate="31.12.2025" result_code="0" re-
sult_message="0" />
    </Positions>

    <Items>
        <Item Number="8427" Amount="1000"/>
        <Item Number="8428" Amount="1000"/>
    </Items>
```

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

21.4. СНЯТИЕ БРОНИРОВАНИЯ СЕРТИФИКАТОВ

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

Метод осуществляет снятие бронирования сертификатов

Описание входных параметров.

- **ReservID** - номер заказа сертификатов

Описание полей в выходных данных

- **ReservID** - номер заказа сертификатов

- **result_code** - Код ошибки

- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>  
<ReqLogin>  
    <!-- Имя пользователя и пароль для авторизации -->  
    <UserName>User</UserName>  
    <UserPass>Password</UserPass>  
</ReqLogin>  
<ReqBody>  
    <InputRow ReservID="52135"/>  
</ReqBody>  
</GateReq>
```

Выходной XML.

```
<GateAnswer>  
<AnswerResult>  
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для  
всего запроса-->  
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
```

```
        вращаемом "RecordSet-e" -->
    </AnswerResult>
    <AnswerBody>
        <Row result_code="0" result_message="OK"/>
    </AnswerBody>
</GateAnswer>
```

21.5. ПРОДАЖА СЕРТИФИКАТОВ

Function SetSoldCertificates(strInp: string): string;

Метод осуществляет продажу сертификатов

Описание входных параметров.

- **ReservID** - номер заказа по сертификатам
- **session** - Идентификатор сессии по сертификатам
- **[TransactionID]** - Идентификатор транзакции или RRN (необязательное поле)
- **[PaymentDate]** - Дата оплаты
- **[PaymentTime]** - Время оплаты
- **[PaymentMethodID]** - Идентификатор способа оплаты (см. список)

Описание полей в выходных данных

- **Amount** - Номинал
- **Barcode** - Штрих-код
- **Number** - Номер сертификата
- **ReservID** - Номер заказа
- **TransactionID** - Идентификатор транзакции или RRN
- **PaymentDate** - Дата оплаты
- **PaymentTime** - Время оплаты
- **StartDate** Дата начала действия сертификата.

- **StopDate** Дата окончания действия сертификата.
- **PaymentMethodID** - Идентификатор способа оплаты (см. список)
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Идентификаторы способов оплаты

- 1 - Кредитная карта
- 2 - Пушкинская карта
- 3 - Система Быстрых Платежей
- 4 - Плати частями
- 5 - Оплата сертификатом
- 6 - Оплата сертификатом/Кредитная карта
- 7 - Оплата сертификатом/СБП

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow ReservID="52135" TransactionID="545422144"
PaymentDate="26.04.2025" PaymentTime="13:44"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>2</RecordCount><!-- Количество записей в воз-
```

```
        вращаемом "RecordSet-e" -->
    </AnswerResult>
    <AnswerBody>
        <Row ReservID="52135" TransactionID="545422144" Payment-
        Date="26.04.2025" PaymentTime="13:44" />
        <Positions>
            <Position Amount="1000" Number="8427" Bar-
            code="093005540388415840" StartDate="01.01.2025"
            StopDate="31.12.2025" result_code="0" result_message="OK" />
            <Position Amount="1000" Number="8428" Bar-
            code="673030412275687458" StartDate="01.01.2025"
            StopDate="31.12.2025" result_code="0" result_message="OK" />
        </Positions>
    </AnswerBody>
</GateAnswer>
```

21.6. ВОЗВРАТ СЕРТИФИКАТОВ

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

Метод осуществляет возврат сертификатов

Описание входных параметров.

- **Barcode** - штрихкод сертификата

Описание полей в выходных данных

- **Amount** Номинал
- **Barcode** Штрих-код
- **StartDate** Дата начала действия сертификата.
- **StopDate** Дата окончания действия сертификата.
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
```

```
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow Amount="1000" Barcode="093005540388415840"
StartDate="01.01.2025" StopDate="31.12.2025"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row Amount="1000" Barcode="093005540388415840"
StartDate="01.01.2025" StopDate="31.12.2025" result_code="0" re-
sult_message="OK" />
</AnswerBody>
</GateAnswer>
```

21.7. ПРОВЕРКА СЕРТИФИКАТОВ

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

Метод осуществляет продажу сертификатов

Описание входных параметров.

- **Amount** Номинал.
- **Barcode** - Штрих-код.
- **Number** - Номер

Описание полей в выходных данных

- **Number** - Номер
- **Barcode** - Штрих-код
- **Amount** - Номинал
- **Balance** - Остаток
- **StartDate** - Дата начала действия сертификата.
- **StopDate** - Дата окончания действия сертификата.
- **SitePermitted** - Разрешено оплачивать сертификатом на сайте.
- **OfficePermitted** - Разрешено оплачивать сертификатом на сайте в кассе.
- **SingleUse** - Флаг сертификат разрешен к использованию только 1 раз (1) или несколько (0).
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow Amount="1000" Number="8427" Barcode="093005540388415840"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
```

```
</AnswerResult>
<AnswerBody>
  <Row Amount="1000" Balance="1000" Number="8427" Bar-
code="093005540388415840" StartDate="01.01.2025"
StopDate="31.12.2025" SitePermitted="1" OfficePermitted="1" re-
sult_code="0" result_message="OK" />
</AnswerBody>
</GateAnswer>
```

22. РЕГИСТРАЦИЯ ПОСЕТИТЕЛЕЙ

- [Список типов документов посетителя](#)
- [COVID 19](#)
 - [Обновить данные заказа](#)
 - [Информация о билете](#)
 - [Регистрация посетителя](#)
- [ДИТ - Мос.ру](#)
 - [Поиск места по штрихкоду и номеру заказа/чек-коду](#)
 - [Обновить данные посетителя](#)

22.1. СПИСОК ТИПОВ ДОКУМЕНТОВ ПОСЕТИТЕЛЯ

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

Метод возвращает список типов документов посетителя

Описание полей в выходных данных

- **DocumentTypeID** ID типа документа
- **DocumentTypeName** Наименование типа документа
- **HasSeries** - Наличие серии у документа
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>3</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row DocumentTypeID="1" DocumentTypeName="Внутренний паспорт
гражданина РФ" HasSeries="1" result_code="0" result_message="OK"
/>
    <Row DocumentTypeID="2" DocumentTypeName="Заграничный паспорт
гражданина РФ" HasSeries="0" result_code="0" result_message="OK"
/>
    <Row DocumentTypeID="3" DocumentTypeName="Свидетельство о
рождении" HasSeries="1" result_code="0" result_message="OK" />
</AnswerBody>
</GateAnswer>
```

22.2. COVID 19

- [Обновить данные заказа](#)
- [Информация о билете](#)
- [Регистрация посетителя](#)

22.2.1. ОБНОВИТЬ ДАННЫЕ ЗАКАЗА

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

Обновляет данные зрителя в заказе.

Метод используется для указания данных посетителей в связи с коронавирусной инфекцией

Описание входных параметров.

- **session** - Идентификатор сессии
- **ReservID** - Идентификатор заказа
- **EmailSpektator** - Email зрителя

Если в методе [GetSettings](#) параметр CheckPersonalData равен 1:

- **NomBilKn** - Идентификатор события.
- **ReservID** - Номер заказа
- **session** - Идентификатор сессии
- **cod_sec** - Идентификатор сектора
- **row** - ряд
- **seat** - место
- **VisitorName** - ФИО посетителя
- **VisitorBirthDate** - Дата рождения посетителя (в формате дд.мм.гггг).
- **VisitorDocTypeID** - Идентификатор типа документа посетителя
- **VisitorDocument** - Серия и номер документа
- **VisitorIsSpectator** - Признак, что посетитель является владельцем заказа
- **VisitorIsLess65** - Флаг - посетителю меньше 65 лет
- **VisitorIsNotChronic** - Флаг - у посетителя нет хронических заболеваний
- **VisitorIsNotCovid** - Флаг посетитель не болен COVID 19
- **VisitorIsNotContact** - Флаг - посетитель не был в контакте с больным COVID 19

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события.

- **ReservID** - Номер заказа
- **session** - Идентификатор сессии
- **cod_sec** - Идентификатор сектора
- **row** - ряд
- **seat** - место
- **VisitorName** - ФИО посетителя
- **VisitorBirthDate** - Дата рождения посетителя (в формате дд.мм.гггг).
- **VisitorDocTypeID** - Идентификатор типа документа посетителя
- **VisitorDocument** - Серия и номер документа
- **VisitorIsSpectator** - Признак, что посетитель является владельцем заказа
- **VisitorIsLess65** - Флаг - посетителю меньше 65 лет
- **VisitorIsNotChronic** - Флаг - у посетителя нет хронических заболеваний
- **VisitorIsNotCovid** - Флаг посетитель не болен COVID 19
- **VisitorIsNotContact** - Флаг - посетитель не был в контакте с больным COVID 19
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="123" ReservID="554" ses-
    sion="UEIDJH7980DFDZMKLFFJOWHY432" cod_sec="1" row="1" seat="1"
    VisitorName="Иванов Иван Иванович" VisitorBirthDate="16.07.1998"
    VisitorDocTypeID="" VisitorDocument="" VisitorIsSpectator="0"
    VisitorIsLess65="1" VisitorIsNotChronic="1" Visito-
    rIsNotCovid="1" VisitorIsNotContact="1"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="123" ReservID="554" ses-
sion="UEIDJH7980DFDZMKLFJOWHY432" cod_sec="1" row="1" seat="1"
VisitorName="Иванов Иван Иванович" VisitorBirthDate="16.07.1998"
VisitorDocTypeID="" VisitorDocument="" VisitorIsSpectator="0"
VisitorIsLess65="1" VisitorIsNotChronic="1" Visito-
rIsNotCovid="1" VisitorIsNotContact="1" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

22.2.2. ИНФОРМАЦИЯ О БИЛЕТЕ

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

Возвращает данные билета по штрихкоду. Метод доступен только для собственного сайта.

Описание входных параметров.

- **Barcode** - штрихкод

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события.
- **cod_sec** - Идентификатор сектора.
- **name_sec** - Наименование сектора
- **row** - ряд
- **seat** - место
- **Barcode** - штрихкод

- **StartRegDate** - Дата начала регистрации
- **EventDate** - Дата события (или дата окончания события, если оно с открытой датой)
- **EventTime** - Время события (или время окончания события, если оно с открытой датой)
- **name_show** - Наименование мероприятия,
- **name_t** Наименование театра
- **name_h** Наименование сцены театра
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Если регистрация посетителя по этому билету уже произведена, метод вернет ошибку с кодом 70.

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow Barcode="506434539915007810"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="1462" cod_sec="213" name_sec="Танцевальный
```

```
партер" row="1" seat="1" Barcode="506434539915007810"
EventDate="01.01.2021" EventTime="19:00" name_show="Спектакль"
name_t="Городской театр" name_h="Основная сцена" result_code="0"
result_message="OK"
</AnswerBody>
</GateAnswer>
```

22.2.3. РЕГИСТРАЦИЯ ПОСЕТИТЕЛЯ

Function SetRegistration(strInp: string): string;

Записывает данные о посетителе. Метод доступен только для собственного сайта.

Описание входных параметров.

- **Barcode** - штрихкод
- **VisitorPhone** - Телефон посетителя
- **VisitorEmail** - Email посетителя
- **VisitorName** - ФИО посетителя
- **VisitorBirthDate** - Дата рождения посетителя (в формате дд.мм.гггг).
- **VisitorDocTypeID** - Идентификатор типа документа посетителя
- **VisitorDocument** - Серия и номер документа
- **VisitorIsSpectator** - Признак, что посетитель является владельцем заказа
- **VisitorIsLess65** - Флаг - посетителю меньше 65 лет
- **VisitorIsNotChronic** - Флаг - у посетителя нет хронических заболеваний
- **VisitorIsNotCovid** - Флаг посетитель не болен COVID 19
- **VisitorIsNotContact** - Флаг - посетитель не был в контакте с больным COVID 19

Описание полей в выходных данных

- **NomBilKn** - идентификатор события.
- **Barcode** - штрихкод

- **cod_sec** - идентификатор сектора.
- **name_sec** - наименование сектора
- **row** - ряд
- **seat** - место
- **VisitorPhone** - Телефон посетителя
- **VisitorEmail** - Email посетителя
- **VisitorName** - ФИО посетителя
- **VisitorBirthDate** - Дата рождения посетителя (в формате дд.мм.гггг). Осуществляется проверка на возраст посетителя на момент даты события.
- **VisitorDocTypeID** - Идентификатор типа документа посетителя
- **VisitorDocument** - Серия и номер документа
- **VisitorIsSpectator** - Признак, что посетитель является владельцем заказа
- **VisitorIsLess65** - Флаг - посетителю меньше 65 лет
- **VisitorIsNotChronic** - Флаг - у посетителя нет хронических заболеваний
- **VisitorIsNotCovid** - Флаг посетитель не болен COVID 19
- **VisitorIsNotContact** - Флаг - посетитель не был в контакте с больным COVID 19
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

В случае ошибки "Дата регистрации на мероприятие больше текущей" (код 69) также возвращается параметр:

- **StartRegDate** - Дата начала регистрации.

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
```

```
</ReqLogin>
<ReqBody>
    <InputRow Barcode="506434539915007810" Visi-
torPhone="+79261234567" VisitorEmail="visitor@email.ru" Visi-
torName="Иванов Иван Иванович" VisitorBirthDate="05.08.1976"
VisitorDocTypeID="" VisitorDocument="" VisitorIsSpectator="1"
VisitorIsLess65="1" VisitorIsNotChronic="1" Visito-
rIsNotCovid="1" VisitorIsNotContact="1"/>

</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row Barcode="506434539915007810" NomBilKn="1462"
cod_sec="213" row="1" seat="1" VisitorPhone="+79261234567" Visi-
torEmail="visitor@email.ru" VisitorName="Иванов Иван Иванович"
VisitorBirthDate="05.08.1976" VisitorDocTypeID="" VisitorDocu-
ment="" VisitorIsSpectator="1" VisitorIsLess65="1" Visito-
rIsNotChronic="1" VisitorIsNotCovid="1" VisitorIsNotContact="1"
result_code="0" result_message="OK"/>

</AnswerBody>
</GateAnswer>
```

22.3. ДИТ - МОС.РУ

- [Поиск места по штрихкоду и номеру заказа/чек-коду](#)
- [Обновить данные посетителя](#)

22.3.1. ПОИСК МЕСТА ПО ШТРИХКОДУ И НОМЕРУ ЗАКАЗА/ЧЕК-КОДУ

Function GetVisitorTicketData(strInp: string): string;

Метод возвращает данные места по штрихкоду и номера заказа или чек-коду.

Описание входных параметров.

- **Barcode** - Штрихкод.
- **ReservID** - Номер заказа
- **CheckCode** - Чек код

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события.
- **cod_sec** Идентификатор сектора.
- **name_sec** Наименование сектора
- **row** - ряд
- **seat** - место
- **Barcode** - штрихкод
- **priceSell** Цена продажи билета (без сервисного сбора, включая ВСЕ скидки и наценки).
- **ServiceFee** - Сервисный сбор
- **SoldByCertificate** Признак продажи по сертификату
- **session** - Идентификатор сессии
- **RequestVisitorData** - Запрашивать данные зрителя (0 - не запрашивать, 1 - запрашивать только ФИО, 2 - запрашивать ФИО + тип, серию и номер)
- **EventDate** - Дата события (или дата окончания события, если оно с открытой датой)
- **EventTime** - Время события (или время окончания события, если оно с открытой датой)
- **Age** - Возрастное ограничение (INT), -1 - не определен. (если передается, например, 16, это означает 16+)
- **EventDuration** Продолжительность события
- **name_show** - Наименование мероприятия,
- **name_t** Наименование театра

- **name_h** Наименование сцены театра
- **ReservID** - Номер заказа
- **CheckCode** - Чек код
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow Barcode="506434539915007810" ReservID="222" />
или
    <InputRow Barcode="506434539915007810"
CheckCode="140921" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="1462" cod_sec="213" name_sec="Танцевальный
партер" row="1" seat="1" Barcode="506434539915007810"
session="324234234234234234" RequestVisitorData="2"
EventDate="01.08.2025" EventTime="19:00" name_show="Спектакль"
name_t="Городской театр" name_h="Основная сцена" result_code="0"
result_message="OK" />
</AnswerBody>
</GateAnswer>
```

22.3.2. ОБНОВИТЬ ДАННЫЕ ПОСЕТИТЕЛЯ

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

Метод записывает / обновляет данные о посетителях по местам заказа.

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **ReservID** - Номер заказа
- **session** Идентификатор сессии
- **cod_sec** Идентификатор сектора
- **row** - ряд
- **seat** - место
- **VisitorFirstName** - Имя посетителя (обязательное поле)
- **VisitorMiddleName** - Отчество посетителя (обязательное поле при наличии отчества)
- **VisitorLastName** - Фамилия посетителя
- **VisitorDocTypeID** - Идентификатор типа документа посетителя (обязательное в случае RequestVisitorData = 2 в GetEventList)
- **VisitorDocumentSeries** - Серия документа
- **VisitorDocumentNumber** - Номер документа
- **OperatorUserID** - Уникальный идентификатор зрителя у билетного оператора (только для бил. операторов т.е. "сторонних" сайтов).
- **SSOID** - Идентификатор Mos.ru
- **Confirmed14** - Подтверждение, что на момент события зрителю нет 14 лет
- **TicketAsGift** - флаг билет в подарок.
- **BirthDate** - Дата рождения. (Для детей, младше 14 лет. Формат дд.мм.гггг).

Описание полей в выходных данных

- **NomBilKn** - идентификатор события.
- **ReservID** - Номер заказа
- **session** идентификатор сессии
- **cod_sec** идентификатор сектора.
- **name_sec** Наименование сектора
- **row** ряд
- **seat** место
- **VisitorFirstName** - Имя посетителя
- **VisitorMiddleName** - Отчество посетителя
- **VisitorLastName** - Фамилия посетителя
- **OperatorUserID** - Уникальный идентификатор зрителя у билетного оператора
- **VisitorDocumentSeries** - Серия документа (в ответе для безопасности будет передано "OK" если была передана серия и "" если нет)
- **VisitorDocumentNumber** - Номер документа (в ответе для безопасности будет передано "received" если был передан номер и "" если нет)
- **SSOID** - идентификатор Mos.ru (в ответе для безопасности будет передано "received" если был передан SSOID и "" если нет)
- **Confirmed14** - Подтверждение, что на момент события зрителю нет 14 лет (в ответе для безопасности будет передано "OK" если был передан Confirmed14 и "" если нет)
- **BirthDate** - Дата рождения (в ответе для безопасности будет передано "OK" если был передан Confirmed14 и "" если нет)
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>  
<ReqLogin>
```

```
<!-- Имя пользователя и пароль для авторизации -->
<UserName>User</UserName>
<UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
  <InputRow ReservID="222" session="SDSU4234LJH433" Nom-
BilKn="1024" cod_sec="1" Row="1" Seat = "1" VisitorLast-
Name="Иванов" VisitorFirstName="Иван" VisitorMiddle-
Name="Иванович" VisitorDocumentTypeId="1" VisitorDocumentSer-
ies="OK" VisitorDocumentNumber="recieved" Opera-
torUserID="op_123456"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>0</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row session="SDSU4234LJH433" ReservID="222" NomBilKn="1024"
cod_sec="1" row="1" seat="1" VisitorLastName="Иванов" Visi-
torFirstName="Иван" VisitorMiddleName="Иванович" Opera-
torUserID="op_123456" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

23. ЗРИТЕЛИ

- [Изменение данных зрителя](#)
- [Получить данные по зрителю](#)
- [Список карт зрителя](#)
- [Валидация карты](#)

23.1. ИЗМЕНЕНИЕ ДАННЫХ ЗРИТЕЛЯ

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

Метод изменяет некоторые данные зрителя

Описание входных параметров.

- **IDSpektator** - Идентификатор зрителя. (для идентификации)
- **EmailSpektator** - Email зрителя (для идентификации)
- **NameSpektator** - ФИО зрителя
- **[RegionID]** - Идентификатор [региона](#) зрителя
- **[RegionCityID]** - Идентификатор [города](#) зрителя
- **[Gender]** - Пол зрителя (0 - не указан, 1 - женский пол, 2 мужской пол, 3 - неопределен)
- **[BirthDate]** - Дата рождения в формате дд.мм.гггг

Описание полей в выходных данных

- **IDSpektator** - Идентификатор зрителя.
- **EmailSpektator** - Email зрителя
- **NameSpektator** - ФИО зрителя
- **[RegionID]** - Идентификатор [региона](#) зрителя
- **[RegionCityID]** - Идентификатор [города](#) зрителя
- **[Gender]** - Пол зрителя (0 - не указан, 1 - женский пол, 2 мужской пол, 3 - неопределен)
- **[BirthDate]** - Дата рождения в формате дд.мм.гггг
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Обратите внимание!

Метод изменяет переданные данные зрителя. Email и телефон остается неизменными!

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow IDSpektator="1253" EmailSpekta-
tor="ivanov@mail.ru" NameSpektator="Иванов Иван Петрович" Re-
gionID="7700000000000" RegionCityID="7700000000000" Gender="1"
BirthDate="12.09.2002" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row IDSpektator="1253" EmailSpektator="ivanov@mail.ru"
NameSpektator="Иванов Иван Петрович" RegionID="7700000000000"
RegionCityID="7700000000000" Gender="1" BirthDate="12.09.2002"
result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

23.2. ПОЛУЧИТЬ ДАННЫЕ ПО ЗРИТЕЛЮ

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

Метод возвращает данные зрителя по дисконтной программе и программе лояльности

Описание входных параметров.

- **IDSpektator** - Идентификатор зрителя. (для идентификации)

- **EmailSpektator** - Email зрителя (для идентификации)

Описание полей в выходных данных

- **IDSpectator** - Идентификатор зрителя.
- **EmailSpektator** Email зрителя (для идентификации)
- **DiscountAmount** Сумма купленных билетов по дисконту
- **RegionID** - Идентификатор региона зрителя,
- **RegionCityID** - Идентификатор города зрителя,
- **RegionName** - [Регион](#) зрителя,
- **CityName** - [Город](#) зрителя,
- **DiscountQty** Кол-во купленных билетов по дисконту
- **CurrentPercent** - Текущий процент скидки по дисконту
- **CurrentPoints** - Кол-во накопленных баллов
- **SumToNextDiscount** - Сумма оставшаяся до начисления следующей скидки
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow IDSpektator="1253" EmailSpekta-
tor="ivanov@mail.ru"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row IDSpektator="1253" EmailSpektator="ivanov@mail.ru" Dis-
countAmount="5000" DiscountQty="2" Points="125" CurrentPer-
cent="10" result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

23.3. СПИСОК КАРТ ЗРИТЕЛЯ

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

*Метод возвращает список дисконтных, накопительных и иных карт зрите-
ля*

Описание входных параметров.

- **IDSpektator** - Идентификатор зрителя. (для идентификации)
- **EmailSpektator** - Email зрителя (для идентификации)

Описание полей в выходных данных

- **IDSpektator** - Идентификатор зрителя
- **EmailSpektator** - Email зрителя
- **CardNumber** Номер дисконтной карты
- **CardBarcode** Штрих-код дисконтной карты
- **DateFrom** - Срок действия дисконтной карты с
- **DateTo** - Срок действия дисконтной карты по
- **isActive** - Флаг активности карты

- **DiscountCard** - Признак дисконтной карты
- **SumToNextDiscount** - Сумма оставшаяся до начисления следующей скидки
- **LoyaltyCard** - Признак карты лояльности
- **CurrentPercent** - Текущий процент скидки по дисконту
- **CurrentPoints** - Кол-во накопленных баллов
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow IDSpektator="1253" EmailSpekta-
tor="ivanov@mail.ru"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>3</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row IDSpektator="1253" EmailSpektator="ivanov@mail.ru"
CardNumber="000125" CardBarcode="123012456345789678" Date-
From="01.01.2025" DateTo="31.12.2025" isActive="1" Discount-
Card="1" LoyaltyCard="0" CurrentPercent="10" Points="0" re-
sult_code="0" result_message="OK"/>
    <Row IDSpektator="1253" EmailSpektator="ivanov@mail.ru"
CardNumber="000126" CardBarcode="876543210987654125" Date-
From="01.01.2025" DateTo="31.12.2025" isActive="1" Discount-
Card="0" LoyaltyCard="1" CurrentPercent="0" Points="125" re-
```

```
sult_code="0" result_message="OK"/>
  <Row IDSpektator="1253" EmailSpektator="ivanov@mail.ru"
CardNumber="000074" CardBarcode="123012438545789421" Date-
From="01.01.2024" DateTo="31.12.2024" isActive="0" Discount-
Card="1" LoyaltyCard="0" CurrentPercent="0" Points="0" re-
sult_code="0" result_message="OK"/>

<Row result_code="0" result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

23.4. ВАЛИДАЦИЯ КАРТЫ

Function ValidateCard(strInp: string): string;

метод возвращает валидна ли дисконтная карта или карта лояльности и данные по карте

Описание входных параметров.

- **CardNumber** Номер или штрихкод карты

Описание полей в выходных данных

- **IDSpektator** - Идентификатор зрителя
- **EmailSpektator** - Email зрителя
- **CardNumber** Номер дисконтной карты
- **CardBarcode** Штрих-код дисконтной карты
- **isValidate** - Флаг валидности карты
- **DateFrom** - Срок действия дисконтной карты с
- **DateTo** - Срок действия дисконтной карты по
- **isActive** - Флаг активности карты
- **DiscountCard** - Признак дисконтной карты
- **SumToNextDiscount** - Сумма оставшаяся до начисления следующей скидки

- **LoyaltyCard** - Признак карты лояльности
- **CurrentPercent** - Текущий процент скидки по дисконту
- **CurrentPoints** - Кол-во накопленных баллов
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow CardNumber="125"/>
    <InputRow CardNumber="74"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row CardNumber="000125" CardBarcode="123012456345789678"
DateFrom="01.01.2025" DateTo="31.12.2025" isValid="1" isAc-
tive="1" IDSpectator="1234" result_code="0" re-
sult_message="OK"/>
    <Row CardNumber="000074" CardBarcode="123012438545789421"
DateFrom="01.01.2024" DateTo="31.12.2024" isValid="0" isAc-
tive="0" IDSpectator="1234" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

- [Выдать приглашения \(служебные места\)](#)
- [Отмена приглашений \(снять служебные места\)](#)
- [Список сетов служебных мест](#)
- [Список мест сета служебных мест](#)

24.1. ВЫДАТЬ ПРИГЛАШЕНИЯ (СЛУЖЕБНЫЕ МЕСТА)

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

Метод выдает служебные места для отправки их по эл. почте

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии
- **NameSpektator** - ФИО зрителя
- **TelSpektator** - Телефон зрителя
- **EmailSpektator** - Email зрителя
- **[RemoteAddress]** - IP адрес зрителя

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место

- **session** - Идентификатор сессии
- **SpektatorID** - Идентификатор зрителя
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="1234" cod_sec="34" row="1" seat="5"
session="903JNSKLSJS82WW8243NF09" NameSpektator="Иванов Иван"
TelSpektator="+7(111)111-1144" EmailSpektator="ivanov@mail.ru"
/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row NomBilKn="1234" cod_sec="34" row="1" seat="5" ses-
sion="903JNSKLSJS82WW8243NF09" SpektatorID="234" result_code="0"
result_message="OK"/>
</AnswerBody>
</GateAnswer>
```

24.2. ОТМЕНА ПРИГЛАШЕНИЙ (СНЯТЬ СЛУЖЕБНЫЕ МЕСТА)

Function FreeOfficialPlaces(strInp: string): string;

Метод отменяет приглашения (снимает служебные места)

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии
- **[RemoteAddress]** - IP адрес зрителя

Описание полей в выходных данных

- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="1234" cod_sec="34" row="1" seat="5"
session="903JNSKLSJS82WW8243NF09" />
</ReqBody>
```

```
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row NomBilKn="1234" cod_sec="34" row="1" seat="5" ses-
sion="903JNSKLSJS82WW8243NF09" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

24.3. СПИСОК СЕТОВ СЛУЖЕБНЫХ МЕСТ

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

Метод возвращает список сетов (наборов) выданных службных мест. Сеты объединены по дате/времени выдачи и зрителю.

Описание входных параметров.

- **NomBilKn** - Идентификатор события
- **[DateFrom]** - Дата начала периода выдачи
- **[DateTo]** - Дата окончания периода выдачи
- **[SpektatorID]** - Идентификатор зрителя

Описание полей в выходных данных

- **SetID** - Идентификатор сета
- **SendDate** - Дата выдачи
- **Qty** - Кол-во билетов в сете

- **SpektatorID** - Идентификатор зрителя
- **NameSpektator** - Имя зрителя
- **TelSpektator** - Телефн зрителя
- **EmailSpektator** - Email зрителя
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow NomBilKn="685" />
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
    <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
    <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
    <Row SetID="245" SendDate="25.07.2025" SpektatorID="32"
NameSpektator="Петров Петр" TelSpektator="+7 (123) 123-4567"
EmailSpektatr="petrov@mail.ru" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

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

Метод возвращает список приглашений (служебных мест) указанного сета (набора).

Описание входных параметров.

- **SetID** - Идентификатор сета (набора) служебных мест

Описание полей в выходных данных

- **SetID** - Идентификатор сета (набора) служебных мест
- **NomBilKn** - Идентификатор события
- **cod_sec** Идентификатор сектора.
- **row** ряд
- **seat** место
- **session** - Идентификатор сессии (если выдан не текущим пользователем шлюза, то пустое значение)
- **result_code** - Код ошибки
- **result_message** - Описание ошибки

Входной XML.

```
<GateReq>
<ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>User</UserName>
    <UserPass>Password</UserPass>
</ReqLogin>
<ReqBody>
    <InputRow SetID="245"/>
</ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
<AnswerResult>
  <ResultCode>0</ResultCode><!-- Код ответа шлюза, общий для
всего запроса-->
  <RecordCount>1</RecordCount><!-- Количество записей в воз-
вращаемом "RecordSet-e" -->
</AnswerResult>
<AnswerBody>
  <Row SetID="245" cod_sec="34" row="1" seat="5" ses-
sion="903JNSKLSJS82WW8243NF09" result_code="0" re-
sult_message="OK"/>
</AnswerBody>
</GateAnswer>
```

ПРИМЕРЫ[Работа с SOAP на языке PHP](#)[Пример запроса списка мест проведения](#)[Пример запроса списка залов места проведения](#)**25. 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 = 'login'; //Логин
$pass = 'password'; //Пароль

$q = ' ?';

$quot = '"';

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

//Получаем массив со списков сцен театра с кодом «8».

$request = '<?xml version="1.0" encoding="UTF-8"'.$q.'>
    <GateReq>
        <ReqLogin>
            <UserName>' . $login . '</UserName><UserPass>' . $pass . '</UserPass>
        </ReqLogin>
        <IP>' . $_SERVER['REMOTE_ADDR'] . '</IP>
        <ReqBody><InputRow cod_t="' . $cod_t . '"></ReqBody>
    </GateReq>';

```

```
$halls = get_value_by_path(parse_xml($client-
>GetHallList($request)), 'GateAnswer/AnswerBody');

//Проходим по списку сцен

for($i=0; $i< count($halls)-1; ++$i) {
    $hall = $halls[$i]['attributes'];

    echo $hall['name_h']. '<BR>'; //Выводим на печать наименование
сцены
    echo $hall['Address_h']. '<BR>'; //Выводим на печать адрес
сцены
    echo $hall['Tel_h'] . '<BR>'; //Выводим на печать тел. сцены
}
```

26. ПРИМЕР ЗАПРОСА СПИСКА МЕСТ ПРОВЕДЕНИЯ

Входной XML.

```
<GateReq>
  <ReqLogin>
    <!-- Имя пользователя и пароль для авторизации -->
    <UserName>Gate</UserName>
    <UserPass>dasjdfpojasdj</UserPass>
  </ReqLogin>
  <ReqBody>
    <InputRow />
  </ReqBody>
</GateReq>
```

Выходной XML.

```
<GateAnswer>
  <AnswerResult>
    <ResultCode>0</ResultCode>
    <RecordCount>14</RecordCount>
  </AnswerResult>
  <AnswerBody>
    <Row cod_t="25" name_t="ЕТ 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" re-
sult_code="0"/>
        <Row cod_t="12" name_t="МХТ им. А.П. Чехова" re-
sult_code="0"/>
        <Row cod_t="28" name_t="Саратовский драмтеатр" re-
sult_code="0"/>
        <Row cod_t="29" name_t="Театр им. Моссовета" re-
sult_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" re-
sult_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" re-
sult_code="0"/>
        <Row cod_t="26" name_t="ЦДКЖ" result_code="0"/>
    </AnswerBody>
</GateAnswer>

```

27. ПРИМЕР ЗАПРОСА СПИСКА ЗАЛОВ МЕСТА ПРОВЕДЕНИЯ

Входной XML.

```

<GateReq>
    <ReqLogin>
        <!-- Имя пользователя и пароль для авторизации -->
        <UserName>Gate</UserName>
        <UserPass>dasjdfpojasdj</UserPass>
    </ReqLogin>
    <ReqBody>
        <InputRow cod_t="12"/>
    </ReqBody>
</GateReq>

```

Выходной XML.

```

<GateAnswer>
    <AnswerResult>
        <ResultCode>0</ResultCode>
        <RecordCount>4</RecordCount>
    </AnswerResult>

    <AnswerBody>
        <Row cod_th="31" name_h="МАЛАЯ ЦЕНА" result_code="0"/>
        <Row cod_th="32" name_h="НОВАЯ ЦЕНА" result_code="0"/>
        <Row cod_th="30" name_h="ОСНОВНАЯ ЦЕНА" re-
sult_code="0"/>
    </AnswerBody>
</GateAnswer>

```

```
<Row cod_th="52" name_h="Саратовский драмтеатр" re-  
sult_code="0"/>  
</AnswerBody>  
</GateAnswer>
```