Интеграция Bitrix24 и otmechalka.com

# Термины

API менеджер – система, которую разрабатывает РШШ и она находится у них под контролем. Эта система занимается тем, что связывает API двух систем – Битрикс24 и Отмечалки.

# Некоторые оговоренные особенности

Филиалы, направления и типы абонементов будут передаваться по id. Эти id будет хранить на своей стороне АПИ менеджер. Со стороны Отмечалки изменений с этими сущностями не происходит.

В Отмечалку в Посетителей и Абонементы нужно добавить новые поля (externalId) и сохранять в них внешний id этих сущностей (это id, которые генерируются для этих сущностей и хранятся на стороне Битрикс24; РШШ будут использовать их для связки АПИ Битрикс24 с АПИ Отмечалки). Затем по этим идентификаторам можно будет производить поиск при помощи API. Во всех случаях externalId должно быть целым числом больше нуля.

За уникальностью этого идентификатора следит система API менеджера. Если в Отмечалке каким-то образом случайно у двух посетителей (или двух абонементов) появится одинаковый externalId, то при осуществлении поиска будет возвращаться первая попавшаяся запись.

# Общее описание API Otmechalka

Все запросы к АПИ выполняются через единую точку входа:

https://rshsh.otmechalka.com/apiv2/<entity>/<action>

Параметры, <entity> и <action> указывают на сущность, с которой производится действие, и собственно действие (метод), которое производится над сущностью.

Все параметры функций передаются в виде json объектов, в котором содержатся как параметры самой функции, так и подпись запроса, которая формируется по алгоритму, описанному ниже.

Название http параметра, который содержит в себе описанную выше строку json: "request". Дальше будет описано, как использовать эту информаицю.

Для работы с АПИ, организации РШШ нужно создать нового пользователя в Отмечалке, от имени которого будут выполняться все действия в системе по АПИ. Этот пользователь, как и все остальные, буде иметь свой логин и пароль. Мы рекомендуем создать виртуального сотрудника-пользователя специально для API, а не использовать аккаунт какого-либо реального сотрудника, чтобы в будущем избежать путаницы с тем кто что сделал в системе.

Может использоваться логин и пароль любого ранее созданного пользователя. В данный момент роль пользователя в системе Отмечалка не учитывается при работе с API. Это значит, что всем пользователям, которые имеют логин/пароль для входа в систему, доступны все функции API, не зависимо от их роли.

## Общий порядок работы по АПИ

1. Авторизация пользователя и создание идентификатора сессии
2. Выполнение одного или нескольких запросов АПИ с указанным идентификатором сессии.

Мы рекомендуем выполнять примерно следующий алгоритм при работе с АПИ Отмечалка:

* проверить предыдущий сохраненный идентификатор сессии на актуальность (для этого есть специальная функция АПИ)
* если предыдущей сессии нет или она неактуальна, то
  + авторизоваться
  + сохранить новый идентификатор сессии
* выполнить запрос к АПИ, используя актуальную сессию (новую или старую).

Также РШШ может хранить у себя время последнего обращения к АПИ и исключить из этого алгоритма постоянную проверку актуальности сессии.

На стороне Отмечалки сессия остается актуальной на протяжении 5-ти дней. Если за время жизни сессии выполнить хотя бы один запрос, то время сессии продлевается, и она остается актуальной на протяжении последующих 5-ти дней.

Обращение к АПИ проверки актуальности сессии - также продлевает время ее жизни.

Если сессия неактуальна, то все запросы к АПИ будут возвращать ошибку. Коды ошибок см. в соответствующем разделе.

Каждый запрос должен быть подписан секретным ключом. Детальнее о подписи запросов см. соответствующий раздел.

## Структура запроса

**Запрос имеет следующую структуру**

{

parameters: {

name: "Направление 1",

shortName: "Напр 1"

},

sessionId: ...,

sign: ...

}

Параметры parameters, sessionId и sign должны обязательно присутствовать в каждом запросе.

* поле parameters содержит ассоциативный массив с параметрами запроса, которые передаются непосредственно в функцию АПИ. Если функция не предполагает получения параметров, то должно быть:

parameters: {}

* поле sessionId должно содержать идентификатор сессии. Может быть пустой строкой.
* Поле sign должно содержать подпись запроса. Как формируется подпись описано в соответствующем разделе.

## Структура ответа

{

ok:1,

errorCode:0,

result:

{

"id": "1",

"name": "Petr",

"lastName": "Cherin"

}

}

В ответ на каждый запрос присылается json-сообщение, которое имеет структуру, указанную выше.

Описание полей:

ok – общий результат выполнения запроса. Если = 1, то запрос выполнился успешно. Если = 0, то произошла ошибка, информация о которой указана в поле errorCode.

errorCode – код ошибки, если запрос был неуспешным. Коды ошибок см. в соответствующем разделе.

result – объект с результатом выполнения запроса. Может не содержать данных, в зависимости от запроса и успешности выполнения функции.

# Перечень и описание функций АПИ

## Создание посетителя

Метод создает нового посетителя.

Группы посетителя не передаются и устанавливаются автоматически. Посетитель привязывается ко всем группам, которые соответствуют переданным филиалам и направлениям.

POST /visitor/add

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Обязательно | Пример заполнения | Комментарий |
| externalVisitorId | нет | 39558 | ID записи из внешней системы, по которому можно будет делать поиск. Целое число > 0 |
| lastName | Да | Петров | Фамилия |
| name | Да | Василий | Имя |
| middleName | Нет | Евгениевич | Отчество |
| phone | Да | 79261234567 | Телефон посетителя в международном формате. Без + и других символов и без разделителей. Только цифры. |
| email | Нет | ivan.pe@gmail.com |  |
| coursesIds | Нет | [11, 22] | Перечень id направлений из Отмечалки. Массив. Может быть пустым. |
| branchOfficesIds | Да | [55, 66] | Перечень id филиалов из Отмечалки, куда нужно записать посетителя. Если не передано, то вернется ошибка нехватки обязательных данных, посетитель без филиала быть не может. |
| birthDate | нет | 1375315200 | День рождения посетителя. Указывается в unixtimestamp. Если не передано, то ДР будет не определён. Может быть отрицательным, что будет означать дату дня рождения ранее 01.01.1970. |
| sex | да | 0, 1 | 0 – женский, 1 - мужской |
| addInfo | нет | Любая строка | Дополнительный комментарий к посетителю.  На стороне Отмечалки этот комментарий сохраняется в "Историю по клиенту" в виде текстовой записи. Также этот комментарий устанавливается "закрепленным вверху". |

Ответ:

{

"id": 12456,

// остальные поля посетителя

"isArchive": 0, 1 // признак архивного посетителя. 0 – посетитель активный, 1 – архивный.

}

## Редактирование посетителя

Метод редактирует посетителя.

Посетитель должен быть привязан ко всем группам, которые соответствуют переданным филиалам и направлениям. В этом запросе передается просто актуальный список филиалов/направлений, а не те филиалы/направления, куда посетителя нужно добавить. Это значит, что посетитель не просто будет добавлен в указанные филиалы/направления, а он будет также удален из тех филиалов/направлений, которые не были переданы в запроса.

Посетитель может быть не удален из какого-либо филиала/направления, если он записан в группу по активному на данный момент абонементу.

Запрос:

POST /visitor/edit/

В качестве идентификатора может использовать один из вариантов:

{

"visitorId": 123456

...

}

Или

{

" extetnalVisitorId": 654321

...

}

Передать нужно хотя бы один из этих параметров. Если переданы оба, то приоритет получит параметр visitorId, а extetnalVisitorId будет проигнорирован.

Все обязательные поля также нужно обязательно передавать, даже если они не изменились. Если не передать обязательный параметр, то в ответе вернется ошибка валидации.

Если параметр не обязательный и он не передан, то он будет просто проигнорирован и в Отмечалке останется предыдущее сохраненное значение.

По сути параметры все те же, что и при добавлении посетителя, кроме extetnalVisitorId и birthDate.

При редактировании посетителя extetnalVisitorId используется для поиска посетителя, а не для сохранения нового значения.

Поле birthDate - если ранее ДР у посетителя был установлен и если не передать ДР, то ДР у посетителя будет удален.

Ответ:

Аналогичный ответу при добавлении посетителя.

## Восстановление клиента из архива

Метод восстанавливает клиента из архива.

Запрос:

POST /visitor/fromarchive

В качестве идентификатора может использовать один из вариантов:

{

"visitorId": 123456

...

}

Или

{

" extetnalVisitorId": 654321

...

}

Ответ:

{} - пустой

## Получение клиента

Метод возвращает клиента по его идентификатору.

Запрос: GET /visitor/get

В качестве идентификатора может использовать один из вариантов:

{

"visitorId": 123456

...

}

Или

{

"extetnalVisitorId": 654321

...

}

Ответ:

{

// данные по посетителю, как из запроса добавления посетителя

}

## Продажа абонемента

Метод добавляет в Отмечалку продажу нового абонемента посетителю. При этом добавляется и денежная операция, а посетитель записывается в группы (если он ранее не был записан).

Группы абонемента не передаются и устанавливаются автоматически - посетитель должен быть записан во все группы, которые указаны в типе абонемента. Этим методом можно создать только те абонементы, где заранее указано число посещений и срок действия. Динамические или не ограниченные абонементы так продать нельзя. Проданный абонемент таким образом считается оплаченным на 100%.

Обратите внимание, что в запросе НЕ указывается стоимость абонемента. Стоимость абонемента определяется типом продаваемого абонемента и берется из настроек абонементов в Отмечалке.

Запрос: POST /ticket/add

Поля запроса

|  |  |  |  |
| --- | --- | --- | --- |
| Название поля | Обязательно | Пример заполнения | Комментарий |
| externalTicketId | нет | 111333 | ID абонемента из внешней системы, по которому можно будет делать поиск. Значение записывается параллельно с id абонемента в Отмечалке. Целое число > 0 |
| externalVisitorId | нет | 939 | Несмотря на то, что оба параметра являются необязательными, предать нужно хотя бы один из этих параметров. Если переданы оба, то приоритет получит параметр visitorId, а externalId будет проигнорирован.  Этот параметр указывает, какому именно посетителю продается абонемент.  visitorExternalId – внешний id посетителя. Целое > 0.  visitorId – id посетителя в Отмечалке. Целое > 0. |
| visitorId | нет | 22981 |
| ticketTypeId | да | 55442 | Id типа абонемента, который продается посетителю. Целое > 0. |
| discountMoney | нет | 150.30, 100, 400.5, 350.00 | Скидка, которая была предоставлена покупателю при продаже этого абонемента.  Этот параметр может принимать одно из следующих значений:  - <параметр в запросе отсутствует> - скидка не будет применена;  - <пустая строка, false или null> - скидка не будет применена;  - 0, 0.0, 0.00 – скидка не будет применена;  - положительное дробное число – будет применена указанная скидка в деньгах (а не в %).  Значение должно быть положительным дробным числом с точностью до двух знаков, в качестве разделителя должна быть использована точка.  Если передано число с излишней точностью (например, 10.001), то будет возвращена ошибка валидации.  Если скидка будет превышать полную стоимость абонемента, то скидка будет уменьшена до стоимости абонемента, при этом никаких уведомлений об ошибке выдано не будет. |
| issueDate | нет | 21.12.2021 | Дата продажи абонемента. Этой датой будет зафиксирована денежная операция в Отмечалке.  Формат строго "dd.mm.YYYY". |
| branchOfficeId | да | 34 | ID филиала, в котором продается абонемент. Это значит, что в Отмечалке деньги пойдут в кассу этого филиала или на безналичный счет этого филиала (в зависимости от параметра isNonCash).  Целое > 0. |
| isNonCash | да | 0 или 1 | Указывается на то, какая это операция, наличная или безналичная:  0 – наличная  1 – безналичная  Обратите внимание, что если указывается 1, то у филиала branchOfficeId в Отмечалке должен быть установлен безналичный счет. |

## Получение абонемента

Метод возвращает абонемент по его внешнему идентификатору.

Запрос: GET /tickets/get

В качестве идентификатора может использовать один из вариантов:

{

"ticketId": 123456

...

}

Или

{

" extetnalTicketId": 654321

...

}

Ответ:

{

"id": 24545676786,

// остальные поля абонемента

}

## Авторизация к API

Для авторизации используются логин и пароль виртуального или реального сотрудника, который добавлен в Отмечалку.

Вход (авторизация) выполняется отправкой запроса по адресу:

POST auth/authorize

Отправляется запрос вида:

parameters: {

username: "user",

password: "123456"

},

sessionId: "",

sign: ...

В результате успешной авторизации получается ответ:

{

ok:1,

errorCode:0,

result:

{

sessionId: "1234567890kjhgfds"

}

}

При каждом последующем запросе нужно передавать sessionId в соответствующем параметре.

Нужно обратить внимание на то, что при отправке запроса на авторизацию значение sessionId равно пустой строке.

## Проверка актуальности сессии

POST auth/sessionalive

**ВНИМАНИЕ!**

* Запрос выполняется методом POST, несмотря на то, что это операция чтения.

Отправляется запрос вида:

parameters: {},

sessionId: "djdjdjdwwowowow338939",

sign: ...

В результате ожидается ответ:

{

ok:1,

errorCode:0,

result:

{

isAlive: 1 (или 0)

}

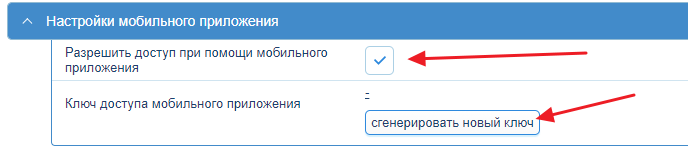
}

## Подпись запросов

Каждый запрос должен быть подписан.

Запрос подписывается передачей дополнительного параметра **sign**.

Для генерации этого поля нужно значение **secretKey**, который получается в Отмечалке. Чтобы его получить, нужно зайти в Отмечалку под Управляющим, перейти в меню Настройки – Тонкие настройки системы. Затем вверху страницы нажать на кнопку "Изменить настройки". На странице редактирования настроек нужно развернуть блок "Настройки мобильного приложения" и включить доступ через моб. приложение. При необходимости нужно сгенерировать новый ключ:



В каждом запросе обязательно должно присутствовать два параметра:

* sessionId – это id сессии, который получен в результате авторизации.
* sign – подпись запроса - хэш по методу base64 от SHA1 от строчки, которая является результатом конкатенации:
  + **sessionId**,
  + **secretKey**,
  + всех **параметров**, передаваемых в запросе, которые отсортированы по ключам по алфавиту.

Конкатенация производится через разделитель в виде нижнего подчеркивания "\_".

Метод SHA1 должен возвращать строку 20-character raw binary string.

**Пример запроса** и формирования подписи (значения sign).

Конкатенация строк сделана в стиле PHP через точку.

$sessionId = '123';

$secretKey = 'abc4567890';

$request = '{

parameters: {

name: "Направление 1",

shortName: "Напр 1"

},

sessionId: ...,

sign: ...

}';

$sign = base64\_encode(

sha1(

$sessionId . '\_' .

'name' . '\_' . 'Направление 1' . '\_' .

'shortName' . '\_' . 'Напр 1' . '\_'

. $secretKey,

true

));

**Функция на PHP, которая формирует корректное значение sign:**

**public** **static** **function** generateApiV1Sign($sessionId, $dataSet, $apiAccessSecretKey)

{

$sign = $sessionId . '\_';

**if**($dataSet != **null**)

{

**if**(is\_object($dataSet))

{

$dataSet = get\_object\_vars($dataSet);

}

$fields = array\_keys($dataSet);

sort($fields, SORT\_STRING);

**foreach** ($fields **as** $field)

{

$sign .= $field . '\_' . (**isset**($dataSet[$field]) ? $dataSet[$field] : '') . '\_';

}

}

$sign = base64\_encode(sha1($sign . $apiAccessSecretKey, true));

**return** $sign;

}

## Список возможных значений error\_code, список может расширяться по мере работы:

|  |  |
| --- | --- |
| **Значение** | **Описание** |
| 0 | Отсутствие ошибки |
| 1 | Ошибка авторизации |
| 2 | Ошибка валидации входных данных |
| 3 | Объект не найден |
| 4 | Необходимо совершать запрос методом POST |
| 5 | Неизвестная ошибка |
| 6 | Не правильная подпись запроса. Не корректное значение параметра sing. |
| 7 | Доступ к API отключен в настройках организации |
| 8 | Необходимо обновление ПО. Обычно означает, что текущая используемая версия API устарела и отключена. Нужно использовать новую версию. |
| 9 | Переданы не все обязательные поля. |
| 10 | Невозможно продать абонемент указанного типа |