Основные положения

Битрикс24 - программный продукт в виде облачного сервиса, а также коробочной версии, созданный компанией "1С-Битрикс".

Внимание! С 1 января 2021 года использование приложений Битрикс24.Маркет, REST API и вебхуков (локальные интеграции в разделе «Разработчикам») будет доступно только на коммерческих тарифах Битрикс24.

Внимание! В связи с прекращением поддержки интернет-браузера Internet Explorer 11 со стороны компании Microsoft **остановлена** поддержка браузера IE ниже 11 версии в платформе «1С-Битрикс: Управление сайтом» и в продукте «1С-Битрикс24»

Разработчики могут создавать собственные приложения или интеграции для Битрикс24, используя открытый REST API, который работает как с облачным, так и с коробочным Битрикс24, а также с "1С-Битрикс: Управление сайтом" начиная с версии 16.6.0.

> При использовании REST в коробочных продуктах необходимо вручную создать папку `/marketplace/` в корне сайта.

В документации дано справочное описание методов. Перед использованием справочника рекомендуется изучить курс Приложения Битрикс24, в котором даётся описание базовых понятий, рассматриваются варианты получения технического доступа к методам REST и предлагаются готовые примеры, включая раздел «Быстрый старт» как для создания тиражных решений, так и для решения частных задач интеграции и расширения функционала Битрикс24 по индивидуальным сценариям.

Примечание по использованию справочника: добавив к адресу любой страницы `#examples` можно быстро перейти к примеру, если он на ней есть.

Для ознакомления с API очень удобно использовать специально созданное приложение, которое представляет собой консоль разработчика и позволяет выполнять произвольные вызовы к REST

API прямо из приложения, ориентируясь на примеры кода в справочнике

Работа с REST возможна не только в облачных версиях Битрикс24, но и в коробочных редакциях. Начиная с версии 18.0.7 удаление модуля Rest из коробочных редакций невозможно.

В курсе **Бот платформа Битрикс24** есть описание собственного REST API для Бот-платформы.

История изменений

Апрель 2022

- Новый раздел Карточка звонка для внешней телефонии.
- Метод Объединение дубликатов
- Новый раздел Свойства отгрузки.
- Новый раздел Импорт.
- Новый раздел Управление привязками дел к сущностям CRM.

Март 2022

- Метод methods будет отключён с 1-го сентября.
- Метод socialnetwork.api.workgroup.list возвращает список групп.
- Метод socialnetwork.api.workgroup.get возвращает данные по рабочей группе.
- Метод disk.rights.getTasks позволяет получить список уровней доступов
- Обновлено описание метода disk.folder.uploadfile
- Обновлено описание метода disk.storage.uploadfile
- В методе sale.cashbox.handler.add в параметре SETTINGS добавлено поле **HTTP_VERSION**, обновлён пример.
- Новый раздел Службы доставки.

Январь 2022

- Сайты: новые методы для работы с сущностью Сайт: landing.site.addFolder, landing.site.getFolders, landing.site.getPreview, landing.site.markFolderDelete,

landing.site.markFolderUnDelete, landing.site.publicationFolder, landing.site.unPublicFolder, landing.site.updateFolder

- Сайты: новые методы для работы с сущностью Страница: landing.landing.move, landing.landing.resolveIdByPublicUrl
- Сайты: новые методы для работы с Блоками на Странице: landing.landing.favoriteBlock, landing.landing.unFavoriteBlock

2021

2020

2019

2018

2017

Информационные блоки

Документация по rest-методам информационных блоков размещены в документации по D7.

© «Битрикс», 2001-2008, «1С-

Права на методы REST

Доступ приложений (и вебхуков) к методам REST API регулируется через доступ к скоупам - модулям Битрикс24, реализующим те или иные методы REST.

В случае с локальными приложениями и вебхуками, пользователь с правами на администрирование Битрикс24 указывает нужные скоупы при добавлении приложения или создании вебхука.

В случае с установкой тиражного приложения, приложение запрашивает у пользователя разрешение на доступ к необходимым скоупам и пользователь либо дает запрашиваемые права, либо прерывает установку приложения.

На текущий момент реализованы следующие скоупы:

Значения	
bizproc	Бизнес-процессы
calendar	Календарь
call	Телефония (совершение звонков). В скоуп входят методы: voximplant.infocall.startwithsound voximplant.infocall.startwithtext
catalog	Торговый каталог
contact_center	Контакт-центр
crm	CRM
department	Структура компании
disk	Диск
documentgenerator	Генератор документов
entity	Хранилище данных
faceid	Распознавание лиц

im	Чат и уведомления
imbot	Создание и управление Чат-ботами
imopenlines	Открытые линии
intranet	Интранет
landing	Сайты
lists	Списки
log	Живая лента
mailservice	Почтовые сервисы
messageservice	Служба сообщений
mobile	Мобильное приложение
pay_system	Платёжные системы
placement	Встраивание приложений
pull	Pull&Push
rpa	Роботизация бизнеса
sale	Интернет-магазин
sonet_group	Рабочие группы
task	Задачи
telephony	Телефония
timeman	Учет рабочего времени
user	Пользователи
userconsent	Работа с соглашениями

Вызов методов с подтверждением

Некоторые методы требуют разрешения администратора портала на вызов. При вызове приложением такого метода администратор портала получит уведомление с предложением разрешить или запретить вызов, а приложение получит ошибку.

Внимание! Разрешение или запрет даются конкретному авторизационному токену, с которым происходит вызов метода. То есть, разрешение действует на время жизни токена и при получении следующего токена нужно получать новое разрешение.

```
GET https://portal.bitrix24.ru/rest/voximplant.user.get?
auth=fkp963yuv1ggkfbs5z3f5hy8lilm0iw6&USER_ID=1
HTTP/1.1 401 Unauthorized

{
    "error": "METHOD_CONFIRM_WAITING",
    "error_description": "Waiting for confirmation"
}
```

Вызов метода до получения подтверждения или ответа даст тот же ответ, но без запроса повторного подтверждения.

При подтверждении администратором разрешения или отказа произойдет вызов обработчика события OnAppMethodConfirm, с передачей ему результата подтверждения, а также токена, которому было выдано это разрешение:

```
array (
  'event' => 'ONAPPMETHODCONFIRM',
  'data' =>
  array (
    'TOKEN' => 'fkp963yuv1ggkfbs5z3f5hy8lilm0iw6',
    'METHOD' => 'voximplant.user.get',
    'CONFIRMED' => '1',
```

```
    'LANGUAGE_ID' => 'ru',
  ),
  'ts' => '1478790852',
  'auth' =>
  array (
    'domain' => 'portal.bitrix24.ru',
    'client_endpoint' =>
'https://portal.bitrix24.ru/rest/',
    'server_andpoint' => 'https://oauth.bitrix.info/rest/',
    'member_id' => '74ef8a46a75104de55d5d4a61b98ab6d',
    'application_token' =>
'c289487163b58658eae5e8b42eaf11b8',
  ),
```

Если администратор разрешил действие, приложение может использовать тот же самый авторизационный токен для работы с запрошенным методом:

```
GET https://portal.bitrix24.ru/rest/voximplant.user.get?
auth=fkp963yuv1ggkfbs5z3f5hy8lilm0iw6&USER_ID=1
HTTP/1.1 200 OK

{
    "result": [
        {
            "DEFAULT_LINE": null,
            "ID": "1",
            "INNER_NUMBER": null,
            "PHONE_ENABLED": "Y",
            "SIP_LOGIN": "****",
            "SIP_PASSWORD": "*****",
            "SIP_SERVER": "*****"
        }
    ]
}
```

В случае запрета вернется соответствующая ошибка:

```
GET https://portal.bitrix24.ru/rest/voximplant.user.get?
auth=fkp963yuv1ggkfbs5z3f5hy8lilm0iw6&USER_ID=1
HTTP/1.1 403 Forbidden

{
    "error": "METHOD_CONFIRM_DENIED",
    "error_description": "Method call denied"
}
```

Список методов, требующих подтверждения:

- voximplant.user.get

Как правильно выгружать большие объемы данных

Часто встает задача импорта каких либо сущностей с портала посредством rest. При этом при большом количестве сущностей прямой подход к задаче, с установкой фильтра и передачей в каждый следующий запрос `start = start+50`, не оптимальный. Так как, при использование `start >= 0` на каждый запрос выполняется еще и запрос подсчета количества элементов удовлетворяющих фильтру. Что при большом количестве элементов, попадающих в него, или при сложной фильтрации работает медленно.

Поэтому в случае если вам не нужно количество элементов (например вам нужно просто 10 последних записей) или вы делаете импорт всех записей по фильтру, передавайте `start = -1`.

Данный параметр отключит выполнение запроса подсчета количества элементов и сильно ускорит выборку.

Для выполнения импорта, при этом необходимо будет отсортировать записи по ID и добавить в фильтр условие ID > значения последнего элемента. И с каждым шагом увеличивать его значение. Значение же последнего элемента брать из последнего значения полученного результата.

Условием остановки импорта будет пустой ответ, или то, что в ответе элементов меньше 50.

Ниже приведен пример кода и сравнение времени выполнения с классическим подходом. Так при 2 387 743 лидах, с одинаковыми правами и фильтром, время выполнения уменьшилось с 49.9 секунд до 0,097 секунд.

Пример

```
$tokenID = 'XXXXXXXXXXXXXXXXXXXXX';
$host = 'XXXX.bitrix24.ru';
$user = 1;

/**
 * Начинаем с нуля или с какого то предыдущего шага
 */
$leadID = 0;
$finish = false;

while (!$finish)
{
   /**
    * Выполняем пока не заберем все данные, в этом случае
не стоит забывать и про задержку между хитами.
    * Либо каждый раз выбираем только 50, начиная с того
элемента, на котором остановилась прошлая итерация
   */

   $http = new \Bitrix\Main\Web\HttpClient();

   $http->setTimeout(5);
   $http->setStreamTimeout(50);

   $json = $http->post(

'https://'.$host.'/rest/'.$user.'/'.$tokenID.'/crm.lead.lis
t/',
      [
         'order' => ['ID' => 'ASC'],
         'filter' => ['>ID' => $leadID],
         'select' => ['ID', 'TITLE', 'DATE_CREATE'],
         'start' => -1
      ]
   );

   $result = \Bitrix\Main\Web\Json::decode($json);
   if (count($result['result']) > 0)
   {
      foreach ($result['result'] as $lead)
      {
         $leadID = $lead['ID'];
      }
      // Do something
   }
   else
   {
      $finish = true;
   }
}
```

```
/*
//Результаты выполнения rest запроса с использованием
count.
   Array
   (
      [result] => Array
      (
         [0] => Array()
         [1] => Array()
         .....
         [49] => Array()

      [next] => 50
      [total] => 2387743
      [time] => Array
      (
         [start] => 1581607213.4833
         [finish] => 1581607263.3997
         [duration] => 49.916450023651
         [processing] => 49.899916887283
         [date_start] => 2020-02-13T18:20:13+03:00
         [date_finish] => 2020-02-13T18:21:03+03:00
      )
)

//Результаты выполнения rest запроса без использования
count.

   Array
   (
      [result] => Array
      (
         [0] => Array()
         [1] => Array()
         .....
         [1] => Array()

      [total] => 0
      [time] => Array
      (
         [start] => 1581609136.3857
         [finish] => 1581609136.4835
         [duration] => 0.097883939743042
         [processing] => 0.068500995635986
         [date_start] => 2020-02-13T18:52:16+03:00
         [date_finish] => 2020-02-13T18:52:16+03:00
      )
)
*/
```

© «Битрикс», 2001-2008, «1С-

Особенности вызовов REST при изменении адреса Битрикс24

Как вы, наверное, знаете, новые облачные Битрикс24 создаются по сгенерированным адресам вида b24-xxx.bitrix24.yy. И в дальнейшем, пользователи имеют возможность изменить этот адрес в любой момент (с некоторыми ограничениями в зависимости от используемого тарифного плана).

Почему это важно иметь в виду? Если ваше приложение делает вызов REST Битрикс24 и при этом использует сохраненный на стороне приложения адрес, то может возникнуть ситуация, когда этот адрес уже не актуален.

В случае обращения по неактуальному адресу, Битрикс24 делает редирект на новый, но такой редирект нужно корректно обрабатывать в своем коде.

Скорее всего, при использовании GET-параметров в вызовах REST, вы просто ничего не заметите, но вот с POST-запросами все несколько сложнее.

В частности, если вы используете PHP и curl, то в зависимости от настроек, POST-запрос при редиректе может «магическим образом» превратиться в GET-запрос, при этом параметры, передававшиеся в POST-запросе попросту теряются.

Существуют два подхода:

1. Выполняя POST-запрос запретить редирект, получить статус запроса 302, взять из результата новый адрес и повторить POST-запрос, но уже по новому адресу. Например, так можно делать в приложениях на Python:

```
response = requests.post(url, allow_redirects=False)
if response.status_code == 302:
    response =
requests.post(response.headers['Location'])
```

2. Использовать опцию `curl_setopt($ch, CURLOPT_POSTREDIR, 3)`, которая позволит обработать ситуацию с редиректом.

А если вы будете использовать для работы с REST наши SDK на PHP и Python, то там мы эти особенности работы уже учли.

Работа с SDK CRest в контексте пользователя

Стандартно библиотека CRest работает под пользователем, установившим приложение. Но бывают ситуации когда необходимо выполнить запрос к порталу Битрикс24 под пользователем чьи токены пришли на страницу в POST. Например, когда пользователь открыл встройку вашего приложения. Для этого можно пронаследовать методы класса CRest:

```
require_once(__DIR__ . '/crest.php');
class CRestCurrent extends CRest
{
   protected static $dataExt = [];
   protected static function getSettingData()
   {
      $return =
static::expandData(file_get_contents(__DIR__ .
'/settings.json'));
      if(is_array($return))
      {
         if(!empty(static::$dataExt))
         {
            $return['access_token'] =
htmlspecialchars(static::$dataExt['AUTH_ID']);
            $return['domain'] =
htmlspecialchars(static::$dataExt['DOMAIN']);
            $return['refresh_token'] =
htmlspecialchars(static::$dataExt['REFRESH_ID']);
            $return['application_token'] =
htmlspecialchars(static::$dataExt['APP_SID']);
         }
         else
         {
            $return['access_token'] =
htmlspecialchars($_REQUEST['AUTH_ID']);
            $return['domain'] =
htmlspecialchars($_REQUEST['DOMAIN']);
            $return['refresh_token'] =
htmlspecialchars($_REQUEST['REFRESH_ID']);
```

```
            $return['application_token'] =
htmlspecialchars($_REQUEST['APP_SID']);
         }

      }

      return $return;
   }

   public static function setDataExt($data)
   {
      static::$dataExt = $data;
   }
}
```

После этого вы можете подключать файл с новым классом в своём коде и использовать методы работы текущего пользователя, когда на странице есть токены.
Проверить работу можно вызвав метод:

```
$result = CRestCurrent::call('user.current');

echo '<pre>';
 print_r($result);
echo '</pre>';
```

Как использовать примеры

Описание

Использование REST API позволяет достаточно просто решать многие задачи интеграции внешних источников или систем с Битрикс24, переносить в Битрикс24 данные и многое другое. Мы постарались дать примеры для наиболее частых задач, которые вы можете использовать в качестве заготовок и модифицировать под свои нужды.

Все примеры в разделе написаны с использованием PHP, поэтому вам нужно размещать этот код на доступном для вас сервере с соблюдением следующих технических условий:

1. Примеры используют модуль cURL для выполнения REST-запросов. Узнайте, как включить модуль cURL на своем сервере.
2. На вашем веб-сервере должен быть установлен валидный SSL-сертификат.
3. Примеры используют базовый [dw]SDK[/dw][di]В начале файла **crest.php** есть обозначение версии: @version 1.1, 1.2 ... 1.10. Рекомендуется периодически скачивать файл и проверять наличие новой версии SDK[/di] в виде класса *CRest* для выполнения запросов и продления токенов авторизации. Возьмите код класса, внесите необходимые правки, связанные с авторизацией в файл **settings.php** и разместите на своем сервере, вставив нужный вам пример из документации.
4. При возникновении проблем с работой SDK вы можете открыть через браузер файл **checkserver.php**, который произведёт минимальную проверку настроек сервера для работы класса CRest.

Если в проекте используется класс CRest и кодировка отличается от utf8, то необходимо сделать 2 дополнительных действия:

1. Открыть файлы из архива и изменить их кодировку на необходимую.
2. В файле **settings.php** объявить константу `C_REST_CURRENT_ENCODING`. Например, если проект в кодировке windows-1251 константа выглядеть должна так: `define('C_REST_CURRENT_ENCODING','windows-1251');`.

Вариант 1. Вызов REST с использованием входящего вебхука

1. Укажите URL вебхука в define `C_REST_WEB_HOOK_URL` в файле **settings.php**:

```
<?
define('C_REST_WEB_HOOK_URL','https://xxx.bitrix24.ru/rest/1/douasdqkjxgc3mgc1/');
```

2. Вставьте текст примера в файл **index.php**:

```
<?
require_once('crest.php');

// put an example below
echo '<PRE>';
print_r(CRest::call(
   'crm.lead.add',
   [
      'fields' =>[
      'TITLE' => 'Название лида',//Заголовок*[string]
      'NAME' => 'Имя',//Имя[string]
      'LAST_NAME' => 'Фамилия',//Фамилия[string]
      ]
   ])
);

echo '</PRE>';
```

3. Укажите URL к примеру в адресной строке браузера https://mydomain.xxx/index.php, чтобы увидеть результат работы примера.

Вариант 2. Вызов REST из локального приложения

1. Вставьте текст примера в файл **index.php**:

```
<?
require_once('crest.php');

// put an example below
echo '<PRE>';
print_r(CRest::call(
   'crm.lead.add',
   [
      'fields' =>[
      'TITLE' => 'Название лида',//Заголовок*[string]
      'NAME' => 'Имя',//Имя[string]
      'LAST_NAME' => 'Фамилия',//Фамилия[string]
      ]
   ])
);

echo '</PRE>';
```

2. В карточке локального приложения укажите URL своего приложения https://mydomain.xxx/index.php и URL скрипта установки https://mydomain.xxx/install.php.
3. Укажите значения параметров **client_id** и **client_secret** для авторизации OAuth 2.0 в define `C_REST_CLIENT_ID` и `C_REST_CLIENT_SECRET` в файле **settings.php**, взяв эти значения из карточки локального приложения.

```
<?
require_once('crest.php');

// put an example below
echo '<PRE>';
print_r(CRest::call(
   'crm.lead.add',
   [
      'fields' =>[
      'TITLE' => 'Название лида',//Заголовок*[string]
      'NAME' => 'Имя',//Имя[string]
      'LAST_NAME' => 'Фамилия',//Фамилия[string]
      ]
   ])
);

echo '</PRE>';
```

4. В списке локальных приложений нажмите правой кнопкой мыши на своё локальное приложение и выберите пункт "Переустановить". Это нужно чтобы корректно сработал **install.php** после того, как вы вставили корректные значения C_REST_CLIENT_ID и C_REST_CLIENT_SECRET.
5. После установки вы увидите результат работы примера. Если пример демонстрирует встраивание виджетов в другие инструменты Битрикс24, необходимо перейти в эти инструменты.

Вариант 3. Вызов REST из тиражного приложения

1. Вставьте текст примера в файл **index.php**

```
<?
require_once('crest.php');

// put an example below
echo '<PRE>';
print_r(CRest::call(
   'crm.lead.add',
   [
      'fields' =>[
      'TITLE' => 'Название лида',//Заголовок*[string]
      'NAME' => 'Имя',//Имя[string]
      'LAST_NAME' => 'Фамилия',//Фамилия[string]
      ]
   ])
);

echo '</PRE>';
```

2. Добавьте тиражное приложение в партнерском кабинете для получения client_id и client_secret и при сохранении приложения.
3. Укажите значения параметров **client_id** и **client_secret** для авторизации OAuth 2.0 в define `C_REST_CLIENT_ID` и `C_REST_CLIENT_SECRET` в файле **settings.php**.

```
<?
require_once('crest.php');

// put an example below
echo '<PRE>';
```

```
print_r(CRest::call(
   'crm.lead.add',
   [
      'fields' =>[
      'TITLE' => 'Название лида',//Заголовок*[string]
      'NAME' => 'Имя',//Имя[string]
      'LAST_NAME' => 'Фамилия',//Фамилия[string]
      ]
   ])
);

echo '</PRE>';
```

4. В карточке приложения добавьте версию и укажите URL своего приложения https://mydomain.xxx/index.php и URL скрипта установки https://mydomain.xxx/install.php в карточке версии.
5. После сохранения версии откройте карточку версии и, нажав на ссылку "Установить на своем Битрикс24", установите свое приложение на любой доступный вам Битрикс24.
6. После установки вы увидите результат работы примера (в случае, если пример демонстрирует встраивание виджетов в другие инструменты Битрикс24, необходимо перейти в эти инструменты).
7. Для реального тиражного приложения необходимо пронаследовать класс CRest, переопределив методы *getSettingData/setSettingData*, которые занимается получением/сохранением токенов авторизации в текстовый файл. Эти методы не предназначены для эксплуатации приложения на нескольких Битрикс24 одновременно.

События

Интерфейс REST позволяет устанавливать свои обработчики серверных событий.

Обработчиком служит URL, который будет вызван после того, как пользователь произведет запрошенное действие на портале Битрикс24, на который установлено приложение. Поскольку запросы будут идти с серверов Битрикс, то любой URL должен быть доступен для GET/POST запросов извне.

Обработчик получает на вход следующие данные:

Поле запроса	Описание
event	Имя сработавшего события
data	Массив данных события. Зависит от события.
auth	Набор данных авторизации.

Обработчик может:

- использовать полученные данные авторизации для выполнения запросов к REST API.
- быть установлен только пользователем с правами администрирования портала.

Примечание.Если обработчику события требуется обращение к Rest API для работы с данными, то настоятельно рекомендуется использовать для этого данные авторизации сохраненные на стороне приложения, а не опираться на получение **access_token**, так как его может и не быть.

При установке события можно указать пользователя, под которым будет авторизовываться обработчик события. По умолчанию обработчику события будет выдана авторизация пользователя, действия которого привели к срабатыванию события.

Обработчик события будет вызван не сразу после срабатывания события, а через некоторое время, зависящее от нагрузки.

Список доступных событий можно получить при помощи REST-метода events.

Установка обработчика события производится при помощи REST-метода event.bind или при помощи функции BX24.callBind js-библиотеки. Получение списка зарегистрированных обработчиков событий производится при помощи REST-метода events.get

Снятие зарегистрированного обработчка события производится при помощи REST-метода event.unbind или при помощи функции BX24.callUnbind js-библиотеки.

Для доступа к каждому событию при регистрации версии приложения должно быть запрошено соответствующее событию право доступа.

Приложение может установить произвольное количество обработчиков одного и того же события, но все обработчики должны быть установлены с авторизацией различных пользователей. Кроме того, вызов обработчика события может зависеть от доступа пользователя, чья авторизация будет выдана обработчику.

Имена событий регистронезависимы.

> **Внимание!** При обновлении версии зарегистрированные обработчики по умолчанию сохранятся, и при необходимости могут быть удалены обработчиком установки новой версии приложения.

Пример

Событие установки приложения

```
array(
  'event' => 'ONAPPINSTALL',
  'data' => array(
    'VERSION' => '1',
    'LANGUAGE_ID' => 'ru',
  ),
  'ts' => '1466439714',
  'auth' => array(
    'access_token' => 's6p6eclrvim6da22ft9ch94ekreb52lv',
    'expires_in' => '3600',
    'scope' => 'entity,im',
    'domain' => 'portal.bitrix24.com',
    'server_endpoint' => 'https://oauth.bitrix.info/rest/',
    'status' => 'F',
    'client_endpoint' =>
'https://portal.bitrix24.ru/rest/',
    'member_id' => 'a223c6b3710f85df22e9377d6c4f7553',
    'refresh_token' => '4s386p3q0tr8dy89xvmt96234v3dljg8',
    'application_token' =>
'51856fefc120afa4b628cc82d3935cce',
  ),
)
```

Все события Rest

Событие	Вызывается
События приложения	
OnAppInstall	Событие на установку приложения
OnAppUpdate	Событие на обновление приложения.
OnAppUninstall	при удалении приложения
OnAppPayment	при оплате приложения
OnAppMethodConfirm	при получении решения администратора портала по запросу на использование

	методов, требующих подтверждения
OnAppTest	
События бот-платформы	
OnImBotMessageAdd	Событие на получение чат-ботом сообщения.
OnImBotMessageUpdate	Событие на обновление сообщения чат-ботом.
OnImBotMessageDelete	Событие на удаление сообщения чат-ботом.
OnImBotJoinChat	Событие на получение информации чат-ботом о включении его в чат (или личную переписку).
OnImBotDelete	Событие на удаление чат-бота.
OnImCommandAdd	Событие на получение чат-ботом команды.
События CRM	
Счета	
onCrmInvoiceAdd	при создании счёта
onCrmInvoiceDelete	при удалении счёта
onCrmInvoiceSetStatus	при изменении статуса счёта
onCrmInvoiceUpdate	при обновлении счёта
onCrmInvoiceUserFieldAdd	при добавлении пользовательского поля счёта

onCrmInvoiceUserFieldUpdate	при изменении пользовательского поля счёта
onCrmInvoiceUserFieldDelete	при удалении пользовательского поля счёта
onCrmInvoiceUserFieldSetEnumValues	при изменении набора значений для пользовательского поля счёта списочного типа
onCrmInvoiceRecurringAdd	при создании нового регулярного счета
onCrmInvoiceRecurringUpdate	при обновлении регулярного счета
onCrmInvoiceRecurringDelete	при удалении регулярного счета
onCrmInvoiceRecurringExpose	при выставлении нового счет из регулярного счета
Товары	
onCrmProductAdd	при создании товара
onCrmProductUpdate	при обновлении товара
onCrmProductDelete	при удалении товара
onCrmProductPropertyAdd	при создании свойства товара
onCrmProductPropertyUpdate	при обновлении свойства товара
onCrmProductPropertyDelete	при удалении свойства товар
onCrmProductSectionAdd	после добавлении раздела товара
onCrmProductSectionUpdate	после изменении раздела товара

onCrmProductSectionDelete	после удалении раздела товара
Лиды	
onCrmLeadAdd	при создании лида
onCrmLeadUpdate	при обновлении лида
onCrmLeadDelete	при удалении лида
onCrmLeadUserFieldAdd	при добавлении пользовательского поля лида
onCrmLeadUserFieldUpdate	при изменении пользовательского поля лида
onCrmLeadUserFieldDelete	при удалении пользовательского поля лида
onCrmLeadUserFieldSetEnumValues	при изменении набора значений для пользовательского поля лида списочного типа
Сделки	
onCrmDealAdd	при создании сделки
onCrmDealUpdate	при обновлении сделки
onCrmDealDelete	при удалении сделки
onCrmDealUserFieldAdd	при добавлении пользовательского поля сделки
onCrmDealUserFieldUpdate	при изменении пользовательского поля сделки
onCrmDealUserFieldDelete	при удалении

	пользовательского поля сделки
onCrmDealUserFieldSetEnumValues	при изменении набора значений для пользовательского поля сделки списочного типа
onCrmDealRecurringAdd	при создании новой регулярной сделки
onCrmDealRecurringUpdate	при обновлении регулярной сделки
onCrmDealRecurringDelete	при удалении регулярной сделки
onCrmDealRecurringExpose	при создании новой сделки и регулярной сделки
Компании	
onCrmCompanyAdd	при создании компании
onCrmCompanyUpdate	при обновлении компании
onCrmCompanyDelete	при удалении компании
onCrmCompanyUserFieldAdd	при добавлении пользовательского поля компании
onCrmCompanyUserFieldUpdate	при изменении пользовательского поля компании
onCrmCompanyUserFieldDelete	при удалении пользовательского поля компании
onCrmCompanyUserFieldSetEnumValues	при изменении набора значений для пользовательского поля компании списочного типа

Контакты	
onCrmContactAdd	при создании контакта
onCrmContactUpdate	при обновлении контакта
onCrmContactDelete	при удалении контакта
onCrmContactUserFieldAdd	при добавлении пользовательского поля
onCrmContactUserFieldUpdate	при изменении пользовательского поля контакта
onCrmContactUserFieldDelete	при удалении пользовательского поля контакта
onCrmContactUserFieldSetEnumValues	при изменении набора значений для пользовательского поля контакта списочного типа
Валюты	
onCrmCurrencyAdd	после создании валюты
onCrmCurrencyUpdate	после обновлении валюты
onCrmCurrencyDelete	после удалении валюты
Дела	
onCrmActivityAdd	при создании дела
onCrmActivityUpdate	при обновлении дела
onCrmActivityDelete	при удалении дела
Коммерческие предложения	

onCrmQuoteAdd	при создании предложения
onCrmQuoteUpdate	при обновлении предложения
onCrmQuoteDelete	при удалении предложения
onCrmQuoteUserFieldAdd	при добавлении пользовательского поля предложения
onCrmQuoteUserFieldDelete	при удалении пользовательского поля предложения
onCrmQuoteUserFieldUpdate	при изменении пользовательского поля предложения
onCrmQuoteUserFieldSetEnumValues	при изменении набора значений для пользовательского поля предложения списочного типа
Реквизиты	
onCrmRequisiteAdd	При добавлении реквизита
onCrmRequisiteUpdate	При изменении реквизита
onCrmRequisiteDelete	При удалении реквизита
onCrmRequisiteUserFieldAdd	при добавлении пользовательского поля реквизита
onCrmRequisiteUserFieldUpdate	при изменении пользовательского поля реквизита
onCrmRequisiteUserFieldDelete	при удалении пользовательского поля реквизита

onCrmRequisiteUserFieldSetEnumValues	при изменении набора значений для пользовательского поля реквизита списочного типа
onCrmBankDetailAdd	При добавлении банковского реквизита
onCrmBankDetailUpdate	При обновлении банковского реквизита
onCrmBankDetailDelete	При удалении банковского реквизита
onCrmAddressRegister	При добавлении адреса
onCrmAddressUnregister	При удалении адреса
Единицы измерений	
onCrmMeasureAdd	При добавлении единицы измерения
onCrmMeasureUpdate	При обновлении единицы измерения
onCrmMeasureDelete	При удалении единицы измерения
Таймлайн	
onCrmTimelineCommentAdd	при добавлении нового комментария в таймлайне
onCrmTimelineCommentUpdate	при обновлении нового комментария в таймлайне
onCrmTimelineCommentDelete	при удалении нового комментария в таймлайне
События Телефонии	
OnVoximplantCallInit	при инициализации звонка (с

	поступлении или начале исходящего звонка).
OnVoximplantCallStart	при начале разговора (ответе оператора при входящем и ответе абонента при исходящем).
OnVoximplantCallEnd	при окончании разговора (запись в историю).
OnExternalCallStart	вызывается, когда пользователь нажимает на телефонный номер в объекта CRM для совершения исходящего звонка.
OnExternalCallBackStart	при заполнении crm-формы обратного звонка.
События Задач	
OnTaskAdd	при добавлении задачи.
OnTaskUpdate	при обновлении задачи.
OnTaskDelete	при удалении задачи.
OnTaskCommentAdd	при добавлении комментария к задаче.
OnTaskCommentUpdate	при проведении операций на комментарием к задаче.
OnTaskCommentDelete	при удалении комментария к задаче.
События Живой ленты	
ONLIVEFEEDPOSTADD	Прокси к событию PHP OnAfterSocNetLogAdd.
ONLIVEFEEDPOSTUPDATE	Прокси к событию PHP

	OnAfterSocNetLogUpdate.
ONLIVEFEEDPOSTDELETE	Прокси к событию PHP OnSocNetLogDelete.
События Рабочих групп	
ONSONETGROUPADD	Событие вызывается после добавления новой рабочей группы. Прокси к событию OnSocNetGroupAdd.
ONSONETGROUPUPDATE	Событие вызывается после изменения рабочей группы. Прокси к событию onSocnetGroupUpdate
ONSONETGROUPDELETE	Вызывается в момент удаления рабочей группы. Прокси к событию OnSocNetGroupDelete
ONSONETGROUPSUBJECTADD	Событие вызывается после создания темы рабочих групп Прокси к событию OnSocNetGroupSubjectAdd.
ONSONETGROUPSUBJECTUPDATE	Событие вызывается после изменения темы рабочих групп. Прокси к событию OnSocNetGroupSubjectUpdate
ONSONETGROUPSUBJECTDELETE	Вызывается перед удалением темы рабочих групп.. Прокси к событию OnSocNetGroupSubjectDelete
События Интернет магазина	
OnSaleOrderSaved	Происходит в конце сохранения заказа, когда заказ и все связанные сущности уже сохранены.

OnSaleBeforeOrderDelete	Вызывается перед удалением заказа.
On[dw]<сущность>[/dw][di] где <сущность> - это: - свойство заказа `PropertyValue`; - оплата `Payment`; - отгрузка `Shipment`; - заказ `Order`. [/di]EntitySaved	Происходит непосредственно после сохранения сущности системы заказов.
On[dw]<сущность>[/dw][di] где <сущность> - это: - свойство заказа `PropertyValue`; - оплата `Payment`; - отгрузка `Shipment`; - заказ `Order`. [/di]Deleted	Вызывается при непосредственном удалении сущности из базы.
События пользователя	
OnUserAdd	При добавлении пользовател в Битрикс24
События Коннектора для внешних мессе	
OnImConnectorLineDelete	Событие удаления открытой линии.
OnImConnectorMessageAdd	Событие нового сообщения и ОЛ.
OnImConnectorMessageUpdate	Событие изменения сообщения из ОЛ.
OnImConnectorMessageDelete	Событие удаления сообщения из ОЛ.

OnImConnectorStatusDelete	Событие удалении конкретного подключения канала приложения конкретной линии.
События Календаря	
OnCalendarEntryAdd	при добавлении события.
OnCalendarEntryUpdate	при изменении события.
OnCalendarEntryDelete	при удалении события.
OnCalendarSectionAdd	при добавлении [dw]секции календаря[/dw] [di]CALENDAR_SECTION_ID. Подробнее[/di]. Также буде вызываться при добавлении ресурса.
OnCalendarSectionUpdate	при изменении секции/ ресурса.
OnCalendarSectionDelete	при удалении секции/ресурса

Как проверить свой обработчик для обработки событий Битрикс24

После регистрирации обработчика ONAPPTEST вручную вызывается метод **event.test**. Это вызывает срабатывание указанного события и позволяет убедиться, что обработчик действительно в состоянии принимать данные о событиях.

Шаг 1

Создайте файл **handler.php** на своём сервере. Убедитесь что он доступен из интернета. Рядом с файлом создайте папку `\log`.
Код файла handler.php:

```
<?
file_put_contents(
   __DIR__ . '/log/' . time() . '.txt',
   var_export($_REQUEST, true)
);
```

шаг 2

Зарегистрируйте событие, указав в поле `handler` путь до файла созданного в шаге 1.

```
<?
$eventBind = CRest::call(
   'event.bind',
   [
```

```
      'event' => 'ONAPPTEST',
      'handler' => 'https://example.com/handler.php'
   ]
);
if($eventBind['result'])
{
   echo 'event bind successful';
}
?>
```

Шаг 3

Заставьте событие сработать вызовом метода с произвольными данными:

```
<?
$result = CRest::call(
   'event.test',
   [
      'any' => 'data'
   ]
);
if($result['result'])
{
   echo 'successful';
}
?>
```

Результат

При успешном вызове в папке `\log` создаётся файл со стандартными данными для событий:

```
array (
    'event' => 'ONAPPTEST',
    'data' =>
    array (
      'QUERY' =>
      array (
        'any' => 'data',
      ),
      'LANGUAGE_ID' => 'en',
    ),
    'ts' => '1573120286',
```

```
    'auth' => array (...)
)
```

BX24.init

```
void BX24.init(Function callback)
```

Добавляет в список обработчик события "библиотека готова к работе". При инициализации приложения библиотека запрашивает данные для работы у родительского фрейма. Некоторые действия могут быть совершены только после получения этих данных (например, работа с настройками приложения, с правами текущего пользователя, отправка запросов к REST и т.д.).

Пример

```
BX24.init(function(){
        console.log('Инициализация
завершена!', BX24.isAdmin());
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.install

```
void BX24.install(Function callback)

void BX24.install(String callback)
```

Возможность установить обработчик события "приложение запускается первый раз для текущего пользователя". Событие возникает сразу после события "библиотека готова к работе", но до запуска обработчиков, установленных в BX24.init. Обработчик должен сигнализировать о завершении процесса настройки вызовом функции BX24.installFinish.

Если в качестве обработчика передана строка, то она считается ссылкой на js-файл, который нужно загрузить и выполнить при срабатывании события. В любом случае, если текущий пользователь первый раз запускает приложение, и установлен хоть один обработчик этого события, BX24.init сработает только после вызова BX24.installFinish.

Пример

```
BX24.install(function(){
        BX24.callMethod('user.current', {},
function(res){
                alert('Приложение Hello
World приветствует вас, ' + res.data().NAME
+ '!');
                BX24.installFinish();
        });
});
```

BX24.installFinish

```
void BX24.installFinish()
```

Функция, сигнализирующая об окончании работы инсталлятора или настройщика приложения.

Если функция вызывается на этапе запуска инсталлятора, то произойдет перезагрузка страницы и запуск приложения. Если на этапе настройки - запуск обработчиков BX24.init. В остальных случая никакого эффекта от вызова не будет.

© «Битрикс», 2001-2008, «1С-

BX24.getAuth

```
Boolean|Object BX24.getAuth()
```

Получение текущих данных для авторизации через OAuth 2.0. Возвращается объект вида

```
{access_token: авторизационный_код, expires_in:
дата_истечения_кода, refresh_token:
токен_продления_авторизации, domain: текущий_домен,
member_id: идентификатор_портала}
```

Дата истечения передается в виде объекта Date.

Работает только после BX24.init. В случае вызова до инициализации приложения или после истечения кода вернет false. При истечении кода новый автоматически генерируется при следующем вызове **BX24.callMethod** или **BX24.refreshAuth**.

Пример

```
console.log(BX24.getAuth());
```

BX24.refreshAuth

```
void BX24.refreshAuth([Function callback])
```

Принудительное обновление ключа авторизации. Функция-обработчик callback получит на вход объект, аналогичный BX24.getAuth(). Работает только после BX24.init.

Пример

```
console.log(BX24.refreshAuth());
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.callMethod

Отправка запроса:

```
void BX24.callMethod(
        String method,
        Object params[,
        Function callback]
);
```

Метод вызывает указанный метод REST-сервиса с указанными параметрам. В качестве параметров метода выступает объект **params**, представляющий собой ассоциативный массив, преобразуемый в строку POST-запроса. Значениями элементов массива могут быть строки или ссылки на DOM-элементы полей формы (см. отправку файлов ниже).

В случае вызова до BX24.init выполнение запроса будет отложено.

Пример

```
BX24.callMethod('user.get', {ID: 10},
function(res){
        if(res.data())
        {
                var user = res.data()[0];
                if(!!user)
                        alert('Пользователя
№' + user.ID + ' зовут ' + user.NAME);
        }
});
```

Обработка результата запроса

Обработчиком результата запроса является функция, получающая на вход объект **ajaxResult** такого вида:

ajaxResult.prototype.data = function() - функция, возвращающая ответ метода в виде массива, объекта или скаляра. См. описание конкретных методов.

Boolean|String ajaxResult.prototype.error = function() - функция, возвращающая ошибку при ее наличии и false при отсутствии.

Некоторые методы могут возвращать большие массивы данных. В этом случае данные возвращаются частями по 50 элементов (методы могут перекрывать это значение, см. описание конкретных методов). Для работы со списками объект ajaxResult имеет следующие методы:

- **Boolean ajaxResult.prototype.more = function()** - функция, возвращающая флаг "есть еще данные для загрузки". Применимо, если метод возвращает список данных.
- **Integer ajaxResult.prototype.total = function()** - функция, возвращающая общее количество доступных записей. Применимо, если метод возвращает список данных.
- **Boolean|XMLHttpRequest|XDomainRequest ajaxResult.prototype.next = function([Function cb])** - функция, запрашивающая следующую страницу данных. По умолчанию будет вызван тот же самый обработчик результата запроса. Переопределить это поведение можно, установив значение параметра cb.

Пример получения постраничного списка пользователей

```
BX24.callMethod('user.get',
{sort:'ID',order:'ASC'}, function(result){
    if(result.error())
    {
        alert('Ошибка запроса: ' +
```

```
result.error());
    }
    else
    {
        console.log(result.data());
        if(result.more())
            result.next();
    }
});
```

BX24.callBatch

В некоторых случаях возникает необходимость отправить несколько запросов подряд. Например, при создании необходимых сущностей в процессе инсталляции приложения. Для оптимизации процесса можно использовать пакетное выполнение запросов.

```
void BX24.callBatch(
        Object|Array calls,
        [Function callback[,
        Boolean bHaltOnError = false]]
);
```

В случае вызова до BX24.init выполнение запроса будет отложено.

Функция отправляет пакет запросов к REST-сервису.

Параметры

Параметр	Описание
calls	Обычный или ассоциативный массив (объект) с запросами. Каждый запрос представляет собой либо массив [имя_метода, параметры_метода], либо объект {method: имя_метода, params: параметры_метода}. В параметрах методов можно использовать макросы, позволяющие получить доступ к результатам предыдущих запросов текущего пакета. Макрос можно составить примерно так: $result[идентификатор_запроса][поле_ответа], где идентификатором запроса служит его ключ в массиве пакета запросов.

callback	Функция-обработчик результата пакетного запроса. На вход получит массив или ассоциативный массив (объект) объектов ajaxResult с ключами, соответствующими ключам из пакета запросов.
bHaltOnError	Флаг "прерывать исполнение пакета в при возникновении ошибки". По умолчанию - false (не прерывать).

Пример

```
BX24.callBatch({
        get_user: ['user.current', {}],
        get_department: {
                method: 'department.get',
                params: {
                        ID:
'$result[get_user][UF_DEPARTMENT]'
                }
        }
}, function(result)
{
        var l =
result.get_department.data().length;
        var str = 'Текущий пользователь ' +
result.get_user.data().NAME + ' ' +
result.get_user.data().LAST_NAME + '
приписан к подразделени' + (l > 1 ? 'ям ' :
'ю ');

        for(var i = 0; i < l; i++)
        {
                str += i == 0 ? '' : ', ';
                str +=
result.get_department.data()[i].NAME;
```

```
        }

        alert(str);
});
```

Смотри также

- batch

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Обработка файлов

Методы REST-сервиса получают файлы в виде строки, закодированной в base64. Также можно отправить обычный массив, первым элементом которого будет имя файла, вторым - содержимое в base64.

В случае полностью клиентского приложения можно либо воспользоваться объектом FileReader, либо просто отдать в качестве значения поля запроса ссылку на элемент формы типа "файл" (`<input type="file">`).

Пример

```
<input type="file" id="testfile"><br />
<span onclick="sendInputFile()">send file
from input</span><br />
<span onclick="sendStaticFile()">send static
file</span><br />
<script type="text/javascript">
function sendInputFile()
{
   BX24.callMethod('entity.item.add', {
      'ENTITY': 'menu',
      'NAME': Math.random(),
      'DETAIL_PICTURE':
document.getElementById('testfile')
   }, function(){
      alert('Finished!');
   });
```

```
}
/*
POST
https://my.bitrix24.com/rest/entity.item.add
.json HTTP/1.1
Host: my.bitrix24.com
Content-Length: 186
Content-Type: text/plain; charset=UTF-8

auth=6a8c365cb010ba42bd5b0f6ae803f47c&ENTITY
=menu&NAME=0.2630483947652045&DETAIL_PICTURE
[0]=1.gif&DETAIL_PICTURE[1]=R0lGODlhAQABAIAA
AP%2F%2F%2FwAAACH5BAEAAAAALAAAAAABAAEAAAICRA
EAOw%3D%3D
*/

function sendStaticFile()
 {
   BX24.callMethod('entity.item.add', {
      'ENTITY': 'menu',
      'NAME': '1.gif',
      'DETAIL_PICTURE': ['1.gif',
'R0lGODlhAQABAIAAAP///wAAACH5BAEAAAAALAAAAAA
BAAEAAAICRAEAOw==']
    }, function(){
        alert('Finished!');
     });
 }

/*
POST
https://my.bitrix24.com/rest/entity.item.add
.json HTTP/1.1
Host: my.bitrix24.com
Content-Length: 173
Content-Type: text/plain; charset=UTF-8
```

```
auth=6a8c365cb010ba42bd5b0f6ae803f47c&ENTITY
=menu&NAME=1.gif&DETAIL_PICTURE[0]=1.gif&DET
AIL_PICTURE[1]=R0lGODlhAQABAIAAAP%2F%2F%2FwA
AACH5BAEAAAAALAAAAAABAAEAAAICRAEAOw%3D%3D
*/
</script>
```

Для методов CRM при добавлении картинки для товара вместо

```
'DETAIL_PICTURE': ['1.gif',
'R0lGODlhAQABAIAAAP///wAAACH5BAEAAAAALAAAAAABAAEAAAICRAE
AOw==']
```

используйте

```
"PREVIEW_PICTURE": {"fileData": ["1.gif",
"R0lGODlhAQABAIAAAP///wAAACH5BAEAAAAALAAAAAABAAEAAAICRAE
AOw=="]}
```

Такая особенность вызвана тем, что в CRM поддерживается удаление файлов.

BX24.callBind

```
BX24.callBind(
        String event,
        String handler[,
        Integer auth_type[,
        Function callback]]
);
```

Интерфейс для метода event.bind, регистрирующего новый обработчик события.

> Работает только при авторизации под пользователем с правами администрирования портала.

Параметры

Параметр	Описание
event	Имя события. Обязательный.
handler	Ссылка на обработчик события. Обязательный.
auth_type	Идентификатор пользователя, под которым авторизуется обработчик события. Не обязательный. По умолчанию будет использоваться авторизация пользователя, действия которого привели к срабатыванию события.
callback	функция-обработчик результата вызова метода. Не обязательный.

Пример:

```
BX24.callBind('OnAppUninstall',
'http://www.my-domain.ru/handler/');
```

BX24.callUnbind

```
BX24.callUnbind(
        String event,
        String handler[,
        Integer auth_type[,
        Function callback]]
);
```

Интерфейс для метода event.unbind, удаляющего зарегистрированный обработчик события.

> **Примечание**: Работает только при авторизации под пользователем с правами администрирования портала.

Параметры

Параметр	Описание
event	Имя события. Не обязательный.
handler	Ссылка на обработчик события. Не обязательный.
auth_type	Идентификатор пользователя, под которым авторизуется обработчик события. Не обязательный. **Примечание**: если требуется удалить обработчики события, установленные с пустым *auth_type* (с авторизацией от имени пользователя, вызвавшего событие), но оставить остальные обработчики, указывайте *auth_type=0* или пустое значение параметра. Если

	требуется удалить обработчики события для всех пользователей, указывайте значение *null*.
callback	функция-обработчик результата вызова метода. Не обязательный.

Пример:

```
BX24.callUnbind('OnAppUninstall',
'http://www.my-domain.ru/handler/');
```

BX24.userOption

Объект **BX24.userOption** используется для работы с настройками текущего пользователя:

- **void BX24.userOption.set(String name, String value)** - установка значения настройки с именем name.
- **void BX24.userOption.get(String name)** - получение значения настройки с именем name.

Объект работает после BX24.init. Установка значения пользовательских настроек происходит сразу.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.appOption

Объект **BX24.appOption** используется для работы с с общими настройками приложения:

- **void BX24.appOption.set(String name, String value[, Function callback])** - установка значения настройки с именем name.
- **void BX24.appOption.get(String name)** - получение значения настройки с именем name.

Объект работает после BX24.init. Установка значений настроек приложения доступа только пользователям с правом управления приложениями (см. BX24.isAdmin). Для настроек приложения может потребоваться обработчик завершения (параметр **callback** в **BX24.appOption.set**).

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.selectUser

```
void BX24.selectUser(Function callback)
```

Показать стандартный диалог одиночного выбора пользователя. Обработчик получит объект вида **{id: Идентификатор_пользователя, name: отформатированное_имя_пользователя}**.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.selectUsers

```
void BX24.selectUsers(Function callback)
```

Показать стандартный диалог множественного выбора пользователей. Обработчик получит массив объектов вида **{id: Идентификатор_пользователя, name: отформатированное_имя_пользователя}**.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.selectAccess

```
void BX24.selectAccess(Array value, Function callback)

void BX24.selectAccess(Function callback)
```

Показать стандартный диалог выбора прав доступа. Логика диалога позволяет задать "заблокированные" для выбора права доступа (параметр value), но не стартовое значение.

Обработчик получит массив объектов вида {id: Идентификатор_права_доступ, name: название_права_доступа}. Примеры идентификаторов:

- U1 - пользователь с идентификатором 1;
- SG4 - группа соцсети с идентификатором 4;
- AU - все авторизованные пользователи.

Информацию по правам доступа можно посмотреть в уроке Права доступа.

© «Битрикс», 2001-2008, «1С-Битрикс»

BX24.selectCRM

```
BX24.selectCRM({
   entityType: Array value,
   multiple: true,
   value: Array value
})
```

Вызов системного диалога выбора сущности CRM.

Параметр	Описание
entityType	Какие типы объектов выводить в диалоге. Варианты значений: ▪ **lead** - Лиды ▪ **contact** - Контакты ▪ **company** - Компании ▪ **deal** - Сделки ▪ **quote** - Предложения
multiple	Можно ли выбирать несколько объектов. По умолчанию - *false*.
value	Какие объекты сразу добавить в выбранные в диалоге. Работает только в случае `multiple = true`.

Что приходит обработчику:

```
{
    "lead": [
        {
            "id": "L_1348",
            "type": "lead",
```

```
            "place": "lead",
            "title": "Мятный гость №2 - Открытая линия
Битрикс",
            "desc": "Гость",
            "url": "/crm/lead/show/1348/"
        }
    ],
    "contact": [
        {
            "id": "C_2",
            "type": "contact",
            "place": "contact",
            "title": "Пупкин Василий",
            "desc": "",
            "url": "/crm/contact/show/2/",
            "image":
"/upload/resize_cache/crm/8b5/25_25_2/MM35_PG13.jpg"
        }
    ],
    "company": [],
    "deal": [],
    "quote": []
}
```


Примеры

```
BX24.selectCRM({
   entityType: ['lead', 'contact',
'company', 'deal', 'quote'],
   multiple: true,
   value: {lead:[1348,2,35], contact:[2],
company:[4,3], deal:[1,2], quote:[1]}
}, function(){
   console.log(arguments);
})
```

```
BX24.selectCRM({
   entityType: ['lead', 'contact',
'company', 'deal', 'quote'],
```

```
   multiple: true,
   value: ['L_1348', 'L_2', 'L_35', 'C_2',
'CO_4', 'CO_3', 'D_1', 'D_2', 'Q_1']
}, function(){
   console.log(arguments);
})
```

```
BX24.selectCRM({
   entityType: ['lead', 'contact',
'company', 'deal', 'quote'],
   multiple: false,
   value: ['L_35']
}, function(){
   console.log(arguments);
})
```

BX24.isAdmin

```
Boolean BX24.isAdmin()
```

Определяет, имеет ли текущий пользователь права администратора. Работает только после BX24.init.

Смотри также

- user.admin

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.getLang

```
String BX24.getLang()
```

Возвращает идентификатор языка текущего портала. Варианты: ru|en|de.

Пример - подгрузка языковых сообщений из внешнего js-файла:

```
BX24.init(function()
{
    BX24.loadScript('lang/' + BX24.getLang()
+ '.js', function()
    {
        alert('Загрузка завершена!');
    });
});
```

Текущий язык определяется на этапе инициализации js-библиотеки, при вызове до BX24.init функция вернет пустое значение.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.resizeWindow

```
void BX24.resizeWindow(Integer width, Integer height[, Function callback])
```

Изменить размер фрейма с приложением, включая встраивания в виде пользовательских полей. В случае превышения шириной фрейма максимально допустимой ширины страницы **Битрикс24** часть фрейма будет скрыта в силу особенностей верстки. Максимально допустимая ширина зависит от разрешения экрана пользователя.

Смотри так же:

- BX24.getScrollSize
- BX24.fitWindow

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.fitWindow

```
void BX24.fitWindow([Function callback])
```

Установливает размер фрейма с приложением в соответствии с размерами содержимого фрейма.

В силу браузерных ограничений метод может только увеличить размер фрейма. Для точной установки размера можно воспользоваться следующей методикой:

- заключать содержимое приложения в контейнер
- вычислять размеры контейнера
- устанавливать размеры фрейма исходя из размеров контейнера через BX24.resizeWindow

Смотри также:

- BX24.resizeWindow
- BX24.getScrollSize

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.reloadWindow

```
void BX24.reloadWindow()
```

Перезагрузить страницу с приложением (всю страницу, не только фрейм).

© «Битрикс», 2001-2008, «1С-

BX24.setTitle

```
void BX24.setTitle(String title[, Function callback])
```

Установить заголовок страницы.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.ready

```
void BX24.ready(Function handler)
```

Установить обработчик события "DOM-структура документа готова к работе". Аналогично jQuery.ready или BX.ready.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.isReady

```
Boolean BX24.isReady()
```

Флаг "DOM-структура документа готова к работе".

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.proxy

```
Function BX24.proxy(Function func, Object thisObject)
```

Аналогична BX.proxy. Аналог: jQuery.proxy, но с одним отличием: при повторном вызове функции с теми же параметрами будет возвращена ссылка на ту же прокси-функцию, которая была результатом первого вызова, а не новая прокси-функция.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.closeApplication

```
void BX24.closeApplication();
```

Метод закрывает открытое модальное окно с приложением (открытым как через BX24.openApplication, так и через модальное окно обработчика мест встраивания CRM_*_LIST_MENU).

Рекомендуется к использованию в CRM_*_LIST_MENU, например, для показа кнопки закрытия. (По умолчанию у пользователей нет никакого способа вернуться в CRM кроме закрытия всплывающего окна по крестику в углу окна.)

Пример

Единый пример для BX24.openApplication и BX24.closeApplication

```
><script src="//api.bitrix24.com/api/v1/">
</script>
<?
// разбор входных данных
$placementOptions = array();
if(array_key_exists('PLACEMENT_OPTIONS',
$_REQUEST))
{
        $placementOptions =
json_decode($_REQUEST['PLACEMENT_OPTIONS'],
true);
}

// если приложение не развернуто, выводим
```

```
кнопку открытия, в противном случае закрытия
if(!isset($placementOptions['opened']))
{
?>
        <span
onclick="openApplication()">Open</span>
<?
}
else
{
?>
        <span
onclick="closeApplication()">Close</span>
<?
}

?>
<script>
        function openApplication()
        {
                BX24.openApplication(
                        {
                                'opened':
true // данные, передаваемые открываемому
приложению
                        },
                        function()
                        {
                                // этот
обработчик сработает, когда приложение будет
закрыто

alert('Application closed!')
                        }
                );

                setTimeout(closeApplication,
```

```
15000); // автоматически закрыть через 15
секунд
        }

        function closeApplication()
        {
                BX24.closeApplication();
        }
</script>
```

BX24.getDomain

```
void BX24.getDomain()
```

Данный метод возвращает **window.location.host** родительского окна, т.е. возвращает адрес портала *Битрикс24*. Например, значение может быть таким: `dev.bitrix24.ru`.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.openApplication

```
void BX24.openApplication([
    Object parameters[,
    Function closeCallback
    ]
]);
```

При вызове метода будет открыто всплывающее окно с фреймом приложения. Приложению будут переданы данные из параметра parameters. При закрытии всплывающего окна будет вызван обработчик closeCallback. Метод может контролировать размеры, заголовок, и лейбл сладера.

Параметры

Параметр	Описание
parameters	Объект с параметрами, которые будут переданы открываемому приложению в виде JSON-строки
closeCallback	Обработчик закрытия приложения
bx24_width	Ширина слайда
bx24_label	Заголовок плашки
bx24_title	Заголовок страницы
bx24_leftBoundary	Слайдер во всю ширину с отступом слева. Не может быть одновременно с bx24_width.

Для плейсментов CRM_*_LIST_MENU заблокировано.

Примеры

Единый пример для BX24.openApplication и BX24.closeApplication

```
<script src="//api.bitrix24.com/api/v1/">
</script>
<?
// разбор входных данных
$placementOptions = array();
if(array_key_exists('PLACEMENT_OPTIONS',
$_REQUEST))
{
        $placementOptions =
json_decode($_REQUEST['PLACEMENT_OPTIONS'],
true);
}

// если приложение не развернуто, выводим
кнопку открытия, в противном случае закрытия
if(!isset($placementOptions['opened']))
{
?>
        <span
onclick="openApplication()">Open</span>
<?
}
else
{
?>
        <span
onclick="closeApplication()">Close</span>
<?
```

```
}

?>
<script>
        function openApplication()
        {
                BX24.openApplication(
                        {
                                'opened':
true // данные, передаваемые открываемому
приложению
                        },
                        function()
                        {
                                // этот
обработчик сработает, когда приложение будет
закрыто

alert('Application closed!')
                        }
                );

                setTimeout(closeApplication,
15000); // автоматически закрыть через 15
секунд
        }

        function closeApplication()
        {
                BX24.closeApplication();
        }
</script>
```

Пример со слайдером

```
BX24.openApplication(
            {
                'opened': true,
                'bx24_width': 450,// int
                'bx24_label': {
                    'bgColor':'pink', //
aqua/green/orange/brown/pink/blue/grey/viole
t
                    'text': 'my task',
                    'color': '#07ff0e',
                },
                'bx24_title': 'my title', //
str
                //'bx24_leftBoundary': 300,
//int
            },
            function()
            {
                console.log('Application
closed!')
            }
       );
```

BX24.openPath

```
BX24.openPath(path,callback);
```

Метод открывает указанный путь внутри портала в слайдере.

Внимание! По соображениям безопасности метод не работает в мобильном приложении.

Параметры

Параметр	Описание

path	Путь внутри портала, может начинаться с:
	`^\/(crm\/(deal|lead|contact|company)|marketplac`
	`9]+|workgroups\/group\/[0-9]+)\/`

callback	функция вызывается в 2 случаях:
	▪ при ошибке открытия
	▪ если указанный путь нет возможности откры
	`"PATH_NOT_AVAILABLE"}`
	▪ в мобильном приложении: `{result: "error`
	`"METHOD_NOT_SUPPORTED_ON_DEVICE"}`
	▪ при закрытии слайда: `{result: "close"}`

Пример:

```
<script src="//api.bitrix24.com/api/v1/">
</script>
<script>
    BX24.init(
        function()
        {
            BX24.openPath(
                '/crm/deal/details/5/',
                function(result)
                {
                    console.log(result);
                }
            );
        }
    );
</script>
```

BX24.proxyContext

```
Object BX24.proxyContext()
```

При вызове изнутри прокси-функцию выдаст ссылку на оригинальный контекст выполнения прокси-функции.

© «Битрикс», 2001-2008, «1С-

BX24.scrollParentWindow

```
BX24.scrollParentWindow (Scroll, Callback)
```

Метод прокручивает родительское окно. В первом параметре нужно указать позицию вертикального ползунка (0 - прокрутить к самому началу). Во втором указывается callback-функция, которая будет вызвана после изменения позиции ползунка.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.bind

```
void BX24.bind(DOMNode element, String eventName, Function func)
```

Установить функцию func в качестве обработчика события eventName объекта element. Аналогична BX.bind.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.unbind

```
void BX24.unbind(DOMNode element, String eventName,
Function func)
```

Убрать функцию func в качестве обработчика события eventName объекта element. Аналогична BX.unbind.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.getScrollSize

```
Object BX24.getScrollSize()
```

Функция вернет внутренние размеры фрейма приложения в виде объекта с полями **scrollWidth** и **scrollHeight**. Возвращаемые значения могут зависеть не только от размеров фрейма, но и от верстки приложения.

Смотри также:

- BX24.resizeWindow
- BX24.fitWindow

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.loadScript

```
void BX24.loadScript(Array|String script[, Function callback])
```

Загрузить и выполнить клиентский javascript-файл script. Можно передать массив файлов, в этом случае они будут загружены последовательно. Обработчик callback будет вызван после завершения всех загрузок.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.im.callTo

```
void BX24.im.callTo(userId[, video=true])
```

Звонок по внутренней связи.

Параметры функции

Параметр	Описание
userId	Идентификатор пользователя портала.
video	**true** - видеозвонок, **false** - аудиозвонок. Необязательный параметр.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.im.phoneTo

```
void BX24.im.phoneTo(number)
```

Звонок на телефонный номер.

Параметры функции

Параметр	Описание
number	Номер телефона (строка). Номер может быть формате: `84012112233` или `8 (495) 711-22-33`.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.im.openMessenger

```
void BX24.im.openMessenger(dialogId)
```

Открытие окна мессенджера.

Параметры функции

Параметр	Описание
dialogId	Идентификатор диалога: ▪ **userId** или **chatXXX** - чат, где XXX - идентификатор чата, это может быть просто цифра. ▪ **sgXXX** - чат группы, где XXX - номер группы соцсети (в этой группе чат должен быть разрешен). ▪ **imol|XXXX** - открытая линия, где XXX - это код, полученный через Rest-метод imopenlines.network.join. Если ничего не передать, будет открыт интерфейс чата с последним открытым диалогом.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

BX24.im.openHistory

```
void BX24.im.openHistory(dialogId)
```

Открытие окна истории.

Параметры функции

Параметр	Описание
dialogId	Идентификатор диалога: ▪ **userId** или **chatXXX** - чат, где XXX - идентификатор чата, это может быть просто цифра. ▪ **imol|XXXX** - открытая линия, где XXX - это номер сессии открытой линии.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Особенности списочных методов и пакетного метода Batch

В REST существует ряд методов, которые возвращают списки элементов - списки сделок, пользователей, комментариев к задаче. Поскольку количество элементов, возвращаемых методами REST, зависит от конкретных условий и параметров, то Битрикс24 возвращает элементы «пакетами» по несколько элементов (в настоящий момент не более, чем по 50).

Пример:

```
https://my.bitrix24.ru/rest/methods.xml?
auth=d161f25928c3184678924ec127edd29a
//получить список доступных методов в формате xml.

https://my.bitrix24.ru/rest/entity.item.get.json?
ENTITY=menu&auth=d161f25928c3184678924ec127edd29a
//получить в формате json список всех элементов сущности
menu.
```

При вызове списочных методов REST возвращает дополнительные значения в ответе:

Для получения следующего пакета элементов, необходимо выполнить тот же самый запрос, указав дополнительный параметр start со значением, пришедшем в параметре next ответа.

Пример:

```
{
        "result":результат выполнения метода,
        "error":ошибка выполнения метода,
        "total":общее количество записей в ответе
списочного метода,
        "next":значение, которое нужно послать для
```

```
получение следующей страницы данных списочного метода
}
```

Пример вывода следующего пакета элементов есть в описании метода crm.lead.list.

Исключение из этого - метод пакетного выполнения запросов batch, который возвращает ответ вида:

```
{
        "result":массив результатов запросов пакета,
        "result_error":массив ошибок запросов пакета,
        "result_total":массив количеств записей в ответах
списочных методов,
        "result_next":массив значений, возвращенных в полях
next запросов
}
```

method.get

```
method.get(name)
```

Метод возвращает 2 параметра:

- `isExisting => true/false` - параметр определяет существует ли метод именно на этом портале;
- `isAvailable => true/false` - параметр определяет доступен ли метод для вызова с текущими доступами (scope) приложения.

Параметры

Параметр	Тип	Описание
name	string	Имя метода для проверки.

Примеры

```
$result = CRest::call(
    'method.get',
    [
        'name' => 'user.get',
    ]
```

```
);
print_r($result);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

methods

Внимание! Данный метод устарел и в срок до 1 сентября 2022 года будет удалён. Настоятельно рекомендуется использовать метод method.get.

```
BX24.callMethod('methods', {});
```

Без параметров - показ списка методов, доступных текущему приложению.

Дополнительные параметры:

Поле	Описание
full=true	показ всех методов.
scope=имя_разрешения	показ методов, входящих в данное разрешение. Если указан параметр без значения (`methods?scope=&auth=xxxxx`), то будут выведены все общие методы.

Пример

```
https://my.bitrix24.ru/rest/methods?scope=&auth=d161f25928c3184678924ec127edd29a - показать все общедоступные методы.
```

Ответ

{"result":
["scope","methods","batch","user.admin","user.access","acce
ss.name"]}

scope

```
BX24.callMethod('scope', {});
```

Если метод вызван без параметров, то он вернет все разрешения, доступные для данного приложения.

Если же метод вызван с параметром *full=true*, то он вернет полный список разрешений.

Пример:

Запрос

```
https://my.bitrix24.ru/rest/scope?
auth=d161f25928c3184678924ec127edd29a
```

Ответ JSON

```
{"result":["entity","user","log","department"]}
```

Ответ XML

```
<response>
        <result>
                <item>entity</item>
                <item>user</item>
                <item>log</item>
                <item>department</item>
        </result>
</response>
```

app.info

Описание и пример

Показ информации о приложении. Метод поддерживает безопасный вызов.

Пример

Запрос

```
http://my.bitrix24.ru/rest/app.info?
auth=d161f25928c3184678924ec127edd29a
```

Ответ JSON


```
{
  "result": {
      "ID":"7",
     "CODE":"local.56017020f07e17.98523769",
     "VERSION":1,
     "STATUS":"L",
     "INSTALLED":true,
     "PAYMENT_EXPIRED":"N",
     "DAYS":null,
     "LICENSE":"ru_project"
  }
}
```


Ответ XML


```
<response>
  <result>
     <ID>7</ID>
```

```
<CODE>local.56017020f07e17.98523769</CODE>
     <VERSION>1</VERSION>
     <STATUS>L</STATUS>
     <INSTALLED>1</INSTALLED>
     <PAYMENT_EXPIRED>N</PAYMENT_EXPIRED>
     <DAYS></DAYS>
     <LICENSE>ru_project</LICENSE>
  </result>
</response>
```

Поля ответа

Поле	Описание
ID	локальный идентификатор приложения на портале
CODE	код приложения
VERSION	установленная версия приложения
STATUS	статус приложения. Возможные значения: ▪ **F** (Free) - бесплатное; ▪ **D** (Demo) - демо-версия; ▪ **T** (Trial) - триальная версия (ограниченная по времени); ▪ **P** (Paid) - оплаченное приложение; ▪ **L** (Local) - локальное приложение. ▪ **S** (Subscription) - подписное приложение.
INSTALLED	[true|false] Статус установленности приложения. Если приложение не установлено, оно доступно только администраторам портала и должно сигнализировать об окончании установки при помощи вызова BX24.installFinish()

PAYMENT_EXPIRED	[Y|N] флаг, показывающий, истек ли оплаченный период или период триального использования.
DAYS	Количество дней, оставшееся до конца оплаченного периода или периода триального использования.
LICENSE	Обозначение тарифа с указанием региона в виде префикса.Состоит из базового языка портала и идентификатора тарифного плана. В случае с тарифами, состав которых менялся при сохранении публичного наименования (как CRM+, Команда и Компания), понять, какой именно тариф действует по этому полю нельзя. Примеры вариантов значений: ▪ **ru_project** - тариф Проект ▪ **ru_tf** - тариф Проект+ ▪ **ru_team** - тариф Команда ▪ **ru_company** - тариф Компания ▪ **ru_demo** - демо-режим ▪ **ru_nfr** - NFR-лицензия ▪ **ru_selfhosted** - версия в которой не установлен модуль Битрикс24(коробка).
LICENSE_TYPE	Внутреннее обозначение тарифа без указания региона. Нужно, чтобы различать тарифы, состав которых менялся при сохранении публичного наименования (как CRM+, Команда и Компания)
LICENSE_FAMILY	Обозначение тарифа без указания региона.

По истечении оплаченного периода приложение продолжит работать в течение периода, отводимого на возможные задержки поступления оплаты (grace period), по окончании которого будет автоматически переключено на демо-режим или заблокировано. При этом значение флага PAYMENT_EXPIRED будет равен Y, а поле DAYS будет содержать отрицательное число.

batch

Описание

Выполнение пакета запросов.

В некоторых случаях возникает необходимость отправить несколько запросов подряд. Для оптимизации процесса можно использовать пакетное выполнение запросов.

Параметры

Параметр	Описание
halt	Определяет прерывать ли последовательность запросов в случае ошибки.
cmd	Массив запросов стандартного вида (следует помнить про квотирование данных запросов; получается, что данные для подзапросов должны пройти двойное квотирование).

Примечание: Количество запросов в пакете ограничено 50.

Массив запросов может быть как с числовыми ключами, так и ассоциативным. В параметрах каждого последующего запроса можно использовать данные предыдущих запросов в таком виде:

```
$result[идентификатор_запроса][поле_ответа]
```

где идентификатором запроса служит его ключ в массиве запросов.

Пример

```
https://my.bitrix24.ru/rest/batch.xml?
auth=d161f25928c3184678924ec127edd29a&halt=0&cmd[get_user]=
user.current%3F&cmd[get_department]=department.get%3FID%3D%
2524result%255Bget_user%255D%255BUF_DEPARTMENT%255D
```

> **Обратите внимание**, что параметры URL-кодированы. Рекомендация кодировать параметры - обязательна, в противном случае корректность результата не гарантируется.

Ответ XML


```
<response>
        <result>
                <result>
                        <get_user>
                                <ID>1</ID>
                                <LOGIN>admin</LOGIN>
                                <ACTIVE>1</ACTIVE>

<EMAIL>sigurd@example.com</EMAIL>
                                <NAME>Одмин</NAME>
                                <LAST_NAME/>
                                <SECOND_NAME/>
                                <PERSONAL_GENDER/>
                                <PERSONAL_PROFESSION/>
                                <PERSONAL_WWW/>
                                <PERSONAL_BIRTHDAY>1955-04-
10T00:00:00+03:00</PERSONAL_BIRTHDAY>

<PERSONAL_PHOTO>/upload/main/80c/44169_C5_PrimalWaterE500CC
.jpg</PERSONAL_PHOTO>
                                <PERSONAL_ICQ/>
                                <PERSONAL_PHONE/>
                                <PERSONAL_FAX/>
                                <PERSONAL_MOBILE/>
                                <PERSONAL_PAGER/>
                                <PERSONAL_STREET/>
                                <PERSONAL_CITY/>
                                <PERSONAL_STATE/>
                                <PERSONAL_ZIP/>

<PERSONAL_COUNTRY>0</PERSONAL_COUNTRY>
                                <WORK_COMPANY/>
                                <WORK_POSITION/>
                                <UF_DEPARTMENT>
                                <item>128</item>
```

```
                                </UF_DEPARTMENT>
                                <UF_INTERESTS/>
                                <UF_SKILLS/>
                                <UF_WEB_SITES/>
                                <UF_XING/>
                                <UF_LINKEDIN/>
                                <UF_FACEBOOK/>
                                <UF_TWITTER/>
                                <UF_SKYPE/>
                                <UF_DISTRICT/>
                                <UF_PHONE_INNER/>
                        </get_user>
                        <get_department>
                                <item>
                                        <ID>128</ID>
                                        <NAME>ИТ-
отдел</NAME>
                                        <SORT>500</SORT>

<PARENT>114</PARENT>

<UF_HEAD>255</UF_HEAD>
                                </item>
                        </get_department>
                </result>
                <result_error/>
                <result_total>
                        <get_department>1</get_department>
                </result_total>
                <result_next/>
        </result>
</response>
```


Пример строки json для помещения в тело POST запроса для метода batch.

```
BX24.callMethod(
   'batch',
   {
      'halt': 0,
      'cmd': {
         'user': 'user.get?ID=1',
         'first_lead': 'crm.lead.add?fields[TITLE]=Test
Title',
         'user_by_name': 'user.search?NAME=Test2',
         'user_lead': 'crm.lead.add?fields[TITLE]=Test
Assigned&fields[ASSIGNED_BY_ID]=$result[user_by_name][0]
[ID]',
      }
   },
```

```
    function(result)
    {
        console.log(result.answer);
    }
);
```

В результате:

- **user** - вернёт пользователя с ID = 1
- **first_lead** - создаст лид
- **user_by_name** - найдёт пользователя с именем "Test2"
- **user_lead** - создаст лид с ответственным пользователем, найденным в user_by_name

На PHP рекомендуется использовать CRest::callBatch()

Смотрите также

- BX24.callBatch

user.admin

Определение, обладает ли текущий пользователь правами на управление настройками приложений.

Пример:

Запрос

```
https://my.bitrix24.ru/rest/user.admin.json?
auth=d161f25928c3184678924ec127edd29a
```

Ответ JSON

```
{"result":true}
```

Смотри также

- BX24.isAdmin

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

user.access

```
BX24.callMethod('user.access', {ACCESS:['U22','U33']});
```

Определение, обладает ли текущий пользователь хотя бы одним из заданного параметром *ACCESS* набора прав.

Параметры

Параметр	Описание
ACCESS	Обязательный. Идентификатор или список идентификаторов прав, доступ к которым нужно проверить.

Пример:

```
https://my.bitrix24.ru/rest/user.access.json?
ACCESS[]=U22&ACCESS[]=U33&auth=d161f25928c3184678924ec127ed
d29a
```

Ответ JSON

```
{"result":false}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

access.name

```
BX24.callMethod('access.name', {ACCESS:['AU']});
```

Получение названий прав доступа.

Параметры

Параметр	Описание
ACCESS	Обязательный. Список идентификаторов прав, названия для которых нужно получить.

Пример:

```
https://my.bitrix24.ru/rest/access.name.xml?
ACCESS[]=AU&auth=d161f25928c3184678924ec127edd29a
```

Ответ XML

```
<response>
        <result>
                <AU>
                        <provider/>
                        <name>Все авторизованные
пользователи</name>
                </AU>
        </result>
</response>
```

profile

Позволяет получить базовую информации о текущем пользователе без каких-либо скоупов в отличие от user.current.

Возвращаемые поля:

```
ADMIN true
ID "1"
LAST_NAME "Востриков"
NAME "Сергей"
PERSONAL_GENDER ""
PERSONAL_PHOTO "https://***.bitrix24.ru/*****
TIME_ZONE null
TIME_ZONE_OFFSET 7200
```

© «Битрикс», 2001-2008, «1С-

server.time

Метод возвращает текущее время сервера в формате YYYY-MM-DDThh:mm:ss±hh:mm.

Параметры

Без параметров

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

events

```
BX24.callMethod('events', {});
```

Показ общего списка событий.

scope=имя_разрешения - показ событий, входящих в данное разрешение. Если указан параметр без значения (events?scope=&auth=xxxxx), то будут выведены все общие события.

Пример:

```
https://my.bitrix24.ru/rest/events?
auth=d161f25928c3184678924ec127edd29a - показать все
доступные
события.
```

Ответ:

```
{"result":["ONAPPUNINSTALL","ONAPPUPDATE"]}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

event.bind

Регистрация нового обработчика события.

Метод может работать как при авторизации под пользователем с правами администрирования портала, так и под обычным пользователем. Метод для пользователя без прав администратора доступен с ограничениями:

- офлайн-события недоступны, попытка установки будет порождать исключение;
- события устанавливаются от имени текущего пользователя (см. описание параметра auth_type); явное указание auth_type, отличного от ID текущего пользователя, также будет порождать исключение;

Поскольку запросы будут идти с серверов Битрикс, то любой URL должен быть доступен для GET/POST запросов извне.

Интерфейс для данного метода - BX24.callBind.

Параметры

Параметр	Описание
event	Имя события. Обязательный.
handler	Ссылка на обработчик события. Обязательный.
auth_type	Идентификатор пользователя, под которым авторизуется обработчик события. Не обязательный. По умолчанию будет использоваться авторизация пользователя,

	действия которого привели к срабатыванию события.
event_type	Значения: online|offline. По умолчанию event_type=online, и поведение метода не меняется. Если вызывается event_type=offline, то метод работает с офлайн событиями.
auth_connector	Ключ источника. Параметр предназначен для офлайн событий. Позволяет исключать ложные срабатывания событий.
options	Дополнительные настройки для регистрируемого события, при наличии.

Пример:

```
https://my.bitrix24.ru/rest/event.bind.json?
auth=a25e86871fcebb24f4d9076caf2e6623&auth_type=0&event=OnA
ppUpdate&handler=http%3A%2F%2Fwww.my-
domain.com%2Fhandler%2F
```

Ответ:

```
{"result":true}
```

event.get

Получение списка зарегистрированных обработчиков событий.

Параметры

Параметров не имеет.

Пример:

```
https://my.bitrix24.ru/rest/event.get?
auth=db12e4b28d20af8a78da516510768de9
```

Ответ:

```
{"result":
[{"event":"ONAPPUNINSTALL","handler":"https:\/\/www.apphost
.com\/eventhandler.php","auth_type":"0"}]}
```

© «Битрикс», 2001-2008, «1С-

event.offline.clear

Метод производит очистку записей в очереди офлайн событий.

Параметры

Метод	Описание	Тип
process_id	Идентификатор процесса, который занимается обработкой записей. Обязательный параметр.	
id	Массив идентификаторов записей, которые нужно вычистить. Не обязателен, по умолчанию будут вычищены все записи, помеченные переданным process_id.	array
message_id	Массив значений поля MESSAGE_ID записей, которые нужно вычистить. Игнорируется, если указан параметр id. Не обязателен, по умолчанию будут вычищены все записи, помеченные переданным process_id.	array

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

event.offline.error

Метод сохраняет запись в базе с пометкой об ошибке при использовании офлайновых событий.

Метод	Описание	Тип
process_id	Идентификатор процесса, занимается обработкой записей. Обязательный параметр.	
message_id	Массив значений поля MESSAGE_ID записей, которые нужно пометить как ошибочные.	array

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

event.offline.get

Метод возвращает приложению первые в очереди офлайновые события согласно установкам фильтра.

Параметры метода

Метод	Описание	Тип
filter	Фильтр записей. По умолчанию отдаются все записи, без фильтрации. Поддерживается фильтрация по полям: ID, TIMESTAMP_X, EVENT_NAME, MESSAGE_ID со стандартными операциями типа =, >, <, <= и так далее.	array
order	Сортировка записей. Поддерживается сортировка по тем же полям, что и в фильтре, на вход принимается массив вида `[поле=>ASC|DESC]`. По умолчанию - [TIMESTAMP_X:ASC].	array
limit	Количество выбираемых записей. По умолчанию 50.	int

Дополнительные параметры

Метод	Описание
clear	Значения: 0|1 - удалять ли выбранные записи. По умолчанию 1
process_id	Идентификатор процесса. Используется если понадобится повторно выбрать еще не обработанные текущим процессом записи.
error	Значения: 0|1 - возвращать ли ошибочные записи (см. ниже). По умолчанию 0.

Метод поддерживает многопоточный разбор. То есть допускается несколько параллельных запросов к /rest/event.offline.get (с соблюдением ограничений на количество запросов в единицу времени), и каждый из них получит разные наборы записей.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

event.offline.list

Метод для чтения текущей очереди без вносения изменений в ее состояние, в отличие от event.offline.get

Параметры

Метод	Описание	Тип
filter	Фильтр записей. По умолчанию отдаются все записи, без фильтрации. Поддерживается фильтрация по полям: ID, TIMESTAMP_X, EVENT_NAME, MESSAGE_ID, PROCESS_ID, ERROR со стандартными операциями типа =, >, <, <= и т.д.	array
order	Сортировка записей. Поддерживается сортировка по тем же полям, что и в фильтре. На вход принимается массив вида [поле=>ASC|DESC]. По умолчанию - [ID:ASC].	array

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

event.unbind

Отмена зарегистрированного обработчика события.

Работает только при авторизации под пользователем с правами администрирования портала.

Параметры

Параметр	Описание
event	Имя события. Обязательный.
handler	Ссылка на обработчик события. Обязательный.
auth_type	Идентификатор пользователя, под которым авторизуется обработчик события. Не обязательный. **Примечание**: если требуется удалить обработчики события, установленные с пустым *auth_type* (с авторизацией от имени пользователя, вызвавшего событие), но оставить остальные обработчики, указывайте *auth_type=0* или пустое значение параметра.
event_type	Значения: online|offline. По умолчанию event_type=online, и поведение метода не меняется. Если казывается event_type=offline, то метод работает с офлайн событиями.

Если какие-либо параметры не указаны, то будут удалены все обработчики события, удовлетворяющие остальным требованиям.

Возвращаемое значение

Метод вернет количество удаленных при вызове обработчиков событий.

Пример:

```
https://my.bitrix24.ru/rest/event.unbind.json?
auth=a25e86871fcebb24f4d9076caf2e6623&auth_type=0&event=OnA
ppUpdate&handler=http%3A%2F%2Fwww.my-domain.ru%2Fhandler%2F
```

Ответ:

```
{"result":{"count":1}}
```

Смотри также

- BX24.callUnbind

OnAppInstall

Событие вызывается сразу после успешной установки приложения на Битрикс24. В обработчик передается **application_token**, который важно сохранить (см Проверка безопасности в событиях).

Параметры

Без параметров

Важно: обработчик данного события можно установить в установочном скрипте приложения (который указывается в карточке версии в отдельном поле).

© «Битрикс», 2001-2008, «1С-

OnAppMethodConfirm

Событие, вызывается при получении решения администратора портала по запросу на использование методов, требующих подтверждения.

Параметры события

Параметр	Описание
TOKEN	Авторизационный токен, с которым было запрошено разрешение
METHOD	Метод REST API, разрешение на использование которого было запрошено.
CONFIRMED	Результат разрешения, 0 - запрещено, 1 - разрешено

Пример данных события:

```
array (
  'event' => 'ONAPPMETHODCONFIRM',
  'data' =>
  array (
    'TOKEN' => 'fkp963yuv1ggkfbs5z3f5hy8lilm0iw6',
    'METHOD' => 'voximplant.user.get',
    'CONFIRMED' => '1',
    'LANGUAGE_ID' => 'ru',
  ),
  'ts' => '1478790852',
  'auth' =>
  array (
```

```
    'domain' => 'portal.bitrix24.ru',
    'client_endpoint' =>
'https://portal.bitrix24.ru/rest/',
    'server_andpoint' => 'https://oauth.bitrix.info/rest/',
    'member_id' => '74ef8a46a75104de55d5d4a61b98ab6d',
    'application_token' =>
'c289487163b58658eae5e8b42eaf11b8',
  ),
)
```

OnAppPayment

Событие, вызываемое при оплате приложения.

Поля ответа:

Поле	Описание
CODE	код приложения
VERSION	установленная версия приложения
STATUS	статус приложения. Возможные значения: ▪ **F** (Free) - бесплатное ▪ **D** (Demo) - демо-версия ▪ **T** (Trial) - триальная версия (ограниченная по времени) ▪ **P** (Paid) - оплаченное приложение
PAYMENT_EXPIRED	[Y|N] флаг, показывающий, истек ли оплаченный период или период триального использования.
DAY	количество дней, оставшееся до конца оплаченного периода или периода триального использования.

Смотри так же

- app.info

OnAppUninstall

Событие, вызываемое при удалении приложения.

Параметры события:

Параметр	Описание
CLEAN=1/0	Значение флага "очистить данные приложения", который задается пользователем при удалении приложения.
application_token	Секретный идентификатор (см Проверка безопасности в событиях).

Внимание: при удалении приложения удаляются все права на доступ приложения к REST API. Поэтому, несмотря на то, что обработчику события будут переданы авторизационные данные, он уже не может использовать API от имени удаленного приложения.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onOfflineEvent

Событие, уведомляющее о появлении новых офлайн событий с некоторой периодичностью.

Приложение может подписаться на события 2 видов:

- Обычные: событие вызывает внешний URL и выполняется действие определяемое этим адресом.
- Офлайн: вместо вызова внешнего URL происходит локальное сохранение событий на портале, откуда потом можно забрать события методами event.offline.*.

У события onOfflineEvent по факту локального сохранения вычисляется необходимость отправки уведомления и, затем, оно отправляется как обычное событие на внешний URL.

Подробнее об [ds]офлайн событиях[/ds][di]Приложение не в всегда в состоянии принимать события. Оно может скрываться за фаерволлами, жить во внутренней сети и т.д. В этом случае используется механизм офлайн-событий, когда приложение подписывается на события, но не указывает URL обработчика.

Подробнее ...[/di].

Параметры

Параметр	Описание	С версии
minTimeout	Таймаут в секундах. По умолчанию 1 сек. Если значения параметра: Если равно 0, вне зависимости от количества добавленных в	

	офлайн очередь событий отправится только 1 событие на адрес обработчика в рамках одного хита; Если больше 0, то при первом срабатывании отправляет одно событие. Далее делается пауза минимум на время таймаута до отправки следующего события.	

Пример

```
CRest::call(
    'event.bind',
    [
       'event' => 'ONOFFLINEEVENT',
       'handler' =>
'https://example.com/handler.php',
       'options' => [
           'minTimeout' => 30,
       ]
   ]
);
```

OnUserAdd

Событие вызывается при добавлении пользователя в Битрикс24. Событие срабатывает не после приглашения, а после того как пользователь зайдёт на портал и зарегистрируется до конца.

Параметр	Описание	С версии
event	ONUSERADD	
data	Массив данных пользователя. Ключи массива: ▪ ID - индентификатор пользователя ▪ ACTIVE - флаг активности ▪ EMAIL - e-mail пользователя ▪ NAME - имя пользователя ▪ LAST_NAME - фамилия пользователя ▪ PERSONAL_GENDER - пол ▪ PERSONAL_BIRTHDAY - дата рождения ▪ UF_DEPARTMENT - массив подразделений, к которым будет добавлен пользователь. Если пользователь регистрируется в Экстранет, то ключ отсутствует.	
auth	Массив авторизационных данных	

ts	Временной штамп создания события	

Проверка безопасности в событиях

Разработчик приложения в обработчиках событий должен убедиться в том, что обработчик вызывается именно Битрикс24, а не злоумышленниками. Для этого Битрикс24 при вызове обработчиков передает дополнительный параметр **application_token**.

В первый раз параметр передаётся в обработчик события OnAppInstall вместе с данными авторизации пользователя, установившего приложение. При помощи этих данных авторизации обработчик события OnAppInstall может удостовериться в актуальности полученного **access_token** и затем запомнить application_token, чтобы в дальнейшем, в своих обработчиках других событий сверять получаемый application_token с сохраненным.

Особенно это актуально в обработчике события OnAppUninstall, поскольку в него не передаются данные авторизации (приложение уже удалено на Битрикс24). Поэтому в случае с OnAppUninstall сверка application_token с сохраненным значением становится единственным способом удостовериться, что обработчик события вызван именно Битрикс24.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

app.option.get

```
app.option.get(
   option
)
```

Получает данные, привязанные к приложению. Если ничего не подать на вход, вернёт все свойства записанные через app.option.set.

Параметры

Параметр	Описание	С версии
option	Строка, один из ключей из свойства **app.option.set**.	

Пример

```
CRest::call('app.option.get',[//вернёт свойство с ключом
'data'
          'option' =>    'data'
       ])
```

```
CRest::call('app.option.get',[]) //вернёт все свойства
```

app.option.set

```
app.option.set(
   options
)
```

Метод привязывает данные к приложению.

Параметры

Метод	Описание	С версии
options	Массив, где ключ - название сохраняемого свойства, а значение - значение свойства. Если подать значение с новым ключом, то метод его запишет, если существующее, то обновит.	

Примеры

```
CRest::call('app.option.set',[
           "options"=>[
               'data' => 'value',
               'data2' => 'value2',
           ]
       ]);
```

```
CRest::call('app.option.set',[
           "options"=>[
               'data' => 'NewValue',
```

```
            ]
        ]),
```

user.option.*

Методы **user.option.*** аналогичны методам **app.option.***, только привязка идёт к приложению и пользователю

user.option.get аналогичен app.option.get.

user.option.set аналогичен app.option.set.

В зависимости от типа приложения может привязываться к установившему пользователю или к пользователю с которым взаимодействует. (Приложения второго типа.)

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Структура таблиц

Описание и пример

Важно! Настоятельно рекомендуется при обновлении адреса передавать полный набор полей адреса в метод обновления. Частичное обновление поддерживается, но нежелательно из-за возможных конкурирующих запросов на обновление адреса другими приложениями.

```
Допустим, у компании был адрес "г. Калининград, ул.
Фридриха Маркса, д. 50, кв. 5".

Чужое приложение: чтение адреса "г. Калининград, ул.
Фридриха Маркса, д. 50, кв. 5".
Ваше приложение: чтение адреса "г. Калининград, ул.
Фридриха Маркса, д. 50, кв. 5".

Чужое приложение: сохранение адреса "г. Светлогорск, ул.
Ленина, д. 100, кв. 10" (исправление ошибки - полная замена
адреса на адрес другого филиала).
Ваше приложение: сохранение части адреса "ул. Карла Маркса"
(исправление ошибки - корректировка названия улицы, а на
самом деле создание новой ошибки).

В результате двух конкурирующих запросов на обновление
адрес компании стал "г. Светлогорск, ул. Карла Маркса, д.
100, кв. 10".
Таким образом, мы можем получить неконсистентный адрес.
```

Внимание! Более полный перечень полей приводится на страницах методов, возвращающих описание полей сущности. Такие методы имеют название *crm.название сущности.fields*.

Сделки

Поле	Описание	Чтение	Запись
ID	Идентификатор сделки. Создается автоматически и уникален в рамках БД.	Да	Нет
TITLE	Название сделки. Обязательное поле.	Да	Да
TYPE_ID	Идентификатор типа сделки.	Да	Да
STAGE_ID	Идентификатор этапа сделки	Да	Да
PROBABILITY	Вероятность заключения сделки (в %)	Да	Да
CURRENCY_ID	Валюта сделки	Да	Да
OPPORTUNITY	Сумма в валюте сделки	Да	Да
COMPANY_ID	Идентификатор компании-контрагента сделки	Да	Да
CONTACT_ID	Идентификатор контактного лица	Да	Да
BEGINDATE	Дата открытия сделки	Да	Да

CLOSEDATE	Дата закрытия сделки	Да	Да
OPENED	Флаг "Сделка доступна для всех"	Да	Да
CLOSED	Флаг "Сделка закрыта"	Да	Да
COMMENTS	Комментарии	Да	Да
ASSIGNED_BY_ID	Идентификатор ответственного за сделку	Да	Да
CREATED_BY_ID	Идентификатор создавшего сделку	Да	Нет
MODIFY_BY_ID	Идентификатор изменившего сделку	Да	Нет
DATE_CREATE	Дата создания	Да	Нет
DATE_MODIFY	Дата изменения	Да	Нет
LEAD_ID	Идентификатор лида	Да	Нет
ADDITIONAL_INFO	Дополнительная информация	Да	Да
ORIGINATOR_ID	Идентификатор внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да	Да
ORIGIN_ID	Внешний ключ,	Да	Да

	используется для операций обмена. Идентификатор объекта внешней информационной базы. Назначение поля может меняться конечным разработчиком.		
CATEGORY_ID	Идентификатор направления сделки.	Да	Да

Лиды

Поле	Описание	Чтение	Запис
ID	Идентификатор лида.	Да	Нет
TITLE	Название лида. Обязательное поле.	Да	Да
NAME	Имя контакта	Да	Да
SECOND_NAME	Отчество контакта	Да	Да
LAST_NAME	Фамилия	Да	Да
COMPANY_TITLE	Название компании	Да	Да
SOURCE_ID	Идентификатор источника	Да	Да

SOURCE_DESCRIPTION	Дополнительно об источнике	Да	Да
STATUS_ID	Идентификатор статуса лида	Да	Да
STATUS_DESCRIPTION	Дополнительная информация о статусе	Да	Да
POST	Должность	Да	Да
ADDRESS	Улица, дом, корпус, строение	Да	Да
ADDRESS_2	Квартира, офис	Да	Да
ADDRESS_CITY	Город	Да	Да
ADDRESS_POSTAL_CODE	Почтовый индекс	Да	Да
ADDRESS_REGION	Район	Да	Да
ADDRESS_PROVINCE	Область	Да	Да
ADDRESS_COUNTRY	Страна	Да	Да
ADDRESS_COUNTRY_CODE	Код страны	Да	Да
CURRENCY_ID	Идентификатор валюты	Да	Да
OPPORTUNITY	Предполагаемая сумма	Да	Да
OPENED	Флаг "Доступен для всех"	Да	Да
COMMENTS	Комментарии	Да	Да
ASSIGNED_BY_ID	Ответственный	Да	Да

CREATED_BY_ID	Создан	Да	Нет
MODIFY_BY_ID	Изменен	Да	Нет
DATE_CREATE	Дата создания	Да	Нет
DATE_MODIFY	Дата изменения	Да	Нет
COMPANY_ID	Идентификатор компании	Да	Нет
CONTACT_ID	Идентификатор контакта	Да	Нет
DATE_CLOSED	Дата закрытия	Да	Нет
PHONE	PHONE	Да	Да
EMAIL	e-mail	Да	Да
WEB	веб-сайт	Да	Да
IM	Контакт в службе обмена мгновенными сообщениями	Да	Да
ORIGINATOR_ID	Идентификатор внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да	Да
ORIGIN_ID	Внешний ключ, используется для операций обмена. Идентификатор объекта внешней информационной	Да	Да

	базы. Назначение поля может меняться конечным разработчиком.		

Компании

Поле	Описание	Чтение	За
ID	Идентификатор компании. Создается автоматически и уникален в рамках БД.	Да	Нет
TITLE	Название компании. Обязательное поле..	Да	Да
COMPANY_TYPE	Тип компании. Выбирается из списка	Да	Да
LOGO	Логотип компании	Да	Да
ADDRESS	Улица, дом, корпус, строение (фактический адрес)	Да	Да
ADDRESS_2	Квартира / офис (фактический адрес)	Да	Да
ADDRESS_CITY	Город (фактический адрес)	Да	Да

ADDRESS_POSTAL_CODE	Почтовый индекс (фактический адрес)	Да	Да
ADDRESS_REGION	Район (фактический адрес)	Да	Да
ADDRESS_PROVINCE	Область (фактический адрес)	Да	Да
ADDRESS_COUNTRY	Страна (фактический адрес)	Да	Да
ADDRESS_COUNTRY_CODE	Код Страны (фактический адрес)	Да	Да
REG_ADDRESS	Улица, дом, корпус, строение (юридический адрес)	Да	Да
REG_ADDRESS_2	Квартира / офис (юридический адрес)	Да	Да
REG_ADDRESS_CITY	Город (юридический адрес)	Да	Да
REG_ADDRESS_POSTAL_CODE	Почтовый индекс (юридический адрес)	Да	Да
REG_ADDRESS_REGION	Район (юридический адрес)	Да	Да
REG_ADDRESS_PROVINCE	Область (юридический адрес)	Да	Да

REG_ADDRESS_COUNTRY	Страна (юридический адрес)	Да	Да
REG_ADDRESS_COUNTRY_CODE	Код Страны (юридический адрес)	Да	Да
ADDRESS_LEGAL	Юридический адрес	Да	Да
BANKING_DETAILS	Банковские реквизиты	Да	Да
INDUSTRY	Сфера деятельности. Выбирается из списка.	Да	Да
EMPLOYEES	Количество сотрудников. Выбирается из списка	Да	Да
CURRENCY_ID	Валюта расчетов	Да	Да
REVENUE	Годовой оборот	Да	Да
OPENED	Флаг "Доступна для всех"	Да	Да
COMMENTS	Комментарии	Да	Да
ASSIGNED_BY_ID	Идентификатор ответственного	Да	Да
CREATED_BY_ID	Создал	Да	Нет
MODIFY_BY_ID	Изменил	Да	Нет
DATE_CREATE	Дата создания	Да	Нет
DATE_MODIFY	Дата изменения	Да	Нет

LEAD_ID	Идентификатор лида	Да	Нет
ORIGINATOR_ID	Идентификатор внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да	Да
ORIGIN_ID	Внешний ключ, используется для операций обмена. Идентификатор объекта внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да	Да
PHONE	PHONE	Да	Да
EMAIL	e-mail	Да	Да
WEB	веб-сайт	Да	Да
IM	Контакт в службе обмена мгновенными сообщениями	Да	Да

Контакты

Поле	Описание	Чтение	Запис

ID	Идентификатор контакта. Создается автоматически и уникален в пределах БД	Да	Да
NAME	Имя контакта. Обязательное поле	Да	Да
SECOND_NAME	Отчество контакта. Обязательное поле	Да	Да
LAST_NAME	Фамилия Контакта. Обязательное поле	Да	Да
PHOTO	Фотография контакта	Да	Да
BIRTHDATE	Дата рождения контакта	Да	Да
TYPE_ID	Тип контакта	Да	Да
SOURCE_ID	Источник	Да	Да
SOURCE_DESCRIPTION	Подробнее об источнике	Да	Да
POST	Должность	Да	Да
ADDRESS	Улица, дом, корпус, строение	Да	Да
ADDRESS_2	Квартира, офис	Да	Да
ADDRESS_CITY	Город	Да	Да

ADDRESS_POSTAL_CODE	Почтовый индекс	Да	Да
ADDRESS_REGION	Район	Да	Да
ADDRESS_PROVINCE	Область	Да	Да
ADDRESS_COUNTRY	Страна	Да	Да
ADDRESS_COUNTRY_CODE	Код страны	Да	Да
OPENED	Флаг "Доступен для всех"	Да	Да
COMMENTS	Комментарии	Да	Да
EXPORT	Флаг разрешения экспорта	Да	Да
ASSIGNED_BY_ID	Идентификатор ответственного	Да	Да
CREATED_BY_ID	Создан	Да	Нет
MODIFY_BY_ID	Изменен	Да	Нет
DATE_CREATE	Дата создания	Да	Нет
DATE_MODIFY	Дата изменения	Да	Нет
COMPANY_ID	Идентификатор компании	Да	Да
LEAD_ID	Идентификатор лида	Да	Нет
PHONE	PHONE	Да	Да
EMAIL	e-mail	Да	Да
WEB	веб-сайт	Да	Да

IM	Контакт в службе обмена мгновенными сообщениями	Да	Да
ORIGINATOR_ID	Идентификатор внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да	Да
ORIGIN_ID	Внешний ключ, используется для операций обмена. Идентификатор объекта внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да	Да

Реквизиты

Название	Описание	Чтение
Общие реквизиты		
ID	Идентификатор реквизита. Создается автоматически и уникален в рамках БД.	Да
[dw]ENTITY_TYPE_ID[/dw] [di]Идентификаторы типов	Тип родительской сущности. Возможные типы: "Компания",	Да

сущностей возвращает метод crm.enum.ownertype[/di]	"Контакт". Обязательное поле.	
ENTITY_ID	Идентификатор родительской сущности. Обязательное поле.	Да
PRESET_ID	Идентификатор шаблона реквизитов. Обязательное поле.	Да
DATE_CREATE	Дата создания.	Да
DATE_MODIFY	Дата изменения.	Да
CREATED_BY_ID	Идентификатор создавшего реквизит.	Да
MODIFY_BY_ID	Идентификатор изменившего реквизит.	Да
NAME	Название реквизита.	Да
CODE	Символьный код реквизита.	Да
XML_ID	Внешний ключ, используется для операций обмена. Идентификатор объекта внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да
ACTIVE	Признак активности.	Да
SORT	Сортировка.	Да
RQ_NAME	Ф.И.О.	Да

RQ_FIRST_NAME	Имя.	Да
RQ_LAST_NAME	Фамилия.	Да
RQ_COMPANY_NAME	Сокращенное наименование организации.	Да
RQ_COMPANY_FULL_NAME	Полное наименование организации.	Да
RQ_COMPANY_REG_DATE	Дата государственной регистрации.	Да
RQ_DIRECTOR	Ген. директор.	Да
RQ_ACCOUNTANT	Гл. бухгалтер.	Да
RQ_CEO_NAME	ФИО первого руководителя.	Да
RQ_CEO_WORK_POS	Должность первого руководителя.	Да
RQ_CONTACT	Контактное лицо.	Да
RQ_EMAIL	E-Mail.	Да
RQ_PHONE	Телефон.	Да
RQ_FAX	Факс.	Да
RQ_IDENT_DOC	Вид документа.	Да
RQ_IDENT_DOC_SER	Серия.	Да
RQ_IDENT_DOC_NUM	Номер.	Да
RQ_IDENT_DOC_DATE	Дата выдачи.	Да
RQ_IDENT_DOC_ISSUED_BY	Кем выдан.	Да

RQ_IDENT_DOC_DEP_CODE	Код подразделения.	Да
RQ_INN	ИНН.	Да
RQ_KPP	КПП.	Да
RQ_USRLE	Handelsregisternummer (для страны DE).	Да
RQ_IFNS	ИФНС.	Да
RQ_OGRN	ОГРН.	Да
RQ_OGRNIP	ОГРНИП.	Да
RQ_OKPO	ОКПО.	Да
RQ_OKTMO	ОКТМО.	Да
RQ_OKVED	ОКВЭД.	Да
RQ_EDRPOU	ЄДРПОУ.	Да
RQ_DRFO	ДРФО.	Да
RQ_KBE	КБЕ.	Да
RQ_IIN	ИИН.	Да
RQ_BIN	БИН.	Да
RQ_VAT_PAYER	Платник ПДВ (для страны UA).	Да
RQ_VAT_ID	VAT ID (идентификационный номер (плательщика) НДС).	Да
RQ_VAT_CERT_SER	Серия свидетельства по НДС.	Да

RQ_VAT_CERT_NUM	Номер свидетельства по НДС.	Да
RQ_VAT_CERT_DATE	Дата свидетельства по НДС.	Да
RQ_RESIDENCE_COUNTRY	Страна резидента.	Да
ORIGINATOR_ID	Идентификатор внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да
Банковские реквизиты		
ID	Идентификатор реквизита. Создается автоматически и уникален в рамках БД.	Да
[dw]ENTITY_TYPE_ID[/dw] [di]Идентификаторы типов сущностей возвращает метод crm.enum.ownertype[/di]	Тип родительской сущности. По умолчанию тип: "Реквизит". Обязательное поле.	Да
ENTITY_ID	Идентификатор родительской сущности. Обязательное поле.	Да
COUNTRY_ID	Идентификатор страны.	Да
DATE_CREATE	Дата создания.	Да
DATE_MODIFY	Дата изменения.	Да
CREATED_BY_ID	Идентификатор создавшего реквизит.	Да

MODIFY_BY_ID	Идентификатор изменившего реквизит.	Да
NAME	Название реквизита.	Да
CODE	Символьный код реквизита.	Да
XML_ID	Внешний ключ, используется для операций обмена. Идентификатор объекта внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да
ACTIVE	Признак активности.	Да
SORT	Сортировка.	Да
RQ_BANK_NAME	Наименование банка.	Да
RQ_BANK_ADDR	Адрес банка.	Да
RQ_BANK_ROUTE_NUM	Bank Routing Number.	Да
RQ_BIK	БИК.	Да
RQ_MFO	МФО.	Да
RQ_ACC_NAME	Bank Account Holder Name.	Да
RQ_ACC_NUM	Bank Account Number.	Да
RQ_IIK	ИИК.	Да
RQ_ACC_CURRENCY	Валюта счёта.	Да

RQ_COR_ACC_NUM	Кор. счёт.	Да
RQ_IBAN	IBAN.	Да
RQ_SWIFT	SWIFT.	Да
RQ_BIC	BIC.	Да
COMMENTS	Комментарий.	Да
ORIGINATOR_ID	Идентификатор внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да
Шаблоны реквизитов		
ID	Идентификатор реквизита. Создается автоматически и уникален в рамках БД.	Да
[dw]ENTITY_TYPE_ID[/dw] [di]Идентификаторы типов сущностей возвращает метод crm.enum.ownertype[/di]	Тип родительской сущности. Обязательное поле.	Да
COUNTRY_ID	Страна, которой соответствует набор полей шаблона реквизита.	Да
DATE_CREATE	Дата создания.	Да
DATE_MODIFY	Дата изменения.	Да
CREATED_BY_ID	Идентификатор создавшего реквизит.	Да
MODIFY_BY_ID	Идентификатор изменившего	Да

	реквизит.	
NAME	Название реквизита.	Да
XML_ID	Внешний ключ, используется для операций обмена. Идентификатор объекта внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да
ACTIVE	Признак активности.	Да
SORT	Сортировка.	Да
Поля шаблонов реквизитов		
ID	Идентификатор реквизита. Создается автоматически и уникален в рамках реквизита.	Да
FIELD_NAME	Название поля.	Да
FIELD_TITLE	Альтернативное название поля для реквизита.	Да
SORT	Сортировка.	Да
IN_SHORT_LIST	Показывать в кратком списке.	Да
Адреса реквизитов		
TYPE_ID	Идентификатор типа адреса. Обязательное поле. Элемент	Да

	перечисления "Тип адреса".	
[dw]ENTITY_TYPE_ID[/dw] [di]Идентификаторы типов сущностей возвращает метод crm.enum.ownertype[/di]	Тип родительской сущности. Возможные типы: "Реквизит", "Компания", "Контакт", "Лид". Обязательное поле.	Да
ENTITY_ID	Идентификатор родительской сущности. Обязательное поле.	Да
ADDRESS_1	Улица, дом, корпус, строение.	Да
ADDRESS_2	Квартира / офис.	Да
CITY	Город.	Да
POSTAL_CODE	Почтовый индекс.	Да
REGION	Район.	Да
PROVINCE	Область.	Да
COUNTRY	Страна.	Да
COUNTRY_CODE	Код страны.	Да

Дела

Поле	Описание	Тип
ASSOCIATED_ENTITY_ID	Идентификатор связанной с делом сущности	integer
AUTOR_ID	Создатель дела	user

AUTOCOMPLETE_RULE	Автозаполнение	integer
BINDINGS	Привязки	crm_activity_binding
COMMUNICATIONS		crm_activity_communica
COMPLETED	Завершено	char
CREATED	Создано	datetime
DEADLINE	Срок исполнения	datetime
DESCRIPTION	Описание	string
DESCRIPTION_TYPE	Тип описания	crm.enum.contenttype
DIRECTION	Направление дела: входящее/ исходящее.	crm.enum.activitydirecti
EDITOR_ID	Кто изменил	user
END_TIME	Время завершения	datetime
FILES	Добавленные файлы	diskfile
ID	Идентификатор дела	integer

LAST_UPDATE	Дата последнего обновления	datetime
LOCATION	Местоположение.	string
NOTIFY_TYPE	Тип уведомлений	crm.enum.activitynotifyt
NOTIFY_VALUE		integer
ORIGINATOR_ID	Идентификатор источника данных	string
ORIGIN_ID	Идентификатор элемента в источнике данных	string
ORIGIN_VERSION	Оригинальная версия	string
OWNER_ID	Собственник	integer

OWNER_TYPE_ID	Тип собственника	crm.enum.ownertype
PRIORITY	Приоритет	crm.enum.activitypriority
PROVIDER_DATA		string
PROVIDER_GROUP_ID		string
PROVIDER_ID	Идентификатор провайдера	string
PROVIDER_TYPE_ID	Идентификатор типа провайдера	string
PROVIDER_PARAMS		object
RESPONSIBLE_ID	Ответстветнный	user
RESULT_CURRENCY_ID		string
RESULT_MARK		integer
RESULT_SOURSE_ID		string
RESULT_STATUS		integer
RESULT_STREAM	Статистика отчётов	integer
RESULT_SUM		double
RESULT_VALUE		double
SETTINGS	Настройки	object
START_TIME	Время начала выполнения	datetime
STATUS	Статус	crm_enum_activitystatus
SUBJECT	Субьект	string

TYPE_ID	Тип	crm_enum_activitytype
WEBDAV_ELEMENTS	Добавленные файлы	diskfile

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Константы CRM

Тип сущности CRM

Тип сущности	Числовой идентификатор (entityTypeId)	Символьный код (entityTypeName)
Лид	1	LEAD
Сделка	2	DEAL
Контакт	3	CONTACT
Компания	4	COMPANY
Счет (старый)	5	INVOICE
Счет (новый)	31	SMART_INVOICE
Предложение	7	QUOTE
Реквизит	8	REQUISITE

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Частые кейсы::Как отправить e-mail клиенту от имени сотрудника с указанием ящика сотрудника

Пример получает e-mail ответственного за контакт и создаёт активити отправки письма с моментальной отправкой.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

```
<?
$contactID = 1;
$resultContact = CRest::call(
        'crm.contact.get',
        [
                'id' => $contactID
        ]
);
$resultActivity = [];
if (!empty($resultContact['result']
['ASSIGNED_BY_ID']) &&
!empty($resultContact['result']['EMAIL']))
{
        $resultUser = CRest::call(
                'user.get',
```

```
                [
                        'filter' => [
                                'ID' =>
$resultContact['result']['ASSIGNED_BY_ID']
                        ]
                ]
        );

        if ($resultUser['result'])
        {
                $contactEmail =
reset($resultContact['result']['EMAIL']);
                $staff =
reset($resultUser['result']);
                if
(!empty($contactEmail['VALUE']) &&
!empty($staff['EMAIL']))
                {
                        $resultActivity =
CRest::call(

'crm.activity.add',
                                [

'fields' => [

"SUBJECT" => "subject email now",

"DESCRIPTION" => "body email now",

"DESCRIPTION_TYPE" => 3,//text,html,bbCode
type id in:
CRest::call('crm.enum.contenttype');

"COMPLETED" => "Y",//send now

"DIRECTION" => 2,//
```

```
CRest::call('crm.enum.activitydirection');

"OWNER_ID" => $contactID,

"OWNER_TYPE_ID" => 3, //
CRest::call('crm.enum.ownertype');

"TYPE_ID" => 4, //
CRest::call('crm.enum.activitytype');

"COMMUNICATIONS" => [

[

'VALUE' => $contactEmail['VALUE'],

'ENTITY_ID' => $contactID,

'ENTITY_TYPE_ID' => 3//
CRest::call('crm.enum.ownertype');

]

],

"START_TIME" => date("Y-m-d H:i:s", time()),

"END_TIME" => date("Y-m-d H:i:s", time() +
3600),

"RESPONSIBLE_ID" => $staff['ID'],

'SETTINGS' => [

'MESSAGE_FROM' => implode(

' ',
```

```
[$staff['NAME'], $staff['LAST_NAME'], '<' .
$staff['EMAIL'] . '>']

),


],
                                        ]
                                ]
                        );
                }
        }
}
if (!empty($resultActivity['result']))
{
        echo json_encode(['message' =>
'Activity add']);
}
elseif
(!empty($resultActivity['error_description']
))
{
        echo json_encode(['message' =>
'Activity not added: ' .
$resultActivity['error_description']]);
}
else
{
        echo json_encode(['message' =>
'Activity not added']);
}
?>
```

© «Битрикс», 2001-2008, «1С-

Частые кейсы::Получение воронки заданного направления с семантикой каждой стадии сделки

Пример выводит все существующие направления сделок с семантикой по каждой стадии.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

```
$arCategory = [];
$result =
CRest::call('crm.dealcategory.list');
if (!empty($result['result']))
{
        $arCategory =
array_column($result['result'], 'NAME',
'ID');

}
$result =
CRest::call('crm.dealcategory.default.get');
//get name default deal category
if (!empty($result['result']))
{
        $arCategory[$result['result']['ID']]
```

```
= $result['result']['NAME'];
}
foreach ($arCategory as $id => $name):
        if ($id > 0)
        {

                $entity_id = 'DEAL_STAGE_' .
$id;
        }
        else
        {
                $entity_id = 'DEAL_STAGE';
        }
        $resultDeal =
CRest::call('crm.status.list', ['filter' =>
['ENTITY_ID' => $entity_id]]);

        if (!empty($resultDeal['result'])):?
>
                <table>
                        <caption><?=$name?>
</caption>
                        <thead>
                        <tr>
                                <th>STATUS
ID</th>

<th>NAME</th>

<th>SEMANTICS</th>
                        </tr>
                        </thead>
                        <tbody>
                        <? foreach
($resultDeal['result'] as $item): ?>
                        <tr <?=
(!empty($item['EXTRA']['COLOR']) ? '
```

```
style="color:' . $item['EXTRA']['COLOR'] .
'"' : '');?>>
                                <td><?
=$item['STATUS_ID']?></td>
                                <td><?
=$item['NAME']?></td>
                                <td><?
=$item['EXTRA']['SEMANTICS']?></td>
                        <tr>
                                <?
endforeach;
                                ?>
                        </tbody>
                </table>
        <? endif; ?>
<? endforeach; ?>
```

Частые кейсы::Генератор документов в crm: создание "путевого листа" по данным сделки

Пример добавления своего шаблона документа и создания на его основе готового документа с данными, например, сделки. В примере **template.docx** - шаблон документа.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

```
$filePath = __DIR__ . '/template.docx'; //
path to template local file
$iDealID = 1;//deal ID
$sDocName = 'Demo Packing Sheet UK';

$resNum = CRest::call(

'crm.documentgenerator.numerator.add',
        [
                'fields' => [
                        'name' => 'Rest
Numerator',
                        'template' =>
'{NUMBER}',
                ]
        ]
);
```

```
if (!empty($resNum['result']['numerator']
['id']))
{
        $resTemplate = CRest::call(

'crm.documentgenerator.template.add',
                [
                        'fields' => [
                                'name' =>
$sDocName,

'numeratorId' => $resNum['result']
['numerator']['id'],//
crm.documentgenerator.numerator.add
                                'region' =>
'uk',//eu,de,ua,by,ru
                                'users' => [

'UA'//User All
                                ],

'entityTypeId' => ['2'],//2 is deal in
CRest::call('crm.enum.ownertype');
                                'file' =>
base64_encode(file_get_contents($filePath))
                        ]
                ]
        );
        if (!empty($resTemplate['result']
['template']['id']))
        {
                $resDoc = CRest::call(

'crm.documentgenerator.document.add',
                        [
                                'templateId'
=> $resTemplate['result']['template']['id'],
```

```
'entityTypeId' => '2',//2 is deal in
CRest::call('crm.enum.ownertype');
                                'entityId'
=> $iDealID,
                        ]
                );
        }
}

if (!empty($resDoc['result']))
{
        echo json_encode(['message' => 'Docx
creat']);
}
elseif
(!empty($resDoc['error_description']))
{
        echo json_encode(['message' => 'Docx
not created: ' .
$resDoc['error_description']]);
}
else
{
        echo json_encode(['message' => 'Docx
not created']);
}
```

Частые кейсы::Встройка в лид в виде пользовательского свойства

Пример добавления в карточку лида собственного пользовательского свойства. Работа примера происходит следующим образом: после первого взаимодействия с свойством в карточке редактирования лида всегда, даже в режиме просмотра, будет подгружаться обработчик от приложения. Обработчик производит запрос к внешнему API для получения региона и оператора данного телефона на территории РФ.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

Код установки свойств (вызывается один раз), где **$handlerUrl** - путь до файла обработчика свойства.

```
<?
$handlerUrl =
'https://yourdomain.yyy/handler.php';
$type = 'phone_data';
$propCode = 'PHONE_DATA';//max length with
prefix UF_CRM_ 20 char

$resultAddPropType = CRest::call(
        'userfieldtype.add',
        [
                'USER_TYPE_ID' => $type,
```

```
                'HANDLER' => $handlerUrl,
                'TITLE' => 'custom type
title',
                'DESCRIPTION' => 'custom
description '.$type
        ]
);

if ($resultAddPropType['result'] == true)
{
        echo 'property type ' . $type . '
has been added successful <br>';
        $resultAddProp = CRest::call(
                'crm.lead.userfield.add',
                [
                        'fields' => [

'USER_TYPE_ID' => $type,
                                'FIELD_NAME'
=> $propCode,
                                'XML_ID' =>
$propCode,
                                'MANDATORY'
=> 'N',

'SHOW_IN_LIST' => 'Y',

'EDIT_IN_LIST' => 'Y',

'EDIT_FORM_LABEL' => 'My string',

'LIST_COLUMN_LABEL' => 'My string
description',
                                'SETTINGS'
=> []
                        ]
                ]
```

```
        );
        if ($resultAddProp['error'])
        {
                echo $resultAddProp['error']
. ': ' .
$resultAddProp['error_description'];
        }
        else
        {
                echo 'property ' . $propCode
. ' has been added successful <br>';
        }
}
elseif ($resultAddPropType['error'])
{
        echo $resultAddPropType['error'] .
': ' .
$resultAddPropType['error_description'];
}
?>
```

Файл обработчика, который вы указали в переменной **$handlerUrl** в коде выше:

```
<?
$placementOptions =
isset($_REQUEST['PLACEMENT_OPTIONS']) ?
json_decode($_REQUEST['PLACEMENT_OPTIONS'],
true) : array();
if ($_REQUEST['PLACEMENT'] ==
'USERFIELD_TYPE'):
        $value =
htmlspecialchars($placementOptions['VALUE'])
;
        if ($placementOptions['ENTITY_ID']
== 'CRM_LEAD' &&
```

```
$placementOptions['ENTITY_VALUE_ID'] > 0)
        {
                $result = CRest::call(
                        'crm.lead.list',
                        [
                                'filter' =>
['ID' =>
intVal($placementOptions['ENTITY_VALUE_ID'])
],
                                'select' =>
['ID', 'PHONE']
                        ]
                );
                if (!empty($result['result']
[0]['PHONE'][0]['VALUE']))
                {
                        $value =
trim($result['result'][0]['PHONE'][0]
['VALUE']);
                        $data =
file_get_contents('http://api.bitroid.info/p
hone/?q='.$value);
                        if($data)
                        {
                                $valueData =
json_decode($data,true);

if(!$valueData['error'])
                                {

$value = implode(', ',
[$valueData['org'],$valueData['region']]);
                                }
                                else
                                {

$value = 'error: '.$valueData['error'];
```

```
                                        }
                                }
                                else
                                {
                                        $value = 'no
data in base'.$value;
                                }
                        }
                        else
                        {
                                $value = 'no data';
                        }
                }


                ?>
                <!DOCTYPE html>
                <html>
                        <head>
                                <script
src="//api.bitrix24.com/api/v1/dev/">
</script>
                        </head>
                        <body style="margin: 0;
padding: 0; background-color: <?
=$placementOptions['MODE'] === 'edit' ?
'#fff'
                                : '#f9fafb'?>;">
                                <?
                                if
($placementOptions['MODE'] === 'edit'): ?>
                                        <input
type="text" style="width: 90%;" value='<?
=$value?>' onkeyup="setValue(this.value)">
                                        <script>

function setValue(value)
```

```
                                                {
BX24.placement.call('setValue', value);
                                                }


BX24.placement.call('setValue', '<?=$value?
>');
                                    </script>

                            <? else: ?>
                                    <?=$value?>
                            <? endif;
                            ?>
                    </body>
            </html>
<? endif;?>
```

Простое добавление лида

Пример позволит разместить на любой странице вашего сайта форму, заполняя которую в битрикс24 будет создаваться новый лид.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

1. Создаём форму на нужной странице:

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1
/jquery.min.js"></script>
<script>
$(document).ready(function() {
    $('#form_to_crm').on( 'submit', function(el)
{//event submit form
        el.preventDefault();//the default action of the
event will not be triggered
        var formData = $(this).serialize();
        $.ajax({
            'method': 'POST',
            'dataType': 'json',
            'url': 'form.php',
            'data':  formData,
            success: function(data){//success callback
                alert(data.message);
            }
        });
    });
});
</script>

<form id="form_to_crm">
        <input type="text" name="NAME"
placeholder="Name" required>
        <input type="text" name="LAST_NAME"
placeholder="Last name">
        <input type="text" name="PHONE"
```

```
placeholder="Phone">
        <input type="text" name="EMAIL" placeholder="E-
mail">
        <input type="submit" value="Submit">
</form>
```

2. Создаём файл **form.php**, для сохранения заполненных форм:

```
<?
        $sName = htmlspecialchars($_POST["NAME"]);
        $sLastName =
htmlspecialchars($_POST["LAST_NAME"]);
        $sPhone = htmlspecialchars($_POST["PHONE"]);
        $sEmail = htmlspecialchars($_POST["EMAIL"]);

        //Форматируем телефон и почту для сохранения
        $arPhone = (!empty($sPhone)) ?
array(array('VALUE' => $sPhone, 'VALUE_TYPE' =>
'WORK')) : array();
        $arEmail = (!empty($sEmail)) ?
array(array('VALUE' => $sEmail, 'VALUE_TYPE' =>
'HOME')) : array();

        $result = CRest::call(
                'crm.lead.add',
                [
                'fields' =>[
                        'TITLE' => 'From the site:
'.implode(' ',[$sName,$sLastName]),//*Lead Name[string]
                        'NAME' => $sName,//Name[string]
                        'LAST_NAME' =>
$sLastName,//Last name[string]
                        'PHONE' =>
$arPhone,//Phone[crm_multifield]
                        'EMAIL' => $arEmail,//E-
mail[crm_multifield]
                        //'HONORIFIC' =>
'',//Salutation[string]
                        //'SECOND_NAME' => '',//Second
name[string]
                        //'BIRTHDATE' => '',//Date of
birth[date]
                        //'COMPANY_TITLE' =>
'',//Company name[string]
                        //'SOURCE_ID' =>
'',//Source[crm_status {CALL:"Call", EMAIL:"E-Mail",
WEB:"Website",
//ADVERTISING:"Advertising", PARTNER:"Existing Client",
RECOMMENDATION:"By Recommendation",
// TRADE_SHOW:"Show/Exhibition", WEBFORM:"CRM form",
CALLBACK:"Callback", RC_GENERATOR:"Sales boost",
```

```
STORE:"Online Store", OTHER:"Other"}]
// CRest::call('crm.status.list',['filter'=>
['ENTITY_ID'=>'SOURCE']]);
                        //'SOURCE_DESCRIPTION' =>
'',//Source information[string]
                        //'STATUS_ID' =>
'',//Status[crm_status {NEW:"Unassigned",
IN_PROCESS:"In Progress", PROCESSED:"Processed",
// CONVERTED:"Good lead", JUNK:"Junk Lead"}]//
CRest::call('crm.status.list',['filter'=>
['ENTITY_ID'=>'STATUS']]);
                        //'STATUS_DESCRIPTION' =>
'',//Status information[string]
                        //'STATUS_SEMANTIC_ID' =>
'',//Status details[string]
                        //'POST' =>
'',//Position[string]
                        //'ADDRESS' =>
'',//Address[string]
                        //'ADDRESS_2' => '',//Address
(line 2)[string]
                        //'ADDRESS_CITY' =>
'',//City[string]
                        //'ADDRESS_POSTAL_CODE' =>
'',//Zip[string]
                        //'ADDRESS_REGION' =>
'',//Region[string]
                        //'ADDRESS_PROVINCE' =>
'',//State / Province[string]
                        //'ADDRESS_COUNTRY' =>
'',//Country[string]
                        //'ADDRESS_COUNTRY_CODE' =>
'',//Country Code[string]
                        //'CURRENCY_ID' =>
'',//Currency[crm_currency]//
CRest::call('crm.currency.list');
                        //'OPPORTUNITY' =>
'',//Total[double]
                        //'OPENED' => '',//Available to
everyone[char]
                        //'COMMENTS' =>
'',//Comment[string]
                        //'HAS_PHONE' => '',//Has
phone[char]
                        //'HAS_EMAIL' => '',//Has
email[char]
                        //'HAS_IMOL' => '',//Has Open
Channel[char]
                        //'ASSIGNED_BY_ID' =>
'',//Responsible person[user]
                        //'CREATED_BY_ID' =>
```

```
'',//Created by[user]
                        //'MODIFY_BY_ID' =>
'',//Modified by[user]
                        //'DATE_CREATE' => '',//Created
on[datetime]
                        //'DATE_MODIFY' =>
'',//Modified on[datetime]
                        //'COMPANY_ID' =>
'',//Company[crm_company]//
CRest::call('crm.company.list');
                        //'CONTACT_ID' =>
'',//Contact[crm_contact]//
CRest::call('crm.contact.list');
                        //'IS_RETURN_CUSTOMER' =>
'',//Repeat lead[char]
                        //'DATE_CLOSED' =>
'',//Completed on[datetime]
                        //'ORIGINATOR_ID' =>
'',//External source[string]
                        //'ORIGIN_ID' => '',//Item ID
in data source[string]
                        //'UTM_SOURCE' => '',//Ad
system[string]
                        //'UTM_MEDIUM' =>
'',//Medium[string]
                        //'UTM_CAMPAIGN' => '',//Ad
campaign UTM[string]
                        //'UTM_CONTENT' =>
'',//Campaign contents[string]
                        //'UTM_TERM' => '',//Campaign
search term[string]
                        //'WEB' =>
'',//Website[crm_multifield]
                        //'IM' =>
'',//Messenger[crm_multifield]
                ]
        ]);
        if(!empty($result['result'])){
                echo json_encode(['message' => 'Lead
add']);
        }elseif(!empty($result['error_description'])){
                echo json_encode(['message' => 'Lead
not added: '.$result['error_description']]);
        }else{
                echo json_encode(['message' => 'Lead
not added']);
        }
?>
```

Добавление лида с учетом режимов CRM (простой или классический режим)

Если CRM работает в простом режиме происходит добавление дела в сделку, в классическом режиме в лид.

Внимание! Для использования данного примера необходимо настроить работу класса *CRest* и подключить файл **crest.php** в файлах, где используется данный класс. Подробнее.

1. Создаём форму на нужной странице:

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1
/jquery.min.js"></script>
        <script>
        $(document).ready(function() {
            $('#form_to_crm').on( 'submit',
function(el) {//event submit form
                el.preventDefault();//the default
action of the event will not be triggered
                var formData = $(this).serialize();
                $.ajax({
                    'method': 'POST',
                    'dataType': 'json',
                    'url': 'form.php',
                    'data':  formData,
                    success: function(data){//success
callback
                        alert(data.message);
                    }
                });
            });
        });
        </script>
```

```
        <form id="form_to_crm">
                <input type="text" name="NAME"
placeholder="Name" required>
                <input type="text" name="LAST_NAME"
placeholder="Last name">
                <input type="text" name="PHONE"
placeholder="Phone">
                <input type="text" name="EMAIL"
placeholder="E-mail">
                <input type="submit" value="Submit">
        </form>
```

2. Создаём файл form.php, для сохранения заполненных форм:

```
<?
        $sName          =
htmlspecialchars($_POST["NAME"]);
        $sLastName    =
htmlspecialchars($_POST["LAST_NAME"]);
        $sPhone         =
htmlspecialchars($_POST["PHONE"]);
        $sEmail         =
htmlspecialchars($_POST["EMAIL"]);

        $arData = [
                'add_lead' => [
                        'method' => 'crm.lead.add',
                        'params' => [
                                'fields'    =>  [
                                        'TITLE' =>
'From the site: ' . implode(' ', [$sName, $sLastName]),
                                        'NAME' =>
(!empty($sName)) ? $sName : 'Empty name',//if simple
mode crm NAME or LAST_NAME required for converting to
contact
                                        'LAST_NAME' =>
$sLastName,
                                        'PHONE' =>
(!empty($sPhone)) ? array(array('VALUE' => $sPhone,
'VALUE_TYPE' => 'HOME')) : array(),
                                        'EMAIL' =>
(!empty($sEmail)) ? array(array('VALUE' => $sEmail,
'VALUE_TYPE' => 'HOME')) : array()
                                ]
                        ]
                ],
                'get_lead' => [
                        'method' => 'crm.lead.get',
                        'params' => [
                                'id' =>
'$result[add_lead]'
```

```
                        ]
                ],
        ];
        $result = CRest::callBatch($arData);

        if(empty($result['result']['result_error']
['add_lead']) && !empty($result['result']['result']
['get_lead'])){
                //if status_id == converted on add then
is simple mode crm
                if($result['result']['result']
['get_lead']['STATUS_ID'] == 'CONVERTED'){
                        $resultDeal =
CRest::call('crm.deal.list',
                                [
                                        'filter'=>[

'LEAD_ID' => $result['result']['result']['add_lead']
                                        ]
                                ]);
                        if(!empty($resultDeal['result']
['0']['ID'])){
                                $resultActivity =
CRest::call('crm.activity.add',//call within an hour
                                        [

'fields' =>[

"OWNER_TYPE_ID"     => 2,//2 - is deal in
CRest::call('crm.enum.ownertype');

"TYPE_ID"           => 2,//2 - is call in
CRest::call('crm.enum.activitytype');

"OWNER_ID"          => $resultDeal['result']['0']
['ID'],//entity id

"COMMUNICATIONS"    => [['VALUE' => $sPhone,'TYPE' =>
'PHONE']],

"START_TIME" =>  date("Y-m-d H:i:s",time()),

"END_TIME" => date("Y-m-d H:i:s",time()+3600),

"SUBJECT" => "Call back",

"DESCRIPTION" => "Call within an hour",
                                                ]
                                        ]);
                        }
```

```
                }else{
                        $resultActivity =
CRest::call('crm.activity.add',//call within an hour
                                [
                                        'fields' =>[

"OWNER_TYPE_ID"     => 1,//1 - is lead in
CRest::call('crm.enum.ownertype');

"TYPE_ID"           => 2,//2 - is call in
CRest::call('crm.enum.activitytype');

"OWNER_ID"          => $result['result']['result']
['add_lead'],//entity id

"COMMUNICATIONS"    => [['VALUE' => $sPhone,'TYPE' =>
'PHONE']],

"START_TIME" =>  date("Y-m-d H:i:s",time()),

"END_TIME" => date("Y-m-d H:i:s",time()+3600),

"SUBJECT" => "Call back",

"DESCRIPTION" => "Call within an hour",
                                        ]
                                ]);
                }

        }else{
                if(!empty($result['result']
['result_error']['add_lead']))
                        $arResult[] =  'error add lead:
'.$result['result']['result_error']['add_lead']
['error_description'];
                if(!empty($result['result']
['result_error']['get_lead']))
                        $arResult[] =  'error get new
lead: '.$result['result']['result_error']['get_lead']
['error_description'];
        }
        ?>
```

Добавление повторного лида

Пример разместит на странице сайта форму CRM. По заполнении формы в Битрикс24 будет создаваться новый лид как повторный с привязкой к нему контакта или компании из старых лидов.

Внимание! Для использования примера необходимо настроить работу класса *CRest* и подключить файл **crest.php** в файлах, где используется данный класс. Подробнее.

1. Создаём форму на нужной странице:

```
 <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1
/jquery.min.js"></script>
        <script>
        $(document).ready(function() {
            $('#form_to_crm').on( 'submit',
function(el) {//event submit form
                el.preventDefault();//the default
action of the event will not be triggered
                var formData = $(this).serialize();
                $.ajax({
                    'method': 'POST',
                    'dataType': 'json',
                    'url': 'form.php',
                    'data':  formData,
                    success: function(data){//success
callback
                        alert(data.message);
                    }
                });
            });
        });
        </script>

        <form id="form_to_crm">
                <input type="text" name="NAME"
placeholder="Name" required>
```

```
                <input type="text" name="LAST_NAME"
placeholder="Last name">
                <input type="text" name="PHONE"
placeholder="Phone">
                <input type="text" name="EMAIL"
placeholder="E-mail">
                <input type="submit" value="Submit">
        </form>
```

2. Создаём файл **form.php**, для сохранения заполненных форм:

```
<?
        $sName = htmlspecialchars($_POST["NAME"]);
        $sLastName =
htmlspecialchars($_POST["LAST_NAME"]);
        $sPhone = htmlspecialchars($_POST["PHONE"]);
        $sEmail = htmlspecialchars($_POST["EMAIL"]);

        $arFields = [
                'TITLE'         => 'From the site: ' .
implode(' ', [$sName, $sLastName]),
                'NAME'          => (!empty($sName)) ?
$sName : 'Empty name',//if simple mode crm NAME or
LAST_NAME required for converting to contact
                'LAST_NAME'   => $sLastName,
                'PHONE'         => (!empty($sPhone)) ?
array(array('VALUE' => $sPhone, 'VALUE_TYPE' =>
'HOME')) : array(),
                'EMAIL'         => (!empty($sEmail)) ?
array(array('VALUE' => $sEmail, 'VALUE_TYPE' =>
'HOME')) : array()
        ];

        $arLeadDuplicate = [];
        if(!empty($sPhone)){//search duplicate by phone
                $arResultDuplicate =
CRest::call('crm.duplicate.findbycomm',[
                        "entity_type" => "LEAD",
                        "type" => "PHONE",
                        "values" => array($sPhone)
                ]);
                if(!empty($arResultDuplicate['result']
['LEAD'])){
                        $arLeadDuplicate = array_merge
($arLeadDuplicate,$arResultDuplicate['result']
['LEAD']);
                }
        }

        if(!empty($sEmail)) {//search duplicate by
```

```
email
                $arResultDuplicate =
CRest::call('crm.duplicate.findbycomm', [
                        "entity_type" => "LEAD",
                        "type" => "EMAIL",
                        "values" => [$sEmail]
                ]);
                if(!empty($arResultDuplicate[ 'result'
][ 'LEAD' ])) {
                        $arLeadDuplicate =
array_merge($arLeadDuplicate, $arResultDuplicate[
'result' ][ 'LEAD' ]);
                }
        }

        if(!empty($arLeadDuplicate)){//get converted
duplicate lead and filling $arFields COMPANY_ID or
CONTACT_ID
                $arDuplicateLead =
CRest::call('crm.lead.list',[
                        "filter" => [
                                '=ID' =>
$arLeadDuplicate,
                                'STATUS_ID' =>
'CONVERTED',
                        ],
                        'select' => [
                                'ID', 'COMPANY_ID',
'CONTACT_ID'
                        ]
                ]);

                if(!empty($arDuplicateLead['result'])){
                        $sCompany =
reset(array_diff(array_column($arDuplicateLead['result'
],'COMPANY_ID','ID'),['']));
                        $sContact =
reset(array_diff(array_column($arDuplicateLead['result'
],'CONTACT_ID','ID'),['']));
                        if($sCompany > 0)
                                $arFields['COMPANY_ID']
= $sCompany;
                        if($sContact > 0)
                                $arFields['CONTACT_ID']
= $sContact;
                }
        }

        $result = CRest::call('crm.lead.add',
                [
                        'fields'    =>  $arFields
```

```
                ]
        );
        if(!empty($result['result'])){
                echo json_encode(['message' => 'Lead
add']);
        }elseif(!empty($result['error_description'])){
                echo json_encode(['message' => 'Lead
not added: '.$result['error_description']]);
        }else{
                echo json_encode(['message' => 'Lead
not added']);
        }
?>
```

Простое добавление контакта

Пример размещения на странице сайта формы по заполнению которой в Битрикс24 создаётся новый контакт.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

1. Создаём форму на нужной странице:

```
<form id="form_to_crm">
                        <input type="text" name="NAME"
placeholder="Name" required>
                        <input type="text"
name="LAST_NAME" placeholder="Last name">
                        <input type="text" name="EMAIL"
placeholder="E-mail">
                        <input type="text" name="PHONE"
placeholder="Phone">
                        <input type="submit"
value="Submit">
                </form>
                <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1
/jquery.min.js"></script>
                <script>
                $(document).ready(function() {
                    $('#form_to_crm').on( 'submit',
function(el) {//event submit form
                        el.preventDefault();//the
default action of the event will not be triggered
                        var formData =
$(this).serialize();
                        $.ajax({
                            'method': 'POST',
                            'dataType': 'json',
                            'url': 'form.php',
                            'data':  formData,
```

```
                                success: function(data)
{//success callback
                                    alert(data.message);
                                }
                            });
                        });
                    });
                    </script>
```

2. Создаём файл **form.php**, для сохранения заполненных форм:

```
<?
                                    $sName =
htmlspecialchars($_POST["NAME"]);

$sLastName =
htmlspecialchars($_POST["LAST_NAME"]);
                                    $sPhone =
htmlspecialchars($_POST["PHONE"]);
                                    $sEmail =
htmlspecialchars($_POST["EMAIL"]);

                                    $arPhone
= (!empty($sPhone)) ? array(array('VALUE'
=> $sPhone, 'VALUE_TYPE' => 'WORK')) :
array();
                                    $arEmail
= (!empty($sEmail)) ? array(array('VALUE'
=> $sEmail, 'VALUE_TYPE' => 'HOME')) :
array();

                                    $result =
CRest::call(

'crm.contact.add',
                                                [

'fields' =>[

'NAME' => $sName,//*First name[string]
```

```
'LAST_NAME' => $sLastName,//*Last
name[string]

'PHONE' =>
$arPhone,//Phone[crm_multifield]

'EMAIL' => $arEmail,//E-
mail[crm_multifield]

//"HONORIFIC" =>
'',//Salutation[crm_status
{HNR_EN_1:"Mr.", HNR_EN_2:"Mrs.",
HNR_EN_3:"Ms.", HNR_EN_4:"Dr."}]
// CRest::call('crm.status.list',
['filter'=>['ENTITY_ID'=>'HONORIFIC']]);

//"SECOND_NAME" => '',//*Second
name[string]

//"PHOTO" => '',//Photo[file]

//"BIRTHDATE" => '',//Date of birth[date]

//"TYPE_ID" => '',//Contact
type[crm_status {CLIENT:"Clients",
SUPPLIER:"Suppliers", PARTNER:"Partners",
OTHER:"Other"}]
// CRest::call('crm.status.list',
['filter'=>
['ENTITY_ID'=>'CONTACT_TYPE']]);

//"SOURCE_ID" => '',//Source[crm_status
{CALL:"Call", EMAIL:"E-Mail",
WEB:"Website", ADVERTISING:"Advertising",
//PARTNER:"Existing Client",
RECOMMENDATION:"By Recommendation",
```

```
TRADE_SHOW:"Show/Exhibition",
WEBFORM:"CRM form", CALLBACK:"Callback",
//RC_GENERATOR:"Sales boost",
STORE:"Online Store", OTHER:"Other"}]
// CRest::call('crm.status.list',
['filter'=>['ENTITY_ID'=>'SOURCE']]);

//"SOURCE_DESCRIPTION" =>
'',//Description[string]

//"POST" => '',//Position[string]

//"ADDRESS" => '',//Address[string]

//"ADDRESS_2" => '',//Address (line 2)
[string]

//"ADDRESS_CITY" => '',//City[string]

//"ADDRESS_POSTAL_CODE" =>
'',//Zip[string]

//"ADDRESS_REGION" => '',//Region[string]

//"ADDRESS_PROVINCE" => '',//State /
Province[string]

//"ADDRESS_COUNTRY" =>
'',//Country[string]

//"ADDRESS_COUNTRY_CODE" => '',//Country
Code[string]

//"COMMENTS" => '',//Comment[string]

//"OPENED" => '',//Available to
everyone[char]
```

```
//"EXPORT" => '',//Mark for export[char]

//"HAS_PHONE" => '',//Has phone[char]

//"HAS_EMAIL" => '',//Has email[char]

//"HAS_IMOL" => '',//Has Open
Channel[char]

//"ASSIGNED_BY_ID" => '',//Responsible
person[user]

//"CREATED_BY_ID" => '',//Created
by[user]

//"MODIFY_BY_ID" => '',//Modified
by[user]

//"DATE_CREATE" => '',//Created
on[datetime]

//"DATE_MODIFY" => '',//Modified
on[datetime]

//"COMPANY_ID" =>
'',//Company[crm_company]//
CRest::call('crm.company.list');

//"COMPANY_IDS" =>
'',//COMPANY_IDS[crm_company]//
CRest::call('crm.company.list');

//"LEAD_ID" => '',//Lead[crm_lead]

//"ORIGINATOR_ID" => '',//External
source[string]
```

```
//"ORIGIN_ID" => '',//Item ID in data
source[string]

//"ORIGIN_VERSION" => '',//Original
version[string]

//"FACE_ID" => '',//FaceID
connection[integer]

//"UTM_SOURCE" => '',//Ad system[string]

//"UTM_MEDIUM" => '',//Medium[string]

//"UTM_CAMPAIGN" => '',//Ad campaign
UTM[string]

//"UTM_CONTENT" => '',//Campaign
contents[string]

//"UTM_TERM" => '',//Campaign search
term[string]

//"WEB" => '',//Website[crm_multifield]

//"IM" => '',//Messenger[crm_multifield]

]

]);

if(!empty($result['result'])){

echo json_encode(['message' => 'Contact
add']);

}elseif(!empty($result['error_description
```

```
'])){

echo json_encode(['message' => 'Contact
not added:
'.$result['error_description']]);
                                }else{

echo json_encode(['message' => 'Contact
not added']);
                                }
                ?>
```

Добавление контакта с реквизитами

Пример размещения на странице сайта формы, после заполнения которой в Битрикс24 создаётся новый контакт с прикреплением реквизитов.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

1. Создайте форму на нужной странице:

```
<?
        $arAddressFields =
CRest::call('crm.address.fields',[]);
        /*
         crm.address.fields return:
https://dev.1c-
bitrix.ru/rest_help/crm/requisite/requisi
te_fields.php#address
        */
        $arRequisiteType =
CRest::call('crm.requisite.preset.list',
                [
                        'select'=>[
                                "ID",
"NAME"
                        ]
                ]);
```

```
if(!empty($arRequisiteType['result'])):
                $arRequisiteType =
array_column($arRequisiteType['result'],'
NAME','ID');
                //unset system address
fields

unset($arAddressFields['result']
['TYPE_ID']);

unset($arAddressFields['result']
['ENTITY_TYPE_ID']);

unset($arAddressFields['result']
['ENTITY_ID']);
                //unset uninteresting
address fields

unset($arAddressFields['result']
['COUNTRY_CODE']);

unset($arAddressFields['result']
['ANCHOR_TYPE_ID']);

unset($arAddressFields['result']
['ANCHOR_ID']);
                ?>
                <form id="form_to_crm">
                        <select
name="REQ_TYPE" required>
                                <option
value="" disabled
selected>Select</option>
                                <?
foreach($arRequisiteType as $id=>$name):?
>
```

```
<option value="<?=$id?>"><?=$name?>
</option>
                                <?
endforeach;?>
                        </select>
                        <input
type="text" name="NAME"
placeholder="Name" required>
                        <input
type="text" name="LAST_NAME"
placeholder="Last name">
                        <input
type="text" name="PHONE"
placeholder="Phone">
                        <?
if(is_array($arAddressFields['result'])):
?>
                                <?
foreach($arAddressFields['result'] as
$key=>$arField):?>

<input type="text" name="ADDRESS[<?=$key?
>]" placeholder="<?=$arField['title']?>"
<?=
($arField['isRequired'])?'required':'';?
>>
                                <?
endforeach;?>
                        <?endif;?>
                        <input
type="submit" value="Submit">
                </form>
        <?else:?>
                <?='No requisite
types.';?>
        <?endif;?>
        <script
```

```
src="https://ajax.googleapis.com/ajax/lib
s/jquery/3.3.1/jquery.min.js"></script>
        <script>
        $(document).ready(function() {
            $('#form_to_crm').on(
'submit', function(el) {//event submit
form
                el.preventDefault();//the
default action of the event will not be
triggered
                var formData =
$(this).serialize();
                $.ajax({
                    'method': 'POST',
                    'dataType': 'json',
                    'url': 'form.php',
                    'data':  formData,
                    success:
function(data){//success callback

alert(data.message);
                    }
                });

            });
        });
        </script>
```

2. Создайте файл **form.php**, для сохранения заполненных форм:

```
<?
        $iRequisitePresetID =
intVal($_POST[ "REQ_TYPE" ]);
        $sName = htmlspecialchars($_POST[
"NAME" ]);
        $sLastName =
```

```
htmlspecialchars($_POST[ "LAST_NAME" ]);
        $sPhone =
htmlspecialchars($_POST[ "PHONE" ]);
        $arAddress = [];
        foreach($_POST[ "ADDRESS" ] as
$key => $val){
                $arAddress[ $key ] =
htmlspecialchars($val);
        }
        $arAddress[ 'TYPE_ID' ] = 1;//1
is actual address in
CRest::call('crm.enum.addresstype');
        $arAddress[ 'ENTITY_TYPE_ID' ] =
8;//8 - is requisite in
CRest::call('crm.enum.ownertype');

        $arPhone = (!empty($sPhone)) ?
array(array('VALUE' => $sPhone,
'VALUE_TYPE' => 'WORK')) : array();

        $result = CRest::call(
                'crm.contact.add',
                [
                        'fields' => [
                                'NAME' =>
$sName,

'LAST_NAME' => $sLastName,
                                'PHONE'
=> $arPhone,
                        ]
                ]
        );
        if(!empty($result[ 'result' ])){
                $resultRequisite =
CRest::call(
```

```
'crm.requisite.add',
                                [

'fields' => [

'ENTITY_TYPE_ID' => 3,//3 - is contact in
CRest::call('crm.enum.ownertype');

'ENTITY_ID' => $result[ 'result'
],//contact id

'PRESET_ID' => $iRequisitePresetID,

'TITLE' => implode(' ', [ $sName,
$sLastName ]),

'ACTIVE' => 'Y',

'NAME' => $sName
                                          ]
                                ]
                        );


if(!empty($resultRequisite[ 'result' ])){
                        $arAddress[
'ENTITY_ID' ] = $resultRequisite[
'result' ];//id requisite
                        $resultAddress =
CRest::call(

'crm.address.add',
                                [

'fields' => $arAddress
                                ]
                        );
```

```
                }
                echo json_encode([
'message' => 'Contact add' ]);
        }elseif(!empty($result[
'error_description' ])){
                echo json_encode([
'message' => 'Contact not added:
'.$result[ 'error_description' ] ]);
        }else{
                echo json_encode([
'message' => 'Contact not added' ]);
        }
?>
```

Простое добавление компании

Пример размещения на странице сайта формы, после заполнения которой в Битрикс24 создаётся новая компания.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

1. Создайте форму на нужной странице:

```
<form id="form_to_crm">
                <input type="text"
name="TITLE" placeholder="Title"
required>
                <input type="text"
name="EMAIL" placeholder="E-mail">
                <input type="text"
name="PHONE" placeholder="Phone">
                <input type="submit"
value="Submit">
        </form>
        <script
src="https://ajax.googleapis.com/ajax/lib
s/jquery/3.3.1/jquery.min.js"></script>
        <script>
        $(document).ready(function() {
            $('#form_to_crm').on(
'submit', function(el) {//event submit
form
```

```
                el.preventDefault();//the
default action of the event will not be
triggered
                var formData =
$(this).serialize();
                $.ajax({
                    'method': 'POST',
                    'dataType': 'json',
                    'url': 'form.php',
                    'data':  formData,
                    success:
function(data){//success callback

alert(data.message);
                    }
                });
            });
        });
        </script>
```

2. Создайте файл **form.php**, для сохранения заполненных форм:

```
<?
                                $sTitle =
htmlspecialchars($_POST["TITLE"]);
                                $sPhone =
htmlspecialchars($_POST["PHONE"]);
                                $sEmail =
htmlspecialchars($_POST["EMAIL"]);

                                $arPhone
= (!empty($sPhone)) ? array(array('VALUE'
=> $sPhone, 'VALUE_TYPE' => 'WORK')) :
array();
                                $arEmail
= (!empty($sEmail)) ? array(array('VALUE'
```

```
=> $sEmail, 'VALUE_TYPE' => 'HOME')) :
array();

                                $result =
CRest::call(

'crm.company.add',
                                        [

'fields' =>[

"TITLE" => $sTitle,//*Company
Name[string]

"COMPANY_TYPE" => 'CUSTOMER',//Company
type[crm_status {CUSTOMER:"Client",
SUPPLIER:"Supplier",
COMPETITOR:"Competitor",
PARTNER:"Partner", OTHER:"Other"}]//
CRest::call('crm.status.list',['filter'=>
['ENTITY_ID'=>'COMPANY_TYPE']]);

"PHONE" =>
$arPhone,//Phone[crm_multifield]

"EMAIL" => $arEmail,//E-
mail[crm_multifield]

//"LOGO" => '',//Logo[file]

//"ADDRESS" => '',//Street
address[string]

//"ADDRESS_2" => '',//Address (line 2)
[string]

//"ADDRESS_CITY" => '',//City[string]
```

```
//"ADDRESS_POSTAL_CODE" =>
'',//Zip[string]

//"ADDRESS_REGION" => '',//Region[string]

//"ADDRESS_PROVINCE" => '',//State /
Province[string]

//"ADDRESS_COUNTRY" =>
'',//Country[string]

//"ADDRESS_COUNTRY_CODE" => '',//Country
Code[string]

//"ADDRESS_LEGAL" => '',//Legal
address[string]

//"REG_ADDRESS" => '',//Billing
Address[string]

//"REG_ADDRESS_2" => '',//Billing Address
(line 2)[string]

//"REG_ADDRESS_CITY" => '',//Billing
City[string]

//"REG_ADDRESS_POSTAL_CODE" =>
'',//Billing Zip[string]

//"REG_ADDRESS_REGION" => '',//Billing
Region[string]

//"REG_ADDRESS_PROVINCE" => '',//Billing
State / Province[string]

//"REG_ADDRESS_COUNTRY" => '',//Billing
```

```
Country[string]

//"REG_ADDRESS_COUNTRY_CODE" =>
'',//Billing Country Code[string]

//"BANKING_DETAILS" => '',//Payment
details[string]

//"INDUSTRY" => '',//Industry[crm_status
{IT:"Information Technology",
TELECOM:"Telecommunication",
MANUFACTURING:"Manufacturing",
BANKING:"Banking Services",
CONSULTING:"Consulting",
FINANCE:"Finance",
GOVERNMENT:"Government",
DELIVERY:"Delivery",
ENTERTAINMENT:"Entertainment",
NOTPROFIT:"Non-profit", OTHER:"Other"}]//
CRest::call('crm.status.list',['filter'=>
['ENTITY_ID'=>'INDUSTRY']]);

//"EMPLOYEES" =>
'',//Employees[crm_status
{EMPLOYEES_1:"less than 50",
EMPLOYEES_2:"50 to 250", EMPLOYEES_3:"250
to 500", EMPLOYEES_4:"over 500"}]//
CRest::call('crm.status.list',['filter'=>
['ENTITY_ID'=>'EMPLOYEES']]);

//"CURRENCY_ID" =>
'',//Currency[crm_currency]//
CRest::call('crm.currency.list');

//"REVENUE" => '',//Annual
revenue[double]
```

```
//"OPENED" => '',//Available to
everyone[char]

//"COMMENTS" => '',//Comment[string]

//"HAS_PHONE" => '',//Has phone[char]

//"HAS_EMAIL" => '',//Has email[char]

//"HAS_IMOL" => '',//Has Open
Channel[char]

//"IS_MY_COMPANY" => '',//My
Company[char]

//"ASSIGNED_BY_ID" => '',//Responsible
person[user]

//"CREATED_BY_ID" => '',//Created
by[user]

//"MODIFY_BY_ID" => '',//Modified
by[user]

//"DATE_CREATE" => '',//Created
on[datetime]

//"DATE_MODIFY" => '',//Modified
on[datetime]

//"CONTACT_ID" =>
'',//Contact[crm_contact]//
CRest::call('crm.contact.list');

//"LEAD_ID" => '',//Lead[crm_lead]

//"ORIGINATOR_ID" => '',//External
```

```
source[string]

//"ORIGIN_ID" => '',//Item ID in data
source[string]

//"ORIGIN_VERSION" => '',//Original
version[string]

//"UTM_SOURCE" => '',//Ad system[string]

//"UTM_MEDIUM" => '',//Medium[string]

//"UTM_CAMPAIGN" => '',//Ad campaign
UTM[string]

//"UTM_CONTENT" => '',//Campaign
contents[string]

//"UTM_TERM" => '',//Campaign search
term[string]

//"WEB" => '',//Website[crm_multifield]

//"IM" => '',//Messenger[crm_multifield]

]

]);


if(!empty($result['result'])){

echo json_encode(['message' => 'Company
add']);

}elseif(!empty($result['error_description
'])){
```

```
echo json_encode(['message' => 'Company
not added:
'.$result['error_description']]);
                                }else{

echo json_encode(['message' => 'Company
not added']);
                                }



                ?>
```

Добавление компании с реквизитами

Пример размещения на странице сайта формы, после заполнения которой в Битрикс24 создаётся новая компания с прикреплением реквизитов.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

1. Создайте форму на нужной странице:

```
<?      $arAddressFields =
CRest::call('crm.address.fields',[]);
        /*
         crm.address.fields return:
https://dev.1c-
bitrix.ru/rest_help/crm/requisite/requisi
te_fields.php#address
        */
        $arRequisiteType =
CRest::call('crm.requisite.preset.list',
                [
                        'select'=>[
                         "ID", "NAME"
                        ]
                ]);

if(!empty($arRequisiteType['result'])):
                $arRequisiteType =
```

```
array_column($arRequisiteType['result'],'
NAME','ID');
                //unset system address
fields

unset($arAddressFields['result']
['TYPE_ID']);

unset($arAddressFields['result']
['ENTITY_TYPE_ID']);

unset($arAddressFields['result']
['ENTITY_ID']);
                //unset uninteresting
address fields

unset($arAddressFields['result']
['COUNTRY_CODE']);

unset($arAddressFields['result']
['ANCHOR_TYPE_ID']);

unset($arAddressFields['result']
['ANCHOR_ID']);
                ?>
                <form id="form_to_crm">
                        <select
name="REQ_TYPE" required>
                                <option
value="" disabled
selected>Select</option>
                                <?
foreach($arRequisiteType as $id=>$name):?
>

<option value="<?=$id?>"><?=$name?>
</option>
```

```
                                <?
endforeach;?>
                        </select>
                        <input
type="text" name="TITLE" placeholder="Org
name" required>
                        <input
type="text" name="INN" placeholder="INN">
                        <input
type="text" name="PHONE"
placeholder="Phone">
                        <?
if(is_array($arAddressFields['result'])):
?>
                                <?
foreach($arAddressFields['result'] as
$key=>$arField):?>

<input type="text" name="ADDRESS[<?=$key?
>]" placeholder="<?=$arField['title']?>"
<?=
($arField['isRequired'])?'required':'';?
>>
                                <?
endforeach;?>
                        <?endif;?>
                        <input
type="submit" value="Submit">
                </form>
        <?else:?>
                <?='No requisite
types.';?>
        <?endif;?>
        <script
src="https://ajax.googleapis.com/ajax/lib
s/jquery/3.3.1/jquery.min.js"></script>
        <script>
```

```
        $(document).ready(function() {
            $('#form_to_crm').on(
'submit', function(el) {//event submit
form
                el.preventDefault();//the
default action of the event will not be
triggered
                var formData =
$(this).serialize();
                $.ajax({
                    'method': 'POST',
                    'dataType': 'json',
                    'url': 'form.php',
                    'data':  formData,
                    success:
function(data){//success callback

alert(data.message);
                    }
                });

            });
        });
        </script>
```

2. Создайте файл **form.php**, для сохранения заполненных форм:

```
<?
        $iRequisitePresetID =
intVal($_POST["REQ_TYPE"]);
        $sTitle =
htmlspecialchars($_POST["TITLE"]);
        $sINN =
htmlspecialchars($_POST["INN"]);
        $sPhone =
htmlspecialchars($_POST["PHONE"]);
```

```
        $arAddress = [];

        foreach($_POST["ADDRESS"] as
$key=>$val){
                $arAddress[$key] =
htmlspecialchars($val);
        }
        $arAddress['TYPE_ID'] = 1;//1 is
actual address in
CRest::call('crm.enum.addresstype');
        $arAddress['ENTITY_TYPE_ID'] =
8;//8 - is requisite in
CRest::call('crm.enum.ownertype');

        $arPhone = (!empty($sPhone)) ?
array(array('VALUE' => $sPhone,
'VALUE_TYPE' => 'WORK')) : array();

        $result = CRest::call(
                'crm.company.add',
                [
                        'fields' =>[
                                'TITLE'
=> $sTitle,

'COMPANY_TYPE' => 'CUSTOMER',//is Client
in CRest::call('crm.status.list',
['filter'=>
['ENTITY_ID'=>'COMPANY_TYPE']])
                                'PHONE'
=> $arPhone,
                        ]
                ]);
        if(!empty($result['result'])){
                $resultRequisite =
CRest::call(
```

```
'crm.requisite.add',
                        [
                                'fields'
=>[

'ENTITY_TYPE_ID' => 4,//4 - is company in
CRest::call('crm.enum.ownertype');

'ENTITY_ID' =>
$result['result'],//company id

'PRESET_ID' => $iRequisitePresetID,

'TITLE' => $sTitle,

'ACTIVE' => 'Y',

'NAME' => $sTitle,

'RQ_INN' => $sINN,

                                ]
                        ]);


if(!empty($resultRequisite['result'])){

$arAddress['ENTITY_ID'] =
$resultRequisite['result'];//id requisite
                        $resultAddress =
CRest::call(

'crm.address.add',
                                [

'fields' =>$arAddress
                                ]);
```

```
                }
                echo
json_encode(['message' => 'Company
add']);

}elseif(!empty($result['error_description
']])){
                echo
json_encode(['message' => 'Company not
added: '.$result['error_description']]);
        }else{
                echo
json_encode(['message' => 'Company not
added']);
        }
?>
```

Добавление сделки с выбором реквизитов существующей компании или контакта

Пример размещения на странице сайта формы, после заполнения которой в Битрикс24 создаётся новая сделка с прикреплением компании с реквизитами.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

1. Создайте форму на нужной странице:

```
<?
$arAddressFields =
CRest::call('crm.address.fields',[]);
        /*
         crm.address.fields return:
https://dev.1c-
bitrix.ru/rest_help/crm/requisite/requisi
te_fields.php#address
        */
        $arRequisiteType =
CRest::call('crm.requisite.preset.list',
                [
                        'select'=>[
```

```
                                    "ID",
"NAME"
                            ]
                ));

if(!empty($arRequisiteType['result'])):
                $arRequisiteType =
array_column($arRequisiteType['result'],'
NAME','ID');
                //unset system address
fields

unset($arAddressFields['result']
['TYPE_ID']);

unset($arAddressFields['result']
['ENTITY_TYPE_ID']);

unset($arAddressFields['result']
['ENTITY_ID']);
                //unset uninteresting
address fields

unset($arAddressFields['result']
['COUNTRY_CODE']);

unset($arAddressFields['result']
['ANCHOR_TYPE_ID']);

unset($arAddressFields['result']
['ANCHOR_ID']);
                ?>
                <form id="form_to_crm">
                        <select
name="REQ_TYPE" required>
                                    <option
value="" disabled
```

```
selected>Select</option>
                                <?
foreach($arRequisiteType as $id=>$name):?
>

<option value="<?=$id?>"><?=$name?>
</option>
                                <?
endforeach;?>
                        </select>
                        <input
type="text" name="TITLE" placeholder="Org
name" required>
                        <input
type="text" name="INN" placeholder="INN">
                        <input
type="text" name="PHONE"
placeholder="Phone">
                        <?
if(is_array($arAddressFields['result'])):
?>
                                <?
foreach($arAddressFields['result'] as
$key=>$arField):?>

<input type="text" name="ADDRESS[<?=$key?
>]" placeholder="<?=$arField['title']?>"
<?=
($arField['isRequired'])?'required':'';?
>>
                                <?
endforeach;?>
                        <?endif;?>
                        <input
type="submit" value="Submit">
                </form>
        <?else:?>
```

```
                No requisite types.
        <?endif;?>
        <script
src="https://ajax.googleapis.com/ajax/lib
s/jquery/3.3.1/jquery.min.js"></script>
        <script>
        $(document).ready(function() {
            $('#form_to_crm').on(
'submit', function(el) {//event submit
form
                el.preventDefault();//the
default action of the event will not be
triggered
                var formData =
$(this).serialize();
                $.ajax({
                    'method': 'POST',
                    'dataType': 'json',
                    'url': 'form.php',
                    'data':  formData,
                    success:
function(data){//success callback

alert(data.message);
                    }
                });

            });
        });
        </script>
```

2. Создайте файл **form.php** для сохранения заполненных форм:

```
<?
        $iRequisitePresetID =
intVal($_POST["REQ_TYPE"]);
```

```
        $sTitle =
htmlspecialchars($_POST["TITLE"]);
        $sINN =
htmlspecialchars($_POST["INN"]);
        $sPhone =
htmlspecialchars($_POST["PHONE"]);
        $arAddress = [];

        foreach($_POST["ADDRESS"] as
$key=>$val){
                $arAddress[$key] =
htmlspecialchars($val);
        }
        $arAddress['TYPE_ID'] = 1;//1 is
actual address in
CRest::call('crm.enum.addresstype');
        $arAddress['ENTITY_TYPE_ID'] =
8;//8 - is requisite in
CRest::call('crm.enum.ownertype');

        $arPhone = (!empty($sPhone)) ?
array(array('VALUE' => $sPhone,
'VALUE_TYPE' => 'WORK')) : array();

        $result = CRest::call(
                'crm.company.add',
                [
                'fields' =>[
                        'TITLE' =>
$sTitle,
                        'COMPANY_TYPE' =>
'CUSTOMER',//is Client in
CRest::call('crm.status.list',['filter'=>
['ENTITY_ID'=>'COMPANY_TYPE']])
                        'PHONE' =>
$arPhone,
                ]
```

```
        ]);
        if(!empty($result['result'])){
                $resultRequisite =
CRest::call(

'crm.requisite.add',
                        [
                                'fields'
=>[

'ENTITY_TYPE_ID' => 4,//4 - is company in
CRest::call('crm.enum.ownertype');

'ENTITY_ID' =>
$result['result'],//company id

'PRESET_ID' => $iRequisitePresetID,

'TITLE' => $sTitle,

'ACTIVE' => 'Y',

'NAME' => $sTitle,

'RQ_INN' => $sINN,

                                ]
                        ]);
                $arDealFields = [
                        'TITLE' =>
$sTitle,
                        'COMPANY_ID' =>
$result['result']
                ];

if(!empty($resultRequisite['result'])){
```

```
$arDealFields['REQUISITE_ID'] =
$resultRequisite['result'];//add
requisite to deal is analogue
"crm.requisite.link.list"

$arAddress['ENTITY_ID'] =
$resultRequisite['result'];//id requisite
                        $resultAddress =
CRest::call(

'crm.address.add',
                                [

'fields' =>$arAddress
                                ]);
                }
                $resultDeal =
CRest::call(
                        'crm.deal.add',
                        [
                                'fields'
=> $arDealFields
                        ]);

                echo
json_encode(['message' => 'add']);

}elseif(!empty($result['error_description
'])){
                echo
json_encode(['message' => 'not added:
'.$result['error_description']]);
        }else{
                echo
json_encode(['message' => 'not added']);
        }?>
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Как добавить клиенту событие календаря

Пример создаёт активити в календарь контакта с необходимостью выполнения в течение часа.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

```
$contactID = 1;
$resultContact = CRest::call(
        'crm.contact.get',
        [
                'id' => $contactID
        ]
);
$resultActivity = [];
if (!empty($resultContact['result']
['ASSIGNED_BY_ID']) &&
!empty($resultContact['result']['PHONE']))
{
        $contactPhone =
reset($resultContact['result']['PHONE']);
        $staffID = $resultContact['result']
['ASSIGNED_BY_ID'];
        $resultActivity = CRest::call(
                'crm.activity.add',
                [
                        'fields' => [
```

```
                                    "SUBJECT" =>
"calendar title",

"DESCRIPTION" => "calendar body",

"DESCRIPTION_TYPE" => 3,//text,html,bbCode
type id in:
CRest::call('crm.enum.contenttype');
                                    "OWNER_ID"
=> $contactID,

"OWNER_TYPE_ID" => 3, //
CRest::call('crm.enum.ownertype');
                                    "TYPE_ID" =>
1, // CRest::call('crm.enum.activitytype');

"COMMUNICATIONS" => [
                                          [

'VALUE' => $contactPhone['VALUE'],

'ENTITY_ID' => $contactID,

'ENTITY_TYPE_ID' => 3//
CRest::call('crm.enum.ownertype');
                                          ]
                                    ],
                                    "START_TIME"
=> date("Y-m-d H:i:s", time()),
                                    "END_TIME"
=> date("Y-m-d H:i:s", time() + 3600),

"RESPONSIBLE_ID" => $staffID,
                              ]
                  ]
        );
}
```

```
if (!empty($resultActivity['result']))
{
        echo json_encode(['message' =>
'Activity add']);
}
elseif
(!empty($resultActivity['error_description']
))
{
        echo json_encode(['message' =>
'Activity not added: ' .
$resultActivity['error_description']]);
}
else
{
        echo json_encode(['message' =>
'Activity not added']);
}
```

Добавление сделки (лида, счета, компреда) с товарами, с применением скидок и налогов

Описание

Примеры создания различных сущностей с одновременным добавленем товаров к ним. Добавляемый товар берётся из Битрикс24 с ценой больше нуля. Все примеры добавляют товар в максимально возможном количестве вариаций, около каждой вариации есть мини комментарий с описанием какими дополнительными условиями будет отображаться товар.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

Прикрепление товаров к сделке

```
 <?
$resultProduct = CRest::call(
        'crm.product.list',
        [
                'filter' => [
                        '>PRICE' => 0,
                ]
        ]
);
```

```
if (empty($resultProduct['result']))
{
        echo 'product error, create product in b24';
        exit;
}
else
{
        $arProduct = $resultProduct['result'][0];
}

//Deal product
$resultDeal = CRest::call(
        'crm.deal.add',
        [
                'fields' => [
                        'TITLE' => 'Example',
                ]
        ]
);
if ($ID = $resultDeal['result'])
{
        $result = CRest::call(
                'crm.deal.productrows.set',
                [
                        'id' => $ID,
                        'rows' => [
                                [//product with auto calc
tax
                                 'PRODUCT_ID' =>
$arProduct['ID'],
                                 'PRICE_EXCLUSIVE' =>
$arProduct['PRICE'],
                                 'TAX_RATE' => 15,
                                 'TAX_INCLUDED' => 'N',
                                 'QUANTITY' => 1
                                ],
                                [//product with tax include
                                 'PRODUCT_ID' =>
$arProduct['ID'],
                                 'PRICE' =>
$arProduct['PRICE'],
                                 'TAX_RATE' => 15,
                                 'TAX_INCLUDED' => 'Y',
                                 'QUANTITY' => 1
                                ],
                                [//product with discount
                                 'PRODUCT_ID' =>
$arProduct['ID'],
                                 'PRICE' =>
$arProduct['PRICE'],
                                 'DISCOUNT_SUM' => 100,
                                 'DISCOUNT_TYPE_ID' =>
1,//is sum discount type
                                 'QUANTITY' => 1
```

```
                                ],
                                [//product with a real
discount
                                 'PRODUCT_ID' =>
$arProduct['ID'],
                                 'PRICE' =>
$arProduct['PRICE'] - 100,
                                 'DISCOUNT_SUM' => 100,
                                 'DISCOUNT_TYPE_ID' =>
1,//is sum discount type
                                 'QUANTITY' => 1
                                ],
                                [//product with discount
percent
                                 'PRODUCT_ID' =>
$arProduct['ID'],
                                 'PRICE_EXCLUSIVE' =>
$arProduct['PRICE'],
                                 'DISCOUNT_RATE' => 10,
                                 'DISCOUNT_TYPE_ID' =>
2,//is percent discount type
                                 'QUANTITY' => 1
                                ],
                                [//product with real
discount percent
                                 'PRODUCT_ID' =>
$arProduct['ID'],
                                 'PRICE_EXCLUSIVE' =>
$arProduct['PRICE'] - ($arProduct['PRICE'] * 0.1),
                                 'DISCOUNT_RATE' => 10,
                                 'DISCOUNT_TYPE_ID' =>
2,//is percent discount type
                                 'QUANTITY' => 1
                                ],
                        ]
                ]
        );
}
else
{
        echo 'error create deal';
        exit;
}

?>
```

Прикрепление товаров к лиду

```
 <?
$resultProduct = CRest::call(
        'crm.product.list',
        [
                'filter' => [
                        '>PRICE' => 0,
                ]
        ]
);

if (empty($resultProduct['result']))
{
        echo 'product error, create product in b24';
        exit;
}
else
{
        $arProduct = $resultProduct['result'][0];
}

//Lead product
$resultLead = CRest::call(
        'crm.lead.add',
        [
                'fields' => [
                        'TITLE' => 'Example',
                ]
        ]
);
if ($ID = $resultLead['result'])
{
        $result = CRest::call(
                'crm.lead.productrows.set',
                [
                        'id' => $ID,
                        'rows' => [
                                [//product with auto calc
tax
                                 'PRODUCT_ID' =>
$arProduct['ID'],
                                 'PRICE_EXCLUSIVE' =>
$arProduct['PRICE'],
                                 'TAX_RATE' => 15,
                                 'TAX_INCLUDED' => 'N',
                                 'QUANTITY' => 1
                                ],
                                [//product with tax include
                                 'PRODUCT_ID' =>
$arProduct['ID'],
                                 'PRICE' =>
$arProduct['PRICE'],
                                 'TAX_RATE' => 15,
                                 'TAX_INCLUDED' => 'Y',
```

```
                                 'QUANTITY' => 1
                                ],
                                [//product with discount
                                 'PRODUCT_ID' =>
$arProduct['ID'],
                                 'PRICE' =>
$arProduct['PRICE'],
                                 'DISCOUNT_SUM' => 100,
                                 'DISCOUNT_TYPE_ID' =>
1,//is sum discount type
                                 'QUANTITY' => 1
                                ],
                                [//product with a real
discount
                                 'PRODUCT_ID' =>
$arProduct['ID'],
                                 'PRICE' =>
$arProduct['PRICE'] - 100,
                                 'DISCOUNT_SUM' => 100,
                                 'DISCOUNT_TYPE_ID' =>
1,//is sum discount type
                                 'QUANTITY' => 1
                                ],
                                [//product with discount
percent
                                 'PRODUCT_ID' =>
$arProduct['ID'],
                                 'PRICE_EXCLUSIVE' =>
$arProduct['PRICE'],
                                 'DISCOUNT_RATE' => 10,
                                 'DISCOUNT_TYPE_ID' =>
2,//is percent discount type
                                 'QUANTITY' => 1
                                ],
                                [//product with real
discount percent
                                 'PRODUCT_ID' =>
$arProduct['ID'],
                                 'PRICE_EXCLUSIVE' =>
$arProduct['PRICE'] - ($arProduct['PRICE'] * 0.1),
                                 'DISCOUNT_RATE' => 10,
                                 'DISCOUNT_TYPE_ID' =>
2,//is percent discount type
                                 'QUANTITY' => 1
                                ],
                        ]
                ]
        );
}
else
{
        echo 'error create lead';
        exit;
}
```

```
?>
```

Создание счёта с товарами

```
 <?
$resultProduct = CRest::call(
        'crm.product.list',
        [
                'filter' => [
                        '>PRICE' => 0,
                ]
        ]
);

if (empty($resultProduct['result']))
{
        echo 'product error, create product in b24';
        exit;
}
else
{
        $arProduct = $resultProduct['result'][0];
}
$resultCompany = CRest::call(
        'crm.company.add',
        [
                'fields' => [
                        'TITLE' => 'Example',
                ]
        ]
);

if ($iCompanyID = $resultCompany['result'])
{
        $resultInvoice = CRest::call(
                'crm.invoice.add',
                [
                        'fields' => [
                                'ORDER_TOPIC' => 'Invoice
by company with product',
                                'UF_COMPANY_ID' =>
$iCompanyID,
                                'PERSON_TYPE_ID' => 1,//1
is company in CRest::call('crm.persontype.list')
                                'PAY_SYSTEM_ID' =>
20,//some in CRest::call('sale.paysystem.list')
                                'STATUS_ID' => 'N',
                                'DATE_INSERT' =>
```

```
date(DATE_ATOM),
                                'DATE_BILL' =>
date(DATE_ATOM),
                                'DATE_PAY_BEFORE' =>
date(DATE_ATOM, time() + 3600 * 24 * 20),//20 day pay

                                'PRODUCT_ROWS' => [
                                        [//product with tax
                                         'PRODUCT_ID' =>
$arProduct['ID'],
                                         'PRODUCT_NAME' =>
$arProduct['NAME'],
                                         'PRICE' =>
$arProduct['PRICE'] + ($arProduct['PRICE'] * 0.15),
                                         'VAT_RATE' =>
0.15,
                                         'QUANTITY' => 1
                                        ],
                                        [//product with
discount sum, percent not supported
                                         'PRODUCT_ID' =>
$arProduct['ID'],
                                         'PRODUCT_NAME' =>
$arProduct['NAME'],
                                         'PRICE' =>
$arProduct['PRICE'],
                                         'DISCOUNT_PRICE'
=> 100,
                                         'QUANTITY' => 1
                                        ],
                                        [//product with
real discount sum, percent not supported
                                         'PRODUCT_ID' =>
$arProduct['ID'],
                                         'PRODUCT_NAME' =>
$arProduct['NAME'],
                                         'PRICE' =>
$arProduct['PRICE'] - 100,
                                         'DISCOUNT_PRICE'
=> 100,
                                         'QUANTITY' => 1
                                        ],
                                ],
                        ]
                ]
        );
}
?>
```

Прикрепление товаров к компред

```
 <?
$resultProduct = CRest::call(
        'crm.product.list',
        [
                'filter' => [
                        '>PRICE' => 0,
                ]
        ]
);

if (empty($resultProduct['result']))
{
        echo 'product error, create product in b24';
        exit;
}
else
{
        $arProduct = $resultProduct['result'][0];
}
$resultCompany = CRest::call(
        'crm.company.add',
        [
                'fields' => [
                        'TITLE' => 'Example',
                ]
        ]
);

if ($iCompanyID = $resultCompany['result'])
{
        $resultQuote = CRest::call(
                'crm.quote.add',
                [
                        'fields' => [
                                "TITLE" => "Quote by
company with product",
                                "OPENED" => "Y",
                                "ASSIGNED_BY_ID" => 1,
                                "CURRENCY_ID" => "USD",
                                "BEGINDATE" =>
date(DATE_ATOM),
                                "CLOSEDATE" =>
date(DATE_ATOM, time() + 3600 * 24 * 20),//20 day
                                'COMPANY_ID' =>
$iCompanyID,
                                'STATUS_ID' => 'N',
                        ]
                ]
        );
        if ($ID = $resultQuote['result'])
        {
                $result = CRest::call(
                        'crm.quote.productrows.set',
```

```
                            [
                                    'id' => $ID,
                                    'rows' => [
                                            [//product with
auto calc tax
                                             'PRODUCT_ID' =>
$arProduct['ID'],
                                             'PRICE_EXCLUSIVE'
=> $arProduct['PRICE'],
                                             'TAX_RATE' => 15,
                                             'TAX_INCLUDED' =>
'N',
                                             'QUANTITY' => 1
                                            ],
                                            [//product with tax
include
                                             'PRODUCT_ID' =>
$arProduct['ID'],
                                             'PRICE' =>
$arProduct['PRICE'],
                                             'TAX_RATE' => 15,
                                             'TAX_INCLUDED' =>
'Y',
                                             'QUANTITY' => 1
                                            ],
                                            [//product with
discount
                                             'PRODUCT_ID' =>
$arProduct['ID'],
                                             'PRICE' =>
$arProduct['PRICE'],
                                             'DISCOUNT_SUM' =>
100,
                                             'DISCOUNT_TYPE_ID'
=> 1,//is sum discount type
                                             'QUANTITY' => 1
                                            ],
                                            [//product with a
real discount
                                             'PRODUCT_ID' =>
$arProduct['ID'],
                                             'PRICE' =>
$arProduct['PRICE'] - 100,
                                             'DISCOUNT_SUM' =>
100,
                                             'DISCOUNT_TYPE_ID'
=> 1,//is sum discount type
                                             'QUANTITY' => 1
                                            ],
                                            [//product with
discount percent
                                             'PRODUCT_ID' =>
$arProduct['ID'],
                                             'PRICE_EXCLUSIVE'
```

```
=> $arProduct['PRICE'],
                                        'DISCOUNT_RATE' =>
10,
                                        'DISCOUNT_TYPE_ID'
=> 2,//is percent discount type
                                        'QUANTITY' => 1
                                       ],
                                       [//product with
real discount percent
                                        'PRODUCT_ID' =>
$arProduct['ID'],
                                        'PRICE_EXCLUSIVE'
=> $arProduct['PRICE'] - ($arProduct['PRICE'] * 0.1),
                                        'DISCOUNT_RATE' =>
10,
                                        'DISCOUNT_TYPE_ID'
=> 2,//is percent discount type
                                        'QUANTITY' => 1
                                       ],
                                ]
                        ]
                );
        }
        else
        {
                echo 'Error create quote';
        }
}
else
{
        echo 'Error create company';
}
?>
```

Как добавить товар со значениями пользовательских полей

Пример заполнения различных свойств при добавлении товара.

Для работы примера создайте папку **/pictures** рядом с исполняемым файлом примера и заполнитее её картинками с названиями "1.jpg" - "6.jpg". Также в начале примера необходимо исправить значения из примера на ваши:

- **$propertyIDSelect** - ID не множественного списочного свойства.
- **$propertySelectValueID** - ID значения не множественного списочного свойства.
- **$propertyIDMultiSelect** - ID множественного списочного свойства.
- **$propertyMultiSelectValueID** - ID значений множественного списочного свойства.
- **$propertyIDFile** - ID не множественного свойства типа файл.
- **$propertyIDMultiFile** - ID множественного свойства типа файл.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

```
<?
        $propertyIDSelect = 106;
        $propertySelectValueID = 85;
```

```
        $propertyIDMultiSelect = 105;
        $propertyMultiSelectValueID =
[79,80,82];

        $propertyIDFile = 107;
        $propertyFilePathToPicture =
'pictures/1.jpg';//relative or full path on
server

        $propertyIDMultiFile = 108;
        $propertyMultiFilePathToPicture =
[//relative or full path on server

'pictures/2.jpg',

'pictures/3.jpg',

'pictures/4.jpg',
        ];

        $standardPreviewPicturePath =
'pictures/5.jpg';//relative or full path on
server
        $standardDetailPicturePath =
'pictures/6.jpg';//relative or full path on
server

        $arFields = [
                'NAME'        => 'Example
product 2',
                'CURRENCY_ID' => 'USD',
                'PRICE'       => 4900,
                'SORT'        => 500
        ];


        if($propertyIDSelect > 0 &&
```

```
$propertySelectValueID > 0)
        {
                $arFields[ 'PROPERTY_' .
$propertyIDSelect ] =
$propertySelectValueID;
        }

        if($propertyIDMultiSelect > 0 &&
is_array($propertyMultiSelectValueID) &&
count($propertyMultiSelectValueID) > 0)
        {
                $arFields[ 'PROPERTY_' .
$propertyIDMultiSelect ] =
$propertyMultiSelectValueID;
        }

        if($propertyIDFile > 0 &&
!empty($propertyFilePathToPicture) &&
file_exists($propertyFilePathToPicture))
        {
                $fileName = end(explode('/',
$propertyFilePathToPicture));
                $arFields[ 'PROPERTY_' .
$propertyIDFile ] = [
                        "fileData" => [
                                $fileName,

base64_encode(file_get_contents($propertyFil
ePathToPicture))
                        ]
                ];
        }

        if($propertyIDMultiFile > 0 &&
is_array($propertyMultiFilePathToPicture) &&
count($propertyMultiFilePathToPicture) > 0)
        {
```

```
foreach($propertyMultiFilePathToPicture as
$path){

if(file_exists($path))
                        {
                                $fileName =
end(explode('/', $path));
                                $arFields[
'PROPERTY_' . $propertyIDMultiFile ][] = [

"fileData" => [

$fileName,

base64_encode(file_get_contents($path))
                                        ]
                                ];
                        }
                }
        }


if(!empty($standardPreviewPicturePath) &&
file_exists($standardPreviewPicturePath))
        {
                $fileName = end(explode('/',
$standardPreviewPicturePath));
                $arFields[ 'PREVIEW_PICTURE'
] = [
                        "fileData" => [
                                $fileName,

base64_encode(file_get_contents($standardPre
viewPicturePath))
                        ]
                ];
```

```
        }


if(!empty($standardDetailPicturePath) &&
file_exists($standardDetailPicturePath))
        {
                $fileName = end(explode('/',
$standardDetailPicturePath));
                $arFields[ 'DETAIL_PICTURE'
] = [
                        "fileData" => [
                                $fileName,

base64_encode(file_get_contents($standardDet
ailPicturePath))
                        ]
                ];
        }

        $result = CRest ::call(
                'crm.product.add',
                [
                        'fields' =>
$arFields

                ]
        );

?>
```

Добавление картинок в поля лида

Пример размещения на странице сайта формы с возможностью прикрепления файлов. По заполнении формы в Битрикс24 будет создаваться новый лид.
В коде ниже есть пользовательские поля:
UF_CRM_1551350435588 - множественное свойство типа файл;
UF_CRM_1551362436225 - свойство типа файл.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

1. Создаём форму на нужной странице:

```
 <script
src="https://ajax.googleapis.com/ajax/lib
s/jquery/3.3.1/jquery.min.js"></script>
        <script>
        $(document).ready(function() {
            $('#form_to_crm_file').on(
'submit', function(el) {//event submit
form
                el.preventDefault();//the
default action of the event will not be
triggered
                var formData = new
FormData(this);
                var xhr = new
XMLHttpRequest();
```

```
                xhr.open("POST",
'form.php');
                xhr.onreadystatechange =
function () {
                    if (this.readyState
=== 4) {
                        if (this.status
>= 200 && this.status < 400) {
                            // Success!
                            var resp =
this.responseText;
                            try {
                                var json
= JSON.parse(resp);
                                if
(typeof json.message !== 'undefined') {

alert(json.message);
                                }

                            } catch (e) {
                                return
false;
                            }

                        } else {

alert('error');
                        }
                    }
                };
                xhr.send(formData);
            });
        });
        </script>
        <form id="form_to_crm_file"
method="post" enctype="multipart/form-
```

```
data">
                <input type="text"
name="NAME" placeholder="Name" required>
                <input type="text"
name="LAST_NAME" placeholder="Last name">
                <input type="text"
name="PHONE" placeholder="Phone">
                <input type="text"
name="EMAIL" placeholder="E-mail">
                <input type="file"
name="FILE" placeholder="File">
                <input type="file"
name="FILES[]" placeholder="Files"
multiple="multiple">

                <input type="submit"
value="Submit">
        </form>
```

2. Создаём файл **form.php**, для сохранения заполненных форм:

```
<?
        $sName =
htmlspecialchars($_POST["NAME"]);
        $sLastName =
htmlspecialchars($_POST["LAST_NAME"]);
        $sPhone =
htmlspecialchars($_POST["PHONE"]);
        $sEmail =
htmlspecialchars($_POST["EMAIL"]);
        //UF_CRM_1551350435588 - multiple
file field
        //UF_CRM_1551362436225 - file
field
        $arFiles = [];
        $arFile = [];
```

```
        //make array multiple files for
add to custom field
        if(!empty($_FILES['FILES']
['tmp_name']) &&
is_array($_FILES['FILES']['tmp_name'])){
                foreach($_FILES['FILES']
['tmp_name'] as $k=>$path){
                        $arFiles[] = [

"fileData" => [

$_FILES['FILES']['name'][$k],

base64_encode(file_get_contents($path))
                                ]
                        ];
                }
        }
        //make file array for add to
custom field
        if(!empty($_FILES['FILE']
['tmp_name'])){
                $arFile = [
                        "fileData" => [

$_FILES['FILE']['name'],

base64_encode(file_get_contents($_FILES['
FILE']['tmp_name']))
                        ]
                ];
        }
        //Форматируем Phone и почту для
сохранения
        $arPhone = (!empty($sPhone)) ?
array(array('VALUE' => $sPhone,
'VALUE_TYPE' => 'WORK')) : array();
```

```
        $arEmail = (!empty($sEmail)) ?
array(array('VALUE' => $sEmail,
'VALUE_TYPE' => 'HOME')) : array();

        $result = CRest::call(
                'crm.lead.add',
                [
                'fields' =>[
                        'TITLE' => 'From
the site: '.implode(' ',
[$sName,$sLastName]),//*Lead Name[string]
                        'NAME' =>
$sName,//Name[string]

'UF_CRM_1551350435588' => $arFiles,

'UF_CRM_1551362436225' => $arFile,
                        'LAST_NAME' =>
$sLastName,//Last name[string]
                        'PHONE' =>
$arPhone,//Phone[string]
                        'EMAIL' =>
$arEmail,//E-mail[string]
                ]
        ]);
        if(!empty($result['result'])){
                echo
json_encode(['message' => 'Lead add']);

}elseif(!empty($result['error_description
'])){
                echo
json_encode(['message' => 'Lead not
added: '.$result['error_description']]);
        }else{
                echo
json_encode(['message' => 'Lead not
```

```
added']);
        }
?>
```

Как сделать свою карточку редактирования лида

Пример автоматической генерирации карточки редактирования лида со всеми полями созданными в Битрикс24 на странице приложения.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

Некоторые типы полей не реализованы в данном примере, на месте полей с неподдерживаемым типом будет выводиться сообщение *field not support*.

Код генерируемой формы:

```
<?
        $ID = intVal($_REQUEST['ID']);
        class CPrintForm
        {
                /**
                 * @var $arParams array
params input keys: 'NAME', 'ID', 'TYPE',
'REQUIRED', 'CHECKED', 'DISABLE',
'MULTIPLE', 'VALUE'
                 * @return string html
select
                 */
                public static function
input($arParams)
                {
```

```
                        $count = 0;
                        $i = 0;
                        $sResult = '';
                        if($arParams[
'MULTIPLE' ] && $arParams[ 'TYPE' ] !=
'file')
                        {
                                $count = 2;
                        }
                        $value = $arParams[
'VALUE' ];
                        while($i <= $count)
                        {

                                if($count >
0)
                                {

$value = $arParams[ 'VALUE' ][ $i ];
                                }
                                $sResult .=
'<input class="form-control' . (($arParams[
'TYPE' ] == 'file') ? '-file' : '') . '"';

if(!empty($arParams[ 'ID' ]))
                                {

$sResult .= ' id="' . $arParams[ 'ID' ] .
'"';
                                }

if(!empty($arParams[ 'NAME' ]))
                                {

$sResult .= ' name="' . $arParams[ 'NAME' ]
. '' . (($arParams[ 'MULTIPLE' ]) ? '[]' :
'') . '"';
```

```
                                    }

if(!empty($arParams[ 'TYPE' ]))
                                    {

$sResult .= ' type="' . $arParams[ 'TYPE' ]
. '"';
                                    }

if(!empty($arParams[ 'REQUIRED' ]))
                                    {

$sResult .= ' required';
                                    }

if(!empty($arParams[ 'DISABLE' ]))
                                    {

$sResult .= ' disabled';
                                    }

if(!empty($arParams[ 'CHECKED' ]))
                                    {

$sResult .= ' checked';
                                    }

if(!empty($arParams[ 'MULTIPLE' ]))
//sometimes work, not for standard
type="text"
                                    {

$sResult .= ' multiple';
                                    }

if(!empty($arParams[ 'VALUE' ]))
                                    {
```

```
$sResult .= ' value="' . $value . '"';
                                }
                                $sResult .=
'>';
                                $i++;
                        }

                        return $sResult;
                }

                /**
                 * @var $arParams array
settings of select params keys: 'NAME',
'ID', 'REQUIRED', 'DISABLE','MULTIPLE',
'VALUE'
                 * @var $arList array of
select options where key is value option and
value is title
                 * @return string html
select
                 */
                public static function
select($arParams, $arList)
                {
                        $sResult = '';
                        if(!empty($arList)
&& is_array($arList))
                        {
                                $sResult .=
'<select class="form-control"'
                                                .
(($arParams[ 'NAME' ]) ? ' name="' .
$arParams[ 'NAME' ] . '' . (($arParams[
'MULTIPLE' ]) ? '[]' : '') . '"' : '')
                                                .
(($arParams[ 'ID' ]) ? ' id="' . $arParams[
```

```
'ID' ] . '"' : '')
                                                .
(($arParams[ 'REQUIRED' ]) ? ' required' :
'')
                                                .
(($arParams[ 'DISABLE' ]) ? ' disabled' :
'')
                                                .
(($arParams[ 'MULTIPLE' ]) ? ' multiple' :
'')
                                                .
'>';
                                $value = [];

if(is_array($arParams[ 'VALUE' ]))
                                {

$value = $arParams[ 'VALUE' ];
                                }
                                else
                                {

$value[] = ($arParams[ 'VALUE' ]) ?
$arParams[ 'VALUE' ] : '';
                                }

foreach($arList as $key => $title)
                                {

$sResult .= '<option value="' . $key . '" '
.

((in_array($key, $value)) ? ' selected' :
'')

. '>' . $title . '</option>';
                                }
```

```
                                        $sResult .=
'</select>';
                                }
                                return $sResult;
                        }
                }


                $arData = [
                        //Get all fields and
standard enum fields
                        'FIELDS' => [
                                'method' =>
'crm.lead.fields',
                                'params' => []
                        ],
                        'FIELD_VALUES_SOURCE_ID' =>
[
                                'method' =>
'crm.status.list',//only 50 first values
                                'params' =>
['filter' => ['ENTITY_ID' =>
'$result[FIELDS][SOURCE_ID][statusType]']]
                        ],
                        'FIELD_VALUES_STATUS_ID' =>
[
                                'method' =>
'crm.status.list',//only 50 first values
                                'params' =>
['filter' => ['ENTITY_ID' =>
'$result[FIELDS][STATUS_ID][statusType]']]
                        ],
                        'FIELD_VALUES_CURRENCY' => [
                                'method' =>
'crm.currency.list',//only 50 first values
                                'params' =>
['filter' => ['ENTITY_ID' =>
```

```
'$result[FIELDS][STATUS_ID][statusType]']]
                ],
                'OWNER_TYPE' => [
                        'method' =>
'crm.enum.ownertype',
                        'params' => []
                ],
        ];
        if($ID > 0)//get item and standard
enum field values if is update form
        {
                $arData['ITEM'] = [
                        'method' =>
'crm.lead.get',
                        'params' => ['id' =>
$ID]
                ];
                $arData['VALUE_COMPANY_ID']
= [
                        'method' =>
'crm.company.get',
                        'params' => ['id' =>
'$result[ITEM][COMPANY_ID]']
                ];
                $arData['VALUE_CONTACT_ID']
= [
                        'method' =>
'crm.contact.get',
                        'params' => ['id' =>
'$result[ITEM][CONTACT_ID]']
                ];
        }

        $arResult =
CRest::callBatch($arData, 0);

        $arResult = $arResult['result']
```

```
['result'];
        $sResult = '';
        $sResultCustom = '';

        if(is_array($arResult['FIELDS'])):
                foreach($arResult[ 'FIELDS'
] as $key => $arField)
                {
                        $value = '';
                        $return = '';
                        if(!empty($arResult[
'ITEM' ][ $key ]))
                        {
                                $value =
$arResult[ 'ITEM' ][ $key ];
                        }
                        $arList =
(isset($arResult[ 'FIELD_VALUES_' . $key ]))
? $arResult[ 'FIELD_VALUES_' . $key ] : [];
                        switch($arField[
'type' ])
                        {
                                case
'crm_status':

if(empty($arList))
                                        {

$arFieldsStatus = \CRest
::get('crm.status.list', ['filter' =>
['ENTITY_ID' => $arField[ 'statusType' ]]]);

if(!empty($arFieldsStatus[ 'result' ]))

$arList = $arFieldsStatus[ 'result' ];
                                        }
```

```
$arList = array_column($arList, 'NAME',
'STATUS_ID');


$return = CPrintForm ::select(

[

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => $arField[ 'isMultiple' ],

'VALUE' => $value

],

$arList);

break;
                                case
'crm_currency':

$arList = array_column($arResult[
'FIELD_VALUES_CURRENCY' ], 'FULL_NAME',
'CURRENCY');

$return = CPrintForm ::select(

[

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],
```

```
'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => $arField[ 'isMultiple' ],

'VALUE' => $value

],

$arList);

break;
                                case
'enumeration':

if(!empty($arField[ 'items' ]))

$arList = array_column($arField[ 'items' ],
'VALUE', 'ID');


$return = CPrintForm ::select(

[

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => $arField[ 'isMultiple' ],

'VALUE' => $value

],
```

```
$arList);

break;
                                    case
'crm_multifield'://its simple example: need
multifield, check data type and more...

if(!empty($value) && is_array($value))

$value = reset($value)[ 'VALUE' ];

$return = CPrintForm ::input(

[

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => false,

'VALUE' => $value,

'TYPE' => 'text',

]);

break;
                                    case
'crm_company':

$return = CPrintForm ::input(

[
```

```
'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => $arField[ 'isMultiple' ],

'VALUE' => $value,

'TYPE' => 'text',

]);

if(!empty($arResult[ 'VALUE_COMPANY_ID' ])
&& $value == $arResult[ 'VALUE_COMPANY_ID' ]
[ 'ID' ])
                                        {

$return .= '(' . $arResult[
'VALUE_COMPANY_ID' ][ 'TITLE' ] . ')';
                                        }

break;
                                  case
'crm_contact':

$return = CPrintForm ::input(

[

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],
```

```
'MULTIPLE' => $arField[ 'isMultiple' ],

'VALUE' => $value,

'TYPE' => 'text',

]);


if(!empty($arResult[ 'VALUE_CONTACT_ID' ])
&& $value == $arResult[ 'VALUE_CONTACT_ID' ]
[ 'ID' ])
                                        {

$return .= '(' . implode(' ', [$arResult[
'VALUE_CONTACT_ID' ][ 'NAME' ], $arResult[
'VALUE_CONTACT_ID' ][ 'LAST_NAME' ]]) . ')';
                                        }

break;
                                case 'file':

$return = CPrintForm ::input(

[

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => $arField[ 'isMultiple' ],

'VALUE' => $value,

'TYPE' => 'file',
```

```
]);

if($arField[ 'isMultiple' ])
                                        {

if(is_array($value))

{

foreach($value as $k => $val)

{

if(!empty($val[ 'downloadUrl' ]))

{

$return .= '<br/><a href="' . $val[
'downloadUrl' ] . '">old file ' . $k .
'</a>';

}

}

}
                                        }
                                        else
                                        {

if(!empty($value[ 'downloadUrl' ]))

{

$return .= '<br/><a href="' . $value[
'downloadUrl' ] . '">old file</a>';
```

```
}
                                                }
break;
                                        case 'date':
if(!empty($value))
                                                {
$value = date('Y-m-d', strtotime($value));
                                                }
$return = CPrintForm ::input(
[
'NAME' => 'form[' . $key . ']',
'REQUIRED' => $arField[ 'isRequired' ],
'DISABLE' => $arField[ 'isReadOnly' ],
'MULTIPLE' => $arField[ 'isMultiple' ],
'VALUE' => $value,
'TYPE' => 'date',
]);
break;
                                        case
'datetime':
if(!empty($value))
                                                {
```

```
$value = date('Y-m-d\TH:i:s',
strtotime($value));
                                        }

$return = CPrintForm ::input(

[

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => $arField[ 'isMultiple' ],

'VALUE' => $value,

'TYPE' => 'datetime-local',

]);

break;
                                  case 'char':

$return = CPrintForm ::input(

[

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => $arField[ 'isMultiple' ],
```

```
'VALUE' => 'Y',

'CHECKED' => ($value == 'Y') ? true : false,

'TYPE' => 'checkbox',

]);

break;

                                case
'boolean':

$return = CPrintForm ::input(

[

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => $arField[ 'isMultiple' ],

'VALUE' => '1',

'CHECKED' => ($value == 'Y') ? true : false,

'TYPE' => 'checkbox',

]);

break;
                                case
'double':
```

```
$return = CPrintForm ::input(
[
'NAME' => 'form[' . $key . ']',
'REQUIRED' => $arField[ 'isRequired' ],
'DISABLE' => $arField[ 'isReadOnly' ],
'MULTIPLE' => $arField[ 'isMultiple' ],
'VALUE' => $value,
'TYPE' => 'number'
]);
break;
                                case 'url':
$return = CPrintForm ::input(
[
'NAME' => 'form[' . $key . ']',
'REQUIRED' => $arField[ 'isRequired' ],
'DISABLE' => $arField[ 'isReadOnly' ],
'MULTIPLE' => $arField[ 'isMultiple' ],
'VALUE' => $value,
'TYPE' => 'text',
```

```
]);

break;
                                case
'integer':

$return = CPrintForm ::input(

[

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => $arField[ 'isMultiple' ],

'VALUE' => $value,

'TYPE' => 'number',

]);

break;
                                case 'user':

$arUser = [];

if(!empty($value))
                                        {

$arUser = CRest ::get('user.get', ['filter'
=> ['ID' => $value]]);
                                        }
```

```
$return = CPrintForm ::input(
[
'NAME' => 'form[' . $key . ']',
'REQUIRED' => $arField[ 'isRequired' ],
'DISABLE' => $arField[ 'isReadOnly' ],
'MULTIPLE' => $arField[ 'isMultiple' ],
'VALUE' => $value,
'TYPE' => 'number'
]);
if(!empty($arUser[ 'result' ]))
                                        {
$return .= '(';
$i = 0;
foreach($arUser[ 'result' ] as $val)
{
$i++;
if($i > 1)
{
$return .= ', ';
```

```
}

$return .= implode(' ', [$val[ 'NAME' ],
$val[ 'LAST_NAME' ]]);

}

$return .= ')';
                                        }


break;
                                case
'money':

list($money, $currency) = explode('|',
$value);

$return = CPrintForm ::input(

[

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => $arField[ 'isMultiple' ],

'VALUE' => $money,

'TYPE' => 'number',

]);

$arList = array_column($arResult[
```

```
'FIELD_VALUES_CURRENCY' ], 'FULL_NAME',
'CURRENCY');

$return .= CPrintForm ::select(

[

'NAME' => $key . '_CURRENCY',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => $arField[ 'isMultiple' ],

'VALUE' => $currency

],

$arList);

break;
                                case
'address':

$return = CPrintForm ::input(

[

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => $arField[ 'isMultiple' ],
```

```
'VALUE' => $value,

'TYPE' => 'text',

]);

break;
                                case
'resourcebooking':

//some code booking

$return = 'field not support';

break;
                                default:


$return = CPrintForm ::input(

[

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField[ 'isRequired' ],

'DISABLE' => $arField[ 'isReadOnly' ],

'MULTIPLE' => $arField[ 'isMultiple' ],

'VALUE' => $value,

'TYPE' => 'text',

]);
```

```
break;
                        }


                        if(strpos($key,
'UF_') === 0)
                        {

$sResultCustom .= '<div class="col-4 mt-3">'
. (($arField[ 'formLabel' ]) ? $arField[
'formLabel' ] : $arField[ 'title' ]) . ': '
. '</div>';

$sResultCustom .= '<div class="col-6 mt-3">'
. $return . '</div>';
                        }
                        else
                        {
                                $sResult .=
'<div class="col-4 mt-3">' . (($arField[
'formLabel' ]) ? $arField[ 'formLabel' ] :
$arField[ 'title' ]) . ': ' . '</div>';
                                $sResult .=
'<div class="col-6 mt-3">' . $return .
'</div>';
                        }
                }

?>
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/boo
tstrap/4.3.1/css/bootstrap.min.css"
crossorigin="anonymous">
        <script
src="https://code.jquery.com/jquery-
3.3.1.slim.min.js" crossorigin="anonymous">
</script>
```

```
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/
popper.js/1.14.7/umd/popper.min.js"
crossorigin="anonymous"></script>
        <script
src="https://stackpath.bootstrapcdn.com/boot
strap/4.3.1/js/bootstrap.min.js"
crossorigin="anonymous"></script>
        <script
src="https://ajax.googleapis.com/ajax/libs/j
query/3.3.1/jquery.min.js"></script>
        <script>
        $(document).ready(function()
        {
            $('#auto_form').on( 'submit',
function(el) {//event submit form
                el.preventDefault();//the
default action of the event will not be
triggered
                var formData = new
FormData(this);
                var xhr = new
XMLHttpRequest();
                xhr.open("POST",
'auto_form.php');
                xhr.onreadystatechange =
function () {
                    if (this.readyState ===
4) {
                        if (this.status >=
200 && this.status < 400) {
                            // Success!
                            var resp =
this.responseText;
                            try {
                                var json =
JSON.parse(resp);
```

```
                                    if (typeof
json.message !== 'undefined') {

alert(json.message);
                                    }
                                } catch (e) {
                                    return
false;
                                }
                            } else {
                                alert('error');
                            }
                        }
                    };
                    xhr.send(formData);
                });
            });
            </script>
            <div class="container">
                    <form id="auto_form"
action="" enctype="multipart/form-data"
method="post">
                            <?
if(!empty($arResult[ 'ITEM' ][ 'ID'
])):// for update entity
                                                ?>

<input type="hidden" name="form[ID]" value="
<?= $arResult[ 'ITEM' ][ 'ID' ] ?>">
                                    <?endif; ?>
                            <h2>Standard
fields</h2>
                            <div class="row">
                                    <?= $sResult
?>
                            </div>
                            <h2>Custom
```

```
fields</h2>
                        <div class="row">
                                <?=
$sResultCustom ?>
                        </div>
                        <div class="row">
                                <div
class="col-sm-10 mt-5">

<input type="submit" class="btn btn-primary"
value="Submit">
                                </div>
                        </div>
                </form>
        </div>
<?endif; ?>
```

Код файла **auto_form.php**, который сохраняет форму:

```
<?
        $arForm = [];
        foreach($_POST[ 'form' ] as $key =>
$item)
        {
                if(is_array($item))
                {
                        $arForm[ $key ] =
[];
                        foreach($item as $k
=> $val)
                        {
                                $arForm[
$key ][ $k ] = htmlspecialchars($val);
                        }
                }
```

```
                else
                {
                        $arForm[ $key ] =
htmlspecialchars($item);
                }
        }
        //make array multiple files for add
to custom field
        if(!empty($_FILES[ 'form' ][
'tmp_name' ]) && is_array($_FILES[ 'form' ][
'tmp_name' ]))
        {
                foreach($_FILES[ 'form' ][
'tmp_name' ] as $key => $files)
                {
                        if(is_array($files))
                        {

foreach($files as $k => $file)
                                {

$arForm[ $key ][ $k ] = [

"fileData" => [

$_FILES[ 'form' ][ 'name' ][ $key ][ $k ],

base64_encode(file_get_contents($file))

]
                                        ];
                                }
                        }
                        else
                        {
                                $arForm[
$key ] = [
```

```
"fileData" => [

$_FILES[ 'form' ][ 'name' ][ $key ],

base64_encode(file_get_contents($files))
                                        ]
                                ];
                        }
                }
        }
        $arResult = CRest
::get('crm.lead.fields', []);
        if(!empty($arResult[ 'result' ]))
        {
                foreach($arResult[ 'result'
] as $key => $prop)
                {
                        if(!isset($arForm[
$key ]))
                        {
                                if(!$prop[
'isReadOnly' ] && $prop[ 'type' ] != 'file')
                                {

if($prop[ 'type' ] == 'enumeration' &&
$prop[ 'isMultiple' ])
                                        {

//if type multiple enumeration to clean
selected value need send: [false]

$arForm[ $key ] = [false];
                                        }

elseif($prop[ 'isMultiple' ])
                                        {
```

```
$arForm[ $key ] = [];
                                        }
                                        else
                                        {

$arForm[ $key ] = '';
                                        }
                                }
                                continue;
                        }
                        //here may be any
check field example by type
                        if($prop[ 'type' ]
== 'crm_multifield')
                        {

if(isset($arForm[ $key ]))
                                {

$arForm[ $key ] = [['VALUE' => $arForm[ $key
]]];
                                }
                        }
                        elseif($prop[ 'type'
] == 'money')
                        {
                                $arForm[
$key ] = implode('|', [$arForm[ $key ],
$arForm[ $key . '_CURRENCY' ]]);

unset($arForm[ $key . '_CURRENCY' ]);
                        }
                }
        }
        $arForm['ID'] =
intVal($arForm['ID']);
```

```
        if($arForm[ 'ID' ] > 0)
        {
                $method = 'crm.lead.update';
                $arParams = [
                        'id' => $arForm[
'ID' ],
                        'fields' => $arForm
                ];
                $arMess = [
                        'success' => 'Lead
update',
                        'error' => 'Lead not
updated',
                ];
        }
        else
        {
                $method = 'crm.lead.add';
                $arParams = [
                        'fields' => $arForm
                ];
                $arMess = [
                        'success' => 'Lead
add',
                        'error' => 'Lead not
added',
                ];
        }
        $result = CRest ::get($method,
$arParams);
        if(!empty($result[ 'result' ]))
        {
                echo json_encode(['message'
=> $arMess[ 'success' ] . (($method ==
'crm.lead.add') ? ' ID:' . $result[ 'result'
] : '')]);
        }
```

```
        elseif(!empty($result[
'error_description' ]))
        {
                echo json_encode(['message'
=> $arMess[ 'error' ] . ': ' . $result[
'error_description' ]]);
        }
        else
        {
                echo json_encode(['message'
=> $arMess[ 'error' ]]);
        }
?>
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Как сделать свою карточку редактирования контакта

Пример автоматической генерирации карточки редактирования контакта со всеми полями созданными в Битрикс24 на странице приложения.

Некоторые типы полей не реализованы в данном примере, на месте полей с неподдерживаемым типом будет выводиться сообщение *field not support*.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

Код генерируемой формы:

```
<?
$ID = intVal($_REQUEST['ID']);

        class CPrintForm
        {
                /**
                 * @return string html
select
                 * @var $arParams array
params input keys: 'NAME', 'ID', 'TYPE',
'REQUIRED', 'CHECKED', 'DISABLE',
'MULTIPLE', 'VALUE'
                 */
                public static function
input($arParams)
```

```
                {
                        $count = 0;
                        $i = 0;
                        $sResult = '';
                        if
($arParams['MULTIPLE'] && $arParams['TYPE']
!= 'file')
                        {
                                $count = 2;
                        }
                        $value =
$arParams['VALUE'];
                        while ($i <= $count)
                        {

                                if ($count >
0)
                                {

$value = $arParams['VALUE'][$i];
                                }
                                $sResult .=
'<input class="form-control' .
(($arParams['TYPE'] == 'file') ? '-file' :
'') . '"';
                                if
(!empty($arParams['ID']))
                                {

$sResult .= ' id="' . $arParams['ID'] . '"';
                                }
                                if
(!empty($arParams['NAME']))
                                {

$sResult .= ' name="' . $arParams['NAME'] .
'' . (($arParams['MULTIPLE']) ? '[]' : '') .
```

```
'"';
                                    }
                                    if
(!empty($arParams['TYPE']))
                                    {

$sResult .= ' type="' . $arParams['TYPE'] .
'"';
                                    }
                                    if
(!empty($arParams['REQUIRED']))
                                    {

$sResult .= ' required';
                                    }
                                    if
(!empty($arParams['DISABLE']))
                                    {

$sResult .= ' disabled';
                                    }
                                    if
(!empty($arParams['CHECKED']))
                                    {

$sResult .= ' checked';
                                    }
                                    if
(!empty($arParams['MULTIPLE']))
                                    {//sometimes
work, not for standard type="text"

$sResult .= ' multiple';
                                    }
                                    if
(!empty($arParams['VALUE']))
                                    {
```

```
$sResult .= ' value="' . $value . '"';
                                }
                                $sResult .=
'>';
                                $i++;
                        }

                        return $sResult;
                }

                /**
                 * @return string html
select
                 * @var $arList array of
select options where key is value option and
value is title
                 * @var $arParams array
settings of select params keys: 'NAME',
'ID', 'REQUIRED', 'DISABLE','MULTIPLE',
'VALUE'
                 */
                public static function
select($arParams, $arList)
                {
                        $sResult = '';
                        if (!empty($arList)
&& is_array($arList))
                        {
                                $sResult .=
'<select class="form-control"' .

(($arParams['NAME']) ? ' name="' .

$arParams['NAME'] .

'' .
```

```
(($arParams['MULTIPLE']) ? '[]' : '') .

'"' : '') .

(($arParams['ID']) ? ' id="' .
$arParams['ID'] . '"' : '') .

(($arParams['REQUIRED']) ? ' required' : '')
.

(($arParams['DISABLE']) ? ' disabled' : '')
.

(($arParams['MULTIPLE']) ? ' multiple' : '')
.
                                        '>';
                                $value = [];
                                if
(is_array($arParams['VALUE']))
                                {

$value = $arParams['VALUE'];
                                }
                                else
                                {

$value[] = ($arParams['VALUE']) ?
$arParams['VALUE'] : '';
                                }
                                foreach
($arList as $key => $title)
                                {

$sResult .= '<option value="' .

$key .
```

```
'" ' .

((in_array($key, $value)) ? ' selected' :
'') .

'>' .

$title .

'</option>';
                                    }
                                    $sResult .=
'</select>';
                            }

                            return $sResult;
                    }

            }

            $arData = [
                    //Get all fields and
standard enum fields
                    'FIELDS' => [
                            'method' =>
'crm.contact.fields',
                            'params' => []
                    ],
                    'FIELD_VALUES_SOURCE_ID' =>
[
                            'method' =>
'crm.status.list',//only 50 first values
                            'params' =>
['filter' => ['ENTITY_ID' =>
'$result[FIELDS][SOURCE_ID][statusType]']]
                    ],
```

```
                'FIELD_VALUES_STATUS_ID' =>
[
                        'method' =>
'crm.status.list',//only 50 first values
                        'params' =>
['filter' => ['ENTITY_ID' =>
'$result[FIELDS][STATUS_ID][statusType]']]
                ],
                'FIELD_VALUES_CURRENCY' => [
                        'method' =>
'crm.currency.list',//only 50 first values
                        'params' =>
['filter' => ['ENTITY_ID' =>
'$result[FIELDS][STATUS_ID][statusType]']]
                ],
                'OWNER_TYPE' => [
                        'method' =>
'crm.enum.ownertype',
                        'params' => []
                ],
        ];
        if ($ID > 0)
        {//get item and standard enum field
values if is update form
                $arData['ITEM'] = [
                        'method' =>
'crm.contact.get',
                        'params' => ['id' =>
$ID]
                ];
                $arData['VALUE_LEAD_ID'] = [
                        'method' =>
'crm.lead.get',
                        'params' => ['id' =>
'$result[ITEM][LEAD_ID]']
                ];
        }
```

```
        $arResult =
CRest::callBatch($arData, 0);

        $arResult = $arResult['result']
['result'];
        $sResult = '';
        $sResultCustom = '';
if (is_array($arResult['FIELDS'])):
        if (isset($arResult['FIELDS']
['COMPANY_ID']))//deprecated use
crm.contact.company.items.get
        {
                unset($arResult['FIELDS']
['COMPANY_ID']);
        }
        if (isset($arResult['FIELDS']
['COMPANY_IDS']))//use
crm.contact.company.items.get
        {
                unset($arResult['FIELDS']
['COMPANY_IDS']);
        }

        foreach ($arResult['FIELDS'] as $key
=> $arField)
        {
                $value = '';
                $return = '';
                if (!empty($arResult['ITEM']
[$key]))
                {
                        $value =
$arResult['ITEM'][$key];
                }
                $arList =
(isset($arResult['FIELD_VALUES_' . $key])) ?
```

```
$arResult['FIELD_VALUES_' . $key] : [];
                switch ($arField['type'])
                {
                        case 'crm_status':
                                if
(empty($arList))
                                {

$arFieldsStatus = \CRest::get(

'crm.status.list',

['filter' => ['ENTITY_ID' =>
$arField['statusType']]]
                                        );
                                        if
(!empty($arFieldsStatus['result']))
                                        {

$arList = $arFieldsStatus['result'];
                                        }
                                }
                                $arList =
array_column($arList, 'NAME', 'STATUS_ID');

                                $return =
CPrintForm::select(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],
```

```
'VALUE' => $value
                                        ],

$arList
                                );
                                break;
                        case 'crm_currency':
                                $arList =
array_column($arResult['FIELD_VALUES_CURRENC
Y'], 'FULL_NAME', 'CURRENCY');
                                $return =
CPrintForm::select(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value
                                        ],

$arList
                                );
                                break;
                        case 'enumeration':
                                if
(!empty($arField['items']))
                                {

$arList = array_column($arField['items'],
'VALUE', 'ID');
                                }
```

```
                                    $return =
CPrintForm::select(
                                            [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value
                                            ],

$arList
                                    );
                                    break;
                            case
'crm_multifield'://its simple example: need
multifield, check data type and more...
                                    if
(!empty($value) && is_array($value))
                                    {

$value = reset($value)['VALUE'];
                                    }
                                    $return =
CPrintForm::input(
                                            [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],
```

```
'MULTIPLE' => false,

'VALUE' => $value,

'TYPE' => 'text',
                                        ]
                                );
                                break;
                        case 'crm_lead':
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'text',
                                        ]
                                );

                                if
(!empty($arResult['VALUE_LEAD_ID']) &&
$value == $arResult['VALUE_LEAD_ID']['ID'])
                                {

$return .= '(' . $arResult['VALUE_LEAD_ID']
['TITLE'] . ')';
                                }
                                break;
                        case 'file':
```

```
                                    $return =
CPrintForm::input(
                                            [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'file',
                                            ]
                                    );
                                    if
($arField['isMultiple'])
                                    {
                                            if
(is_array($value))
                                            {

foreach ($value as $k => $val)

{

if (!empty($val['downloadUrl']))

{

$return .= '<br/><a href="' .
$val['downloadUrl'] . '">old file ' . $k .
'</a>';

}
```

```
}
                                        }
                                }
                                else
                                {
                                        if
(!empty($value['downloadUrl']))
                                        {

$return .= '<br/><a href="' .
$value['downloadUrl'] . '">old file</a>';
                                        }
                                }
                                break;
                        case 'date':
                                if
(!empty($value))
                                {

$value = date('Y-m-d', strtotime($value));
                                }
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'date',
```

```
                                        ]
                                );
                                break;
                        case 'datetime':
                                if
(!empty($value))
                                {

$value = date('Y-m-d\TH:i:s',
strtotime($value));
                                }
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'datetime-local',
                                        ]
                                );
                                break;
                        case 'char':
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],
```

```
'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => 'Y',

'CHECKED' => ($value == 'Y') ? true : false,

'TYPE' => 'checkbox',
                                        ]
                                );
                                break;

                        case 'boolean':
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => '1',

'CHECKED' => ($value == 'Y') ? true : false,

'TYPE' => 'checkbox',
                                        ]
                                );
                                break;
                        case 'double':
                                $return =
```

```
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'number'
                                        ]
                                );
                                break;
                        case 'user':
                                $arUser =
[];
                                if
(!empty($value))
                                {

$arUser = CRest::get('user.get', ['filter'
=> ['ID' => $value]]);
                                }
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],
```

```
'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'number'
                                        ]
                                );
                                if
(!empty($arUser['result']))
                                {

$return .= '(';
                                        $i =
0;

foreach ($arUser['result'] as $val)
                                        {

$i++;

if ($i > 1)

{

$return .= ', ';

}

$return .= implode(' ', [$val['NAME'],
$val['LAST_NAME']]);
                                        }

$return .= ')';
                                }

                                break;
                        case 'url':
```

```
                                    $return =
CPrintForm::input(
                                            [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'text',
                                            ]
                                    );
                                    break;
                            case 'integer':
                                    $return =
CPrintForm::input(
                                            [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'number',
                                            ]
                                    );
                                    break;
```

```
                        case 'money':
                                list($money,
$currency) = explode('|', $value);
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $money,

'TYPE' => 'number',
                                        ]
                                );
                                $arList =
array_column($arResult['FIELD_VALUES_CURRENC
Y'], 'FULL_NAME', 'CURRENCY');
                                $return .=
CPrintForm::select(
                                        [

'NAME' => $key . '_CURRENCY',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $currency
                                        ],
```

```
$arList
                                );
                                break;
                        case 'address':
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'text',
                                        ]
                                );
                                break;
                        case
'resourcebooking':
                                //some code
booking
                                $return =
'field not support';
                                break;
                        default:

                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',
```

```
'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'text',
                                        ]
                                );

                                break;
                }

                if (strpos($key, 'UF_') ===
0)
                {
                        $sResultCustom .=
'<div class="col-4 mt-3">' .

(($arField['formLabel']) ?
$arField['formLabel'] : $arField['title']) .
                                ': ' .
                                '</div>';
                        $sResultCustom .=
'<div class="col-6 mt-3">' . $return .
'</div>';
                }
                else
                {
                        $sResult .= '<div
class="col-4 mt-3">' .

(($arField['formLabel']) ?
$arField['formLabel'] : $arField['title']) .
```

```
                                    ': ' .
                                    '</div>';
                            $sResult .= '<div
class="col-6 mt-3">' . $return . '</div>';
                }
        }

        ?>
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/boo
tstrap/4.3.1/css/bootstrap.min.css"
                  crossorigin="anonymous">
        <script
src="https://code.jquery.com/jquery-
3.3.1.slim.min.js" crossorigin="anonymous">
</script>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/
popper.js/1.14.7/umd/popper.min.js"

crossorigin="anonymous"></script>
        <script
src="https://stackpath.bootstrapcdn.com/boot
strap/4.3.1/js/bootstrap.min.js"

crossorigin="anonymous"></script>
        <script
src="https://ajax.googleapis.com/ajax/libs/j
query/3.3.1/jquery.min.js"></script>
        <script>
            $(document).ready(function () {

$('#auto_form').on('submit', function (el)
{//event submit form

el.preventDefault();//the default action of
the event will not be triggered
```

```
var formData = new FormData(this);
var xhr = new XMLHttpRequest();
xhr.open("POST", 'auto_form.php');
xhr.onreadystatechange = function () {
if (this.readyState === 4)
{
if (this.status >= 200 && this.status < 400)
{
// Success!
var resp = this.responseText;
try
{
var json = JSON.parse(resp);
if (typeof json.message !== 'undefined')
{
alert(json.message);
}
} catch (e)
```

```
{

return false;

}

}

else

{

alert('error');

}

}
                                        };

xhr.send(formData);
                                 });
                         });
                 </script>
        <div class="container">
                         <form id="auto_form"
action="" enctype="multipart/form-data"
method="post">
                                 <?if
(!empty($arResult['ITEM']['ID'])):// for
update entity   ?>

<input type="hidden" name="form[ID]" value="
<?=$arResult[ 'ITEM' ][ 'ID' ]    ?>">
                                 <?endif;?>
                                 <h2>Standard
fields</h2>
                                 <div
```

```
class="row">
                                        <?
=$sResult?>
                                </div>
                                <h2>Custom
fields</h2>
                                <div
class="row">
                                        <?
=$sResultCustom?>
                                </div>
                                <div
class="row">
                                        <div
class="col-sm-10 mt-5">

<input type="submit" class="btn btn-primary"
value="Submit">

</div>
                                </div>
                        </form>
                </div>

        <? endif;?>
```

Файл **auto_form.php**:

```
<?

        $arForm = [];
        foreach ($_POST['form'] as $key =>
$item)
        {
                if (is_array($item))
```

```
                {
                        $arForm[$key] = [];
                        foreach ($item as $k
=> $val)
                        {

$arForm[$key][$k] = htmlspecialchars($val);
                        }
                }
                else
                {
                        $arForm[$key] =
htmlspecialchars($item);
                }
        }
        //make array multiple files for add
to custom field
        if (!empty($_FILES['form']
['tmp_name']) && is_array($_FILES['form']
['tmp_name']))
        {
                foreach ($_FILES['form']
['tmp_name'] as $key => $files)
                {
                        if
(is_array($files))
                        {
                                foreach
($files as $k => $file)
                                {

$arForm[$key][$k] = [

"fileData" => [

$_FILES['form']['name'][$key][$k],
```

```
base64_encode(file_get_contents($file))

]
                                                ];
                                        }
                                }
                                else
                                {

$arForm[$key] = [

"fileData" => [

$_FILES['form']['name'][$key],

base64_encode(file_get_contents($files))
                                                ]
                                        ];
                                }
                        }
                }
                $arResult =
CRest::get('crm.contact.fields', []);
                if (!empty($arResult['result']))
                {
                        foreach ($arResult['result']
as $key => $prop)
                        {
                                if
(!isset($arForm[$key]))
                                {
                                        if
(!$prop['isReadOnly'] && $prop['type'] !=
'file')
                                        {
                                                if
($prop['type'] == 'enumeration' &&
```

```
$prop['isMultiple'])
                                        {

//if type multiple enumeration to clean
selected value need send: [false]

$arForm[$key] = [false];
                                        }

elseif ($prop['isMultiple'])
                                        {

$arForm[$key] = [];
                                        }
                                        else
                                        {

$arForm[$key] = '';
                                        }
                                }
                                continue;
                        }
                        //here may be any
check field example by type
                        if ($prop['type'] ==
'crm_multifield')
                        {
                                if
(isset($arForm[$key]))
                                {

$arForm[$key] = [['VALUE' =>
$arForm[$key]]];
                                }
                        }
                        elseif
($prop['type'] == 'money')
```

```
                        {
$arForm[$key] = implode('|', [$arForm[$key],
$arForm[$key . '_CURRENCY']]);

unset($arForm[$key . '_CURRENCY']);
                        }
                }
        }
        $arForm['ID'] =
intVal($arForm['ID']);
        if ($arForm['ID'] > 0)
        {
                $method =
'crm.contact.update';
                $arParams = [
                        'id' =>
$arForm['ID'],
                        'fields' => $arForm
                ];
                $arMess = [
                        'success' =>
'Contact update',
                        'error' => 'Contact
not updated',
                ];
        }
        else
        {
                $method = 'crm.contact.add';
                $arParams = [
                        'fields' => $arForm
                ];
                $arMess = [
                        'success' =>
'Contact add',
                        'error' => 'Contact
```

```
not added',
                ];
        }
        $result = CRest::get($method,
$arParams);
        if (!empty($result['result']))
        {
                echo json_encode(
                        ['message' =>
$arMess['success'] . (($method ==
'crm.contact.add') ? ' ID:' .
$result['result'] : '')]
                );
        }
        elseif
(!empty($result['error_description']))
        {
                echo json_encode(['message'
=> $arMess['error'] . ': ' .
$result['error_description']]);
        }
        else
        {
                echo json_encode(['message'
=> $arMess['error']]);
        }

        ?>
```

Как сделать свою карточку редактирования компании

Пример автоматической генерирации карточки редактирования компании со всеми полями созданными в Битрикс24 на странице вашего приложения.

Некоторые типы полей не реализованы в примере, на месте полей с неподдерживаемым типом будет выводиться сообщение *field not support*.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

Код генерируемой формы:

```
<?
        $ID = intVal($_REQUEST['ID']);
        class CPrintForm
        {
                /**
                 * @return string html
select
                 * @var $arParams array
params input keys: 'NAME', 'ID', 'TYPE',
'REQUIRED', 'CHECKED', 'DISABLE',
'MULTIPLE', 'VALUE'
                 */
                public static function
input($arParams)
                {
```

```
                        $count = 0;
                        $i = 0;
                        $sResult = '';
                        if
($arParams['MULTIPLE'] && $arParams['TYPE']
!= 'file')
                        {
                                $count = 2;
                        }
                        $value =
$arParams['VALUE'];
                        while ($i <= $count)
                        {

                                if ($count >
0)
                                {

$value = $arParams['VALUE'][$i];
                                }
                                $sResult .=
'<input class="form-control' .
(($arParams['TYPE'] == 'file') ? '-file' :
'') . '"';
                                if
(!empty($arParams['ID']))
                                {

$sResult .= ' id="' . $arParams['ID'] . '"';
                                }
                                if
(!empty($arParams['NAME']))
                                {

$sResult .= ' name="' . $arParams['NAME'] .
'' . (($arParams['MULTIPLE']) ? '[]' : '') .
'"';
```

```
                                    }
                                    if
(!empty($arParams['TYPE']))
                                    {

$sResult .= ' type="' . $arParams['TYPE'] .
'"';
                                    }
                                    if
(!empty($arParams['REQUIRED']))
                                    {

$sResult .= ' required';
                                    }
                                    if
(!empty($arParams['DISABLE']))
                                    {

$sResult .= ' disabled';
                                    }
                                    if
(!empty($arParams['CHECKED']))
                                    {

$sResult .= ' checked';
                                    }
                                    if
(!empty($arParams['MULTIPLE']))
                                    {//sometimes
work, not for standard type="text"

$sResult .= ' multiple';
                                    }
                                    if
(!empty($arParams['VALUE']))
                                    {
```

```
$sResult .= ' value="' . $value . '"';
                                }
                                $sResult .=
'>';
                                $i++;
                        }

                        return $sResult;
                }

                /**
                 * @return string html
select
                 * @var $arList array of
select options where key is value option and
value is title
                 * @var $arParams array
settings of select params keys: 'NAME',
'ID', 'REQUIRED', 'DISABLE','MULTIPLE',
'VALUE'
                 */
                public static function
select($arParams, $arList)
                {
                        $sResult = '';
                        if (!empty($arList)
&& is_array($arList))
                        {
                                $sResult .=
'<select class="form-control"' .

(($arParams['NAME']) ? ' name="' .

$arParams['NAME'] .

'' .
```

```
(($arParams['MULTIPLE']) ? '[]' : '') .

'"' : '') .

(($arParams['ID']) ? ' id="' .
$arParams['ID'] . '"' : '') .

(($arParams['REQUIRED']) ? ' required' : '')
.

(($arParams['DISABLE']) ? ' disabled' : '')
.

(($arParams['MULTIPLE']) ? ' multiple' : '')
.
                                        '>';
                                $value = [];
                                if
(is_array($arParams['VALUE']))
                                {

$value = $arParams['VALUE'];
                                }
                                else
                                {

$value[] = ($arParams['VALUE']) ?
$arParams['VALUE'] : '';
                                }
                                foreach
($arList as $key => $title)
                                {

$sResult .= '<option value="' .

$key .
```

```
'" ' .

((in_array($key, $value)) ? ' selected' :
'') .

'>' .

$title .

'</option>';
                                    }
                                    $sResult .=
'</select>';
                            }

                            return $sResult;
                    }
            }

            $arData = [
                    //Get all fields and
standard enum fields
                    'FIELDS' => [
                            'method' =>
'crm.company.fields',
                            'params' => []
                    ],
                    'FIELD_VALUES_SOURCE_ID' =>
[
                            'method' =>
'crm.status.list',//only 50 first values
                            'params' =>
['filter' => ['ENTITY_ID' =>
'$result[FIELDS][SOURCE_ID][statusType]']]
                    ],
                    'FIELD_VALUES_STATUS_ID' =>
[
```

```
                        'method' =>
'crm.status.list',//only 50 first values
                        'params' =>
['filter' => ['ENTITY_ID' =>
'$result[FIELDS][STATUS_ID][statusType]']]
                ],
                'FIELD_VALUES_CURRENCY' => [
                        'method' =>
'crm.currency.list',//only 50 first values
                        'params' =>
['filter' => ['ENTITY_ID' =>
'$result[FIELDS][STATUS_ID][statusType]']]
                ],
                'OWNER_TYPE' => [
                        'method' =>
'crm.enum.ownertype',
                        'params' => []
                ],

        ];
        if ($ID > 0)
        {//get item and standard enum field
values if is update form
                $arData['ITEM'] = [
                        'method' =>
'crm.company.get',
                        'params' => ['id' =>
$ID]
                ];
                $arData['VALUE_CONTACT_ID']
= [
                        'method' =>
'crm.company.contact.items.get',
                        'params' => ['id' =>
$ID]
                ];
                $arData['VALUE_LEAD_ID'] = [
```

```
                        'method' =>
'crm.lead.get',
                        'params' => ['id' =>
'$result[ITEM][LEAD_ID]']
                ];
        }

        $arResult =
CRest::callBatch($arData, 0);

        $arResult = $arResult['result']
['result'];
        $sResult = '';
        $sResultCustom = '';
if (is_array($arResult['FIELDS'])):

        foreach ($arResult['FIELDS'] as $key
=> $arField)
        {
                $value = '';
                $return = '';
                if (!empty($arResult['ITEM']
[$key]))
                {
                        $value =
$arResult['ITEM'][$key];
                }
                $arList =
(isset($arResult['FIELD_VALUES_' . $key])) ?
$arResult['FIELD_VALUES_' . $key] : [];
                switch ($arField['type'])
                {
                        case 'crm_status':
                                if
(empty($arList))
                                {
```

```
$arFieldsStatus = \CRest::get(

'crm.status.list',

['filter' => ['ENTITY_ID' =>
$arField['statusType']]]
                                        );
                                        if
(!empty($arFieldsStatus['result']))
                                        {

$arList = $arFieldsStatus['result'];
                                        }
                                }
                                $arList =
array_column($arList, 'NAME', 'STATUS_ID');

                                $return =
CPrintForm::select(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value
                                        ],

$arList
                                );
                                break;
                        case 'crm_currency':
                                $arList =
```

```
array_column($arResult['FIELD_VALUES_CURRENC
Y'], 'FULL_NAME', 'CURRENCY');
                                $return =
CPrintForm::select(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value
                                        ],

$arList
                                );
                                break;
                        case 'enumeration':
                                if
(!empty($arField['items']))
                                {

$arList = array_column($arField['items'],
'VALUE', 'ID');
                                }

                                $return =
CPrintForm::select(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],
```

```
'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value
                                        ],

$arList
                                );
                                break;
                            case
'crm_multifield'://its simple example: need
multifield, check data type and more...
                                if
(!empty($value) && is_array($value))
                                {

$value = reset($value)['VALUE'];
                                }
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => false,

'VALUE' => $value,

'TYPE' => 'text',
                                        ]
                                );
                                break;
```

```
                        case 'crm_lead':
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'text',
                                        ]
                                );

                                if
(!empty($arResult['VALUE_LEAD_ID']) &&
$value == $arResult['VALUE_LEAD_ID']['ID'])
                                {

$return .= '(' . $arResult['VALUE_LEAD_ID']
['TITLE'] . ')';
                                }
                                break;
                        case 'crm_contact':
                                $arContact =
[];
                                if
(!empty($arResult['VALUE_' . $key]))
                                {

$arContact = $arResult['VALUE_' . $key];
                                }
```

```
                                elseif
(!empty($value))
                                {

$arContact = CRest::get('crm.contact.list',
['filter' => ['ID' => $value]]);
                                }

                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'text',
                                        ]
                                );
                                if
(!empty($arContact['result']))
                                {

$return .= '(';
                                        $i =
0;

foreach ($arContact['result'] as $val)
                                        {

$i++;
```

```
if ($i > 1)

{

$return .= ', ';

}

$return .= implode(' ', [$val['NAME'],
$val['LAST_NAME']]);
                                        }

$return .= ')';
                                }
                                break;
                        case 'file':
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'file',
                                        ]
                                );
                                if
($arField['isMultiple'])
                                {
```

```
                                                if
(is_array($value))
                                                {

foreach ($value as $k => $val)

{

if (!empty($val['downloadUrl']))

{

$return .= '<br/><a href="' .
$val['downloadUrl'] . '">old file ' . $k .
'</a>';

}

}
                                                }
                                        }
                                        else
                                        {
                                                if
(!empty($value['downloadUrl']))
                                                {

$return .= '<br/><a href="' .
$value['downloadUrl'] . '">old file</a>';
                                                }
                                        }
                                        break;
                                case 'date':
                                        if
(!empty($value))
                                        {
```

```
$value = date('Y-m-d', strtotime($value));
                                }
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'date',
                                        ]
                                );
                                break;
                        case 'datetime':
                                if
(!empty($value))
                                {

$value = date('Y-m-d\TH:i:s',
strtotime($value));
                                }
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],
```

```
'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'datetime-local',
                                        ]
                                );
                                break;
                        case 'char':
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => 'Y',

'CHECKED' => ($value == 'Y') ? true : false,

'TYPE' => 'checkbox',
                                        ]
                                );
                                break;

                        case 'boolean':
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',
```

```
'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => '1',

'CHECKED' => ($value == 'Y') ? true : false,

'TYPE' => 'checkbox',
                                        ]
                                );
                                break;
                        case 'double':
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'number'
                                        ]
                                );
                                break;
                        case 'user':
                                $arUser =
[];
```

```
                                    if
(!empty($value))
                                    {

$arUser = CRest::get('user.get', ['filter'
=> ['ID' => $value]]);
                                    }
                                    $return =
CPrintForm::input(
                                            [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'number'
                                            ]
                                    );
                                    if
(!empty($arUser['result']))
                                    {

$return .= '(';
                                            $i =
0;

foreach ($arUser['result'] as $val)
                                            {

$i++;
```

```
if ($i > 1)

{

$return .= ', ';

}

$return .= implode(' ', [$val['NAME'],
$val['LAST_NAME']]);
                                        }

$return .= ')';
                                }

                                break;
                        case 'url':
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'text',
                                        ]
                                );
                                break;
                        case 'integer':
                                $return =
```

```
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'number',
                                        ]
                                );
                                break;
                        case 'money':
                                list($money,
$currency) = explode('|', $value);
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $money,

'TYPE' => 'number',
                                        ]
                                );
```

```
                                $arList =
array_column($arResult['FIELD_VALUES_CURRENC
Y'], 'FULL_NAME', 'CURRENCY');
                                $return .=
CPrintForm::select(
                                        [

'NAME' => $key . '_CURRENCY',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $currency
                                        ],

$arList
                                );
                                break;
                        case 'address':
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'text',
```

```
                                        ]
                                );
                                break;
                        case
'resourcebooking':
                                //some code
booking
                                $return =
'field not support';
                                break;
                        default:

                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'text',
                                        ]
                                );

                                break;
                }

                if (strpos($key, 'UF_') ===
0)
                {
                        $sResultCustom .=
```

```
'<div class="col-4 mt-3">' .

(($arField['formLabel']) ?
$arField['formLabel'] : $arField['title']) .
                                ': ' .
                                '</div>';
                        $sResultCustom .=
'<div class="col-6 mt-3">' . $return .
'</div>';
                }
                else
                {
                        $sResult .= '<div
class="col-4 mt-3">' .

(($arField['formLabel']) ?
$arField['formLabel'] : $arField['title']) .
                                ': ' .
                                '</div>';
                        $sResult .= '<div
class="col-6 mt-3">' . $return . '</div>';
                }
        }

        ?>
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/boo
tstrap/4.3.1/css/bootstrap.min.css"
                  crossorigin="anonymous">
        <script
src="https://code.jquery.com/jquery-
3.3.1.slim.min.js" crossorigin="anonymous">
</script>
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/
popper.js/1.14.7/umd/popper.min.js"
```

```
crossorigin="anonymous"></script>
        <script
src="https://stackpath.bootstrapcdn.com/boot
strap/4.3.1/js/bootstrap.min.js"

crossorigin="anonymous"></script>
        <script
src="https://ajax.googleapis.com/ajax/libs/j
query/3.3.1/jquery.min.js"></script>
        <script>
            $(document).ready(function () {

$('#auto_form').on('submit', function (el)
{//event submit form

el.preventDefault();//the default action of
the event will not be triggered
                                            var
formData = new FormData(this);
                                            var
xhr = new XMLHttpRequest();

xhr.open("POST", 'auto_form.php');

xhr.onreadystatechange = function () {

if (this.readyState === 4)

{

if (this.status >= 200 && this.status < 400)

{

// Success!

var resp = this.responseText;
```

```
try
{
var json = JSON.parse(resp);
if (typeof json.message !== 'undefined')
{
alert(json.message);
}
} catch (e)
{
return false;
}
}
else
{
alert('error');
}
}
                                                    };
xhr.send(formData);
```

```
                                    });
                            });
                    </script>
            <div class="container">
                            <form id="auto_form"
action="" enctype="multipart/form-data"
method="post">
                                    <?if
(!empty($arResult['ITEM']['ID'])):// for
update entity?>

<input type="hidden" name="form[ID]" value="
<?=$arResult[ 'ITEM' ][ 'ID' ]?>">
                                            <?
endif;?>
                                    <h2>Standard
fields</h2>
                                    <div
class="row">
                                            <?
=$sResult?>
                                    </div>
                                    <h2>Custom
fields</h2>
                                    <div
class="row">
                                            <?
=$sResultCustom?>
                                    </div>
                                    <div
class="row">
                                            <div
class="col-sm-10 mt-5">

<input type="submit" class="btn btn-primary"
value="Submit">
```

```
</div>
                                        </div>
                                </form>
                        </div>
                <?endif;?>
```

Файл **auto_form.php**:

```
<?
$arForm = [];
foreach ($_POST['form'] as $key => $item)
{
        if (is_array($item))
        {
                $arForm[$key] = [];
                foreach ($item as $k =>
$val)
                {
                        $arForm[$key][$k] =
htmlspecialchars($val);
                }
        }
        else
        {
                $arForm[$key] =
htmlspecialchars($item);
        }
}
//make array multiple files for add to
custom field
if (!empty($_FILES['form']['tmp_name']) &&
is_array($_FILES['form']['tmp_name']))
{
        foreach ($_FILES['form']['tmp_name']
as $key => $files)
        {
```

```
                if (is_array($files))
                {
                        foreach ($files as
$k => $file)
                        {

$arForm[$key][$k] = [

"fileData" => [

$_FILES['form']['name'][$key][$k],

base64_encode(file_get_contents($file))
                                        ]
                                ];
                        }
                }
                else
                {
                        $arForm[$key] = [
                                "fileData"
=> [

$_FILES['form']['name'][$key],

base64_encode(file_get_contents($files))
                                ]
                        ];
                }
        }
}
$arResult = CRest::get('crm.company.fields',
[]);
if (!empty($arResult['result']))
{
        foreach ($arResult['result'] as $key
=> $prop)
```

```
        {
                if (!isset($arForm[$key]))
                {
                        if
(!$prop['isReadOnly'] && $prop['type'] !=
'file')
                        {
                                if
($prop['type'] == 'enumeration' &&
$prop['isMultiple'])
                                {
                                        //if
type multiple enumeration to clean selected
value need send: [false]

$arForm[$key] = [false];
                                }
                                elseif
($prop['isMultiple'])
                                {

$arForm[$key] = [];
                                }
                                else
                                {

$arForm[$key] = '';
                                }
                        }
                        continue;
                }
                //here may be any check
field example by type
                if ($prop['type'] ==
'crm_multifield')
                {
                        if
```

```
(isset($arForm[$key]))
                        {

$arForm[$key] = [['VALUE' =>
$arForm[$key]]];
                        }
                }
                elseif ($prop['type'] ==
'money')
                {
                        $arForm[$key] =
implode('|', [$arForm[$key], $arForm[$key .
'_CURRENCY']]);
                        unset($arForm[$key .
'_CURRENCY']);
                }
        }
}
$arForm['ID'] = intVal($arForm['ID']);
if ($arForm['ID'] > 0)
{
        $method = 'crm.company.update';
        $arParams = [
                'id' => $arForm['ID'],
                'fields' => $arForm
        ];
        $arMess = [
                'success' => 'Company
update',
                'error' => 'Company not
updated',
        ];
}
else
{
        $method = 'crm.company.add';
        $arParams = [
```

```
                'fields' => $arForm
        ];
        $arMess = [
                'success' => 'Company add',
                'error' => 'Company not
added',
        ];
}
$result = CRest::get($method, $arParams);
if (!empty($result['result']))
{
        echo json_encode(
                ['message' =>
$arMess['success'] . (($method ==
'crm.company.add') ? ' ID:' .
$result['result'] : '')]
        );
}
elseif
(!empty($result['error_description']))
{
        echo json_encode(['message' =>
$arMess['error'] . ': ' .
$result['error_description']]);
}
else
{
        echo json_encode(['message' =>
$arMess['error']]);
}
?>
```

Как сделать свою карточку редактирования сделки

Пример автоматической генерировации карточки редактирования сделки со всеми полями созданными в Битрикс24 на странице приложения.

Некоторые типы полей не реализованы в примере, на месте полей с неподдерживаемым типом будет выводиться сообщение *field not support*.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

Код генерируемой формы:

```
<?
$ID = intVal($_REQUEST['ID']);
class CPrintForm
{
        /**
         * @return string html select
         * @var $arParams array params input
keys: 'NAME', 'ID', 'TYPE', 'REQUIRED',
'CHECKED', 'DISABLE', 'MULTIPLE', 'VALUE'
         */
        public static function
input($arParams)
        {
                $count = 0;
                $i = 0;
```

```
                $sResult = '';
                if ($arParams['MULTIPLE'] &&
$arParams['TYPE'] != 'file')
                {
                        $count = 2;
                }
                $value = $arParams['VALUE'];
                while ($i <= $count)
                {

                        if ($count > 0)
                        {
                                $value =
$arParams['VALUE'][$i];
                        }
                        $sResult .= '<input
class="form-control' . (($arParams['TYPE']
== 'file') ? '-file' : '') . '"';
                        if
(!empty($arParams['ID']))
                        {
                                $sResult .=
' id="' . $arParams['ID'] . '"';
                        }
                        if
(!empty($arParams['NAME']))
                        {
                                $sResult .=
' name="' .

$arParams['NAME'] .
                                        '' .

(($arParams['MULTIPLE']) ? '[]' : '') .
                                        '"';
                        }
                        if
```

```
(!empty($arParams['TYPE']))
                        {
                                $sResult .=
' type="' . $arParams['TYPE'] . '"';
                        }
                        if
(!empty($arParams['REQUIRED']))
                        {
                                $sResult .=
' required';
                        }
                        if
(!empty($arParams['DISABLE']))
                        {
                                $sResult .=
' disabled';
                        }
                        if
(!empty($arParams['CHECKED']))
                        {
                                $sResult .=
' checked';
                        }
                        if
(!empty($arParams['MULTIPLE']))
                        {//sometimes work,
not for standard type="text"
                                $sResult .=
' multiple';
                        }
                        if
(!empty($arParams['VALUE']))
                        {
                                $sResult .=
' value="' . $value . '"';
                        }
                        $sResult .= '>';
```

```
                        $i++;
                }

                return $sResult;
        }

        /**
         * @return string html select
         * @var $arList array of select
options where key is value option and value
is title
         * @var $arParams array settings of
select params keys: 'NAME', 'ID',
'REQUIRED', 'DISABLE','MULTIPLE', 'VALUE'
         */
        public static function
select($arParams, $arList)
        {
                $sResult = '';
                if (!empty($arList) &&
is_array($arList))
                {
                        $sResult .= '<select
class="form-control"' .

(($arParams['NAME']) ? ' name="' .

$arParams['NAME'] .
                                        '' .

(($arParams['MULTIPLE']) ? '[]' : '') .
                                        '"'
: '') .

(($arParams['ID']) ? ' id="' .
$arParams['ID'] . '"' : '') .
```

```
(($arParams['REQUIRED']) ? ' required' : '')
.

(($arParams['DISABLE']) ? ' disabled' : '')
.

(($arParams['MULTIPLE']) ? ' multiple' : '')
.
                                '>';
                        $value = [];
                        if
(is_array($arParams['VALUE']))
                        {
                                $value =
$arParams['VALUE'];
                        }
                        else
                        {
                                $value[] =
($arParams['VALUE']) ? $arParams['VALUE'] :
'';
                        }
                        foreach ($arList as
$key => $title)
                        {
                                $sResult .=
'<option value="' .
                                        $key
.
                                        '" '
.

((in_array($key, $value)) ? ' selected' :
'') .
                                        '>'
.
```

```
$title .

'</option>';
                        }
                        $sResult .=
'</select>';
                }

                return $sResult;
        }
}

$arData = [
        //Get all fields and standard enum
fields
        'FIELDS' => [
                'method' =>
'crm.deal.fields',
                'params' => []
        ],
        'FIELD_VALUES_SOURCE_ID' => [
                'method' =>
'crm.status.list',//only 50 first values
                'params' => ['filter' =>
['ENTITY_ID' => '$result[FIELDS][SOURCE_ID]
[statusType]']]
        ],
        'FIELD_VALUES_STATUS_ID' => [
                'method' =>
'crm.status.list',//only 50 first values
                'params' => ['filter' =>
['ENTITY_ID' => '$result[FIELDS][STATUS_ID]
[statusType]']]
        ],
        'FIELD_VALUES_CURRENCY' => [
                'method' =>
'crm.currency.list',//only 50 first values
```

```
                'params' => ['filter' =>
['ENTITY_ID' => '$result[FIELDS][STATUS_ID]
[statusType]']]
        ],
        'OWNER_TYPE' => [
                'method' =>
'crm.enum.ownertype',
                'params' => []
        ],

];
if ($ID > 0)
{//get item and standard enum field values
if is update form
        $arData['ITEM'] = [
                'method' => 'crm.deal.get',
                'params' => ['id' => $ID]
        ];
        $arData['VALUE_CATEGORY_ID'] =
[//only 50 first unlocked values

'method' => 'crm.dealcategory.list',

'params' => ['filter' => ['IS_LOCKED' =>
'N']]
        ];
        $arData['VALUE_LEAD_ID'] = [
                'method' => 'crm.lead.get',
                'params' => ['id' =>
'$result[ITEM][LEAD_ID]']
        ];
        //QUOTE_ID is deprecated use
crm.quote.list:
        $arData['VALUE_QUOTE_ID'] = [
                'method' =>
'crm.quote.list',
                'params' => ['filter' =>
```

```
['DEAL_ID' => $ID]]
        ];
}

$arResult = CRest::callBatch($arData, 0);

$arResult = $arResult['result']['result'];
$sResult = '';
$sResultCustom = '';
if (is_array($arResult['FIELDS'])):
if (isset($arResult['FIELDS']
['CONTACT_ID']))//deprecated use
crm.deal.contact.items.get
{
        unset($arResult['FIELDS']
['CONTACT_ID']);
}
if (isset($arResult['FIELDS']
['CONTACT_IDS']))// use
crm.deal.contact.items.get
{
        unset($arResult['FIELDS']
['CONTACT_IDS']);
}

foreach ($arResult['FIELDS'] as $key =>
$arField)
{
        $value = '';
        $return = '';
        if (!empty($arResult['ITEM'][$key]))
        {
                $value = $arResult['ITEM']
[$key];
        }
        $arList =
(isset($arResult['FIELD_VALUES_' . $key])) ?
```

```
$arResult['FIELD_VALUES_' . $key] : [];
        switch ($arField['type'])
        {
                case 'crm_category':
                        if
(!empty($arResult['VALUE_' . $key]))
                        {
                                $arList =
array_column($arResult['VALUE_' . $key],
'NAME', 'ID');
                        }
                        $return =
CPrintForm::select(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value
                                ],
                                $arList
                        );

                        break;
                case 'crm_quote'://only for
QUOTE_ID read only type
                        if
(!empty($arResult['VALUE_QUOTE_ID']))
                        {
                                $return .=
implode(', ',
array_column($arResult['VALUE_QUOTE_ID'],
```

```
'TITLE'));
                        }
                        break;
                case 'location':

                        $return .= 'field
not support location';
                        break;
                case 'crm_status':
                        if (empty($arList))
                        {

$arFieldsStatus = \CRest::get(

'crm.status.list',

['filter' => ['ENTITY_ID' =>
$arField['statusType']]]
                                );
                                if
(!empty($arFieldsStatus['result']))
                                {

$arList = $arFieldsStatus['result'];
                                }
                        }
                        $arList =
array_column($arList, 'NAME', 'STATUS_ID');

                        $return =
CPrintForm::select(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],
```

```
'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value
                                ],
                                $arList
                        );
                        break;
                case 'crm_currency':
                        $arList =
array_column($arResult['FIELD_VALUES_CURRENC
Y'], 'FULL_NAME', 'CURRENCY');
                        $return =
CPrintForm::select(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value
                                ],
                                $arList
                        );
                        break;
                case 'enumeration':
                        if
(!empty($arField['items']))
                        {
                                $arList =
array_column($arField['items'], 'VALUE',
'ID');
```

```
                        }

                        $return =
CPrintForm::select(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value
                                ],
                                $arList
                        );
                        break;
                case 'crm_multifield'://its
simple example: need multifield, check data
type and more...
                        if (!empty($value)
&& is_array($value))
                        {
                                $value =
reset($value)['VALUE'];
                        }
                        $return =
CPrintForm::input(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],
```

```
'MULTIPLE' => false,

'VALUE' => $value,

'TYPE' => 'text',
                                ]
                        );
                        break;
                case 'crm_lead':
                        $return =
CPrintForm::input(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'text',
                                ]
                        );

                        if
(!empty($arResult['VALUE_LEAD_ID']) &&
$value == $arResult['VALUE_LEAD_ID']['ID'])
                        {
                                $return .=
'(' . $arResult['VALUE_LEAD_ID']['TITLE'] .
')';
                        }
                        break;
```

```
                case 'crm_company':
                        $arCompany = [];
                        if
(!empty($arResult['VALUE_' . $key]))
                        {
                                $arCompany =
$arResult['VALUE_' . $key];
                        }
                        elseif
(!empty($value))
                        {
                                $arCompany =
CRest::get('crm.company.list', ['filter' =>
['ID' => $value]]);
                        }
                        $return =
CPrintForm::input(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'text',
                                ]
                        );
                        if
(!empty($arCompany['result']))
                        {
                                $return .=
'(';
```

```
                                $i = 0;
                                foreach
($arCompany['result'] as $val)
                                {

$i++;
                                        if
($i > 1)
                                        {

$return .= ', ';
                                        }

$return .= $val['TITLE'];
                                }
                                $return .=
')';
                        }
                        break;
                case 'crm_contact':
                        $arContact = [];
                        if
(!empty($arResult['VALUE_' . $key]))
                        {
                                $arContact =
$arResult['VALUE_' . $key];
                        }
                        elseif
(!empty($value))
                        {
                                $arContact =
CRest::get('crm.contact.list', ['filter' =>
['ID' => $value]]);
                        }

                        $return =
CPrintForm::input(
```

```
                                [
'NAME' => 'form[' . $key . ']',
'REQUIRED' => $arField['isRequired'],
'DISABLE' => $arField['isReadOnly'],
'MULTIPLE' => $arField['isMultiple'],
'VALUE' => $value,
'TYPE' => 'text',
                                ]
                        );
                        if
(!empty($arContact['result']))
                        {
                                $return .=
'(';
                                $i = 0;
                                foreach
($arContact['result'] as $val)
                                {
$i++;
                                        if
($i > 1)
                                        {
$return .= ', ';
                                        }
$return .= implode(' ', [$val['NAME'],
$val['LAST_NAME']]);
                                }
                                $return .=
```

```
')';
                        }
                        break;
                case 'file':
                        $return =
CPrintForm::input(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'file',
                                ]
                        );
                        if
($arField['isMultiple'])
                        {
                                if
(is_array($value))
                                {

foreach ($value as $k => $val)
                                        {

if (!empty($val['downloadUrl']))

{

$return .= '<br/><a href="' .
$val['downloadUrl'] . '">old file ' . $k .
```

```
'</a>';

}
                                                }
                                        }
                                }
                                else
                                {
                                        if
(!empty($value['downloadUrl']))
                                        {

$return .= '<br/><a href="' .
$value['downloadUrl'] . '">old file</a>';
                                        }
                                }
                                break;
                        case 'date':
                                if (!empty($value))
                                {
                                        $value =
date('Y-m-d', strtotime($value));
                                }
                                $return =
CPrintForm::input(
                                        [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,
```

```
'TYPE' => 'date',
                                ]
                        );
                        break;
                case 'datetime':
                        if (!empty($value))
                        {
                                $value =
date('Y-m-d\TH:i:s', strtotime($value));
                        }
                        $return =
CPrintForm::input(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'datetime-local',
                                ]
                        );
                        break;
                case 'char':
                        $return =
CPrintForm::input(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],
```

```
'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => 'Y',

'CHECKED' => ($value == 'Y') ? true : false,

'TYPE' => 'checkbox',
                                ]
                        );
                        break;

                case 'boolean':
                        $return =
CPrintForm::input(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => '1',

'CHECKED' => ($value == 'Y') ? true : false,

'TYPE' => 'checkbox',
                                ]
                        );
                        break;
                case 'double':
                        $return =
CPrintForm::input(
```

```
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'number'
                                ]
                        );
                        break;
                case 'user':
                        $arUser = [];
                        if (!empty($value))
                        {
                                $arUser =
CRest::get('user.get', ['filter' => ['ID' =>
$value]]);
                        }
                        $return =
CPrintForm::input(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,
```

```
'TYPE' => 'number'
                                ]
                        );
                        if
(!empty($arUser['result']))
                        {
                                $return .=
'(';
                                $i = 0;
                                foreach
($arUser['result'] as $val)
                                {

$i++;
                                        if
($i > 1)
                                        {

$return .= ', ';
                                        }

$return .= implode(' ', [$val['NAME'],
$val['LAST_NAME']]);
                                }
                                $return .=
')';
                        }

                        break;
                case 'url':
                        $return =
CPrintForm::input(
                                [

'NAME' => 'form[' . $key . ']',
```

```
'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'text',
                                ]
                        );
                        break;
                case 'integer':
                        $return =
CPrintForm::input(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'number',
                                ]
                        );
                        break;
                case 'money':
                        list($money,
$currency) = explode('|', $value);
                        $return =
CPrintForm::input(
                                [
```

```
'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $money,

'TYPE' => 'number',
                                ]
                        );
                        $arList =
array_column($arResult['FIELD_VALUES_CURRENC
Y'], 'FULL_NAME', 'CURRENCY');
                        $return .=
CPrintForm::select(
                                [

'NAME' => $key . '_CURRENCY',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $currency
                                ],
                                $arList
                        );
                        break;
                case 'address':
                        $return =
CPrintForm::input(
```

```
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,

'TYPE' => 'text',
                                ]
                        );
                        break;
                case 'resourcebooking':
                        //some code booking
                        $return = 'field not
support';
                        break;
                default:

                        $return =
CPrintForm::input(
                                [

'NAME' => 'form[' . $key . ']',

'REQUIRED' => $arField['isRequired'],

'DISABLE' => $arField['isReadOnly'],

'MULTIPLE' => $arField['isMultiple'],

'VALUE' => $value,
```

```
'TYPE' => 'text',
                                ]
                        );

                        break;
        }

        if (strpos($key, 'UF_') === 0)
        {
                $sResultCustom .= '<div
class="col-4 mt-3">' .

(($arField['formLabel']) ?
$arField['formLabel'] : $arField['title']) .
                        ': ' .
                        '</div>';
                $sResultCustom .= '<div
class="col-6 mt-3">' . $return . '</div>';
        }
        else
        {
                $sResult .= '<div
class="col-4 mt-3">' .

(($arField['formLabel']) ?
$arField['formLabel'] : $arField['title']) .
                        ': ' .
                        '</div>';
                $sResult .= '<div
class="col-6 mt-3">' . $return . '</div>';
        }
}

?>
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/boo
tstrap/4.3.1/css/bootstrap.min.css"
```

```
          crossorigin="anonymous">
<script src="https://code.jquery.com/jquery-
3.3.1.slim.min.js" crossorigin="anonymous">
</script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/
popper.js/1.14.7/umd/popper.min.js"
                crossorigin="anonymous">
</script>
<script
src="https://stackpath.bootstrapcdn.com/boot
strap/4.3.1/js/bootstrap.min.js"
                crossorigin="anonymous">
</script>
<script
src="https://ajax.googleapis.com/ajax/libs/j
query/3.3.1/jquery.min.js"></script>
<script>
          $(document).ready(function () {
                $('#auto_form').on('submit',
function (el) {//event submit form

el.preventDefault();//the default action of
the event will not be triggered
                        var formData = new
FormData(this);
                        var xhr = new
XMLHttpRequest();
                        xhr.open("POST",
'auto_form.php');

xhr.onreadystatechange = function () {
                                if
(this.readyState === 4)
                                {
                                        if
(this.status >= 200 && this.status < 400)
```

```
                                                {

// Success!

var resp = this.responseText;

try

{

var json = JSON.parse(resp);

if (typeof json.message !== 'undefined')

{

alert(json.message);

}

} catch (e)

{

return false;

}
                                                }
                                                else
                                                {

alert('error');
                                                }
                                        }
                                };
                                xhr.send(formData);
                        });
```

```
        });
</script>
<div class="container">
        <form id="auto_form" action=""
enctype="multipart/form-data" method="post">
                <?if
(!empty($arResult['ITEM']['ID'])):// for
update entity?>
                                <input
type="hidden" name="form[ID]" value="<?
=$arResult[ 'ITEM' ][ 'ID' ]?>">
                        <?endif;?>
                <h2>Standard fields</h2>
                <div class="row">
                        <?=$sResult?>
                </div>
                <h2>Custom fields</h2>
                <div class="row">
                        <?=$sResultCustom?>
                </div>
                <div class="row">
                        <div class="col-sm-
10 mt-5">
                                <input
type="submit" class="btn btn-primary"
value="Submit">
                        </div>
                </div>
        </form>
</div>
<?endif;?>
```

Файл **auto_form.php**:

```
<?
$arForm = [];
```

```
foreach ($_POST['form'] as $key => $item)
{
        if (is_array($item))
        {
                $arForm[$key] = [];
                foreach ($item as $k =>
$val)
                {
                        $arForm[$key][$k] =
htmlspecialchars($val);
                }
        }
        else
        {
                $arForm[$key] =
htmlspecialchars($item);
        }
}
//make array multiple files for add to
custom field
if (!empty($_FILES['form']['tmp_name']) &&
is_array($_FILES['form']['tmp_name']))
{
        foreach ($_FILES['form']['tmp_name']
as $key => $files)
        {
                if (is_array($files))
                {
                        foreach ($files as
$k => $file)
                        {

$arForm[$key][$k] = [

"fileData" => [

$_FILES['form']['name'][$key][$k],
```

```
base64_encode(file_get_contents($file))
                                        ]
                                ];
                        }
                }
                else
                {
                        $arForm[$key] = [
                                "fileData"
=> [

$_FILES['form']['name'][$key],

base64_encode(file_get_contents($files))
                                ]
                        ];
                }
        }
}
$arResult = CRest::get('crm.deal.fields',
[]);
if (!empty($arResult['result']))
{
        foreach ($arResult['result'] as $key
=> $prop)
        {
                if (!isset($arForm[$key]))
                {
                        if
(!$prop['isReadOnly'] && $prop['type'] !=
'file')
                        {
                                if
($prop['type'] == 'enumeration' &&
$prop['isMultiple'])
                                {
```

```
                                                //if
type multiple enumeration to clean selected
value need send: [false]

$arForm[$key] = [false];
                                }
                                elseif
($prop['isMultiple'])
                                {

$arForm[$key] = [];
                                }
                                else
                                {

$arForm[$key] = '';
                                }
                        }
                        continue;
                }
                //here may be any check
field example by type
                if ($prop['type'] ==
'crm_multifield')
                {
                        if
(isset($arForm[$key]))
                        {

$arForm[$key] = [['VALUE' =>
$arForm[$key]]];
                        }
                }
                elseif ($prop['type'] ==
'money')
                {
                        $arForm[$key] =
```

```
implode('|', [$arForm[$key], $arForm[$key .
'_CURRENCY']]);
                        unset($arForm[$key .
'_CURRENCY']);
                }
        }
}
$arForm['ID'] = intVal($arForm['ID']);
if ($arForm['ID'] > 0)
{
        $method = 'crm.deal.update';
        $arParams = [
                'id' => $arForm['ID'],
                'fields' => $arForm
        ];
        $arMess = [
                'success' => 'Deal update',
                'error' => 'Deal not
updated',
        ];
}
else
{
        $method = 'crm.deal.add';
        $arParams = [
                'fields' => $arForm
        ];
        $arMess = [
                'success' => 'Deal add',
                'error' => 'Deal not added',
        ];
}
$result = CRest::get($method, $arParams);
if (!empty($result['result']))
{
        echo json_encode(
                ['message' =>
```

```
$arMess['success'] . (($method ==
'crm.deal.add') ? ' ID:' . $result['result']
: '')]
        );
}
elseif
(!empty($result['error_description']))
{
        echo json_encode(['message' =>
$arMess['error'] . ': ' .
$result['error_description']]);
}
else
{
        echo json_encode(['message' =>
$arMess['error']]);
}

?>
```

Как изменить номера телефонов и e-mail на примере контакта

Описание

Примеры добавления/изменения/удаления телефона и e-mail на примере контакта.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

Работа с E-Mail

```
$sEmail1 = rand(111111111, 999999999) .
'@nomail.com';
$sEmail2 = rand(111111111, 999999999) .
'@nomail.com';

$arNewEmail = [
        [
                'VALUE' => $sEmail1,
                'VALUE_TYPE' => 'HOME'
        ],
        [
                'VALUE' => $sEmail2,
                'VALUE_TYPE' => 'HOME'
```

```
        ]
];

//create contact with phone
$newContact = CRest::call(
        'crm.contact.add',
        [
                'fields' => [
                        'NAME' => 'CHANGE
EMAIL',
                        'EMAIL' =>
$arNewEmail

                ]
        ]
);
if ($newContact['result'] > 0)
{
        //get contact with email
        $newContactData = CRest::call(
                'crm.contact.get',
                [
                        'id' =>
$newContact['result']
                ]
        );

        //change 1 email and delete 2 email
        if (!empty($newContactData['result']
['EMAIL'][0]) &&
!empty($newContactData['result']['EMAIL']
[1]))
        {
                $arUpdateEmail = [
                        [//change
                         'ID' =>
$newContactData['result']['EMAIL'][0]['ID'],
```

```
                            'VALUE' =>
rand(111111111, 999999999) . '@nomail.com'
                        ],
                        [//delete
                            'ID' =>
$newContactData['result']['EMAIL'][1]['ID'],
                            'VALUE' =>
''//empty value for delete email
                        ],
                ];
                $resultContactChange =
CRest::call(

'crm.contact.update',
                        [
                                'id' =>
$newContactData['result']['ID'],
                                'fields' =>
[

'EMAIL' => $arUpdateEmail
                                ]
                        ]
                );
        }
}
else
{
        echo 'error creat contact ' .
$newContact['error_description'];
}
```

Работа с телефонами

```
$sPhone1 = rand(111111111, 999999999);
$sPhone2 = rand(111111111, 999999999);

$arNewPhone = [
        [
                'VALUE' => $sPhone1,
                'VALUE_TYPE' => 'HOME'
        ],
        [
                'VALUE' => $sPhone2,
                'VALUE_TYPE' => 'HOME'
        ]
];

//creat contact with phone
$newContact = CRest::call(
        'crm.contact.add',
        [
                'fields' => [
                        'NAME' => 'CHANGE
PHONE',
                        'PHONE' =>
$arNewPhone

                ]
        ]
);
if ($newContact['result'] > 0)
{
        //get contact with phone
        $newContactData = CRest::call(
                'crm.contact.get',
                [
                        'id' =>
$newContact['result']
                ]
        );
```

```
        //change 1 phone and delete 2 phone
        if (!empty($newContactData['result']
['PHONE'][0]) &&
!empty($newContactData['result']['PHONE']
[1]))
        {
                $arUpdatePhone = [
                        [//change
                         'ID' =>
$newContactData['result']['PHONE'][0]['ID'],
                         'VALUE' =>
rand(111111111, 999999999)
                        ],
                        [//delete
                         'ID' =>
$newContactData['result']['PHONE'][1]['ID'],
                         'VALUE' =>
''//empty value for delete phone
                        ],
                ];
                $resultContactChange =
CRest::call(

'crm.contact.update',
                        [
                                'id' =>
$newContactData['result']['ID'],
                                'fields' =>
[

'PHONE' => $arUpdatePhone
                                ]
                        ]
                );
        }
}
```

```
else
{
        echo 'error creat contact ' .
$newContact['error_description'];
}
```

Как изменить даты в деле-событии

Пример изменения времени запланированного дела на завтра начало в это время, завершение +2 часа.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

```
$activityID = 42;
$timeStart = time() + 86400;//tomorrow
$timeEnd = time() + 86400 + 7200;//tomorrow
plus 2 hours

CRest::call(
        'crm.activity.update',
        [
                'id' => $activityID,
                'fields' => [
                        "START_TIME" =>
date("Y-m-d H:i:s", $timeStart),
                        "END_TIME" =>
date("Y-m-d H:i:s", $timeEnd),
                ]
        ]
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Как изменить значения пользовательских полей товара

Описание

Примеры работы с различными свойствами товара.

Для работы примеров необходимо создать папку **/pictures** рядом с исполняемым файлом примера и заполнить её картинками с названиями "1.jpg" - "6.jpg". Также в начале примера необходимо исправить значения из примера на ваши:

- **$propertyIDSelect** - ID не множественного списочного свойства.
- **$propertySelectValueID** - ID значения не множественного списочного свойства.
- **$propertyIDMultiSelect** - ID множественного списочного свойства.
- **$propertyMultiSelectValueID** - ID значений множественного списочного свойства.
- **$propertyIDFile** - ID не множественного свойства типа файл.
- **$propertyIDMultiFile** - ID множественного свойства типа файл.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

Изменение товара

```
$idProduct = 10339;

$propertyIDSelect = 106;
$propertySelectValueID = 85;

$propertyIDMultiSelect = 105;
$propertyMultiSelectValueID = [79, 80, 82];

$propertyIDFile = 107;
$propertyFilePathToPicture =
'pictures/1.jpg';//relative or full path on
server

$propertyIDMultiFile = 108;
$propertyMultiFilePathToPicture =
[//relative or full path on server

'pictures/2.jpg',

'pictures/3.jpg',

'pictures/4.jpg',
];

$standardPreviewPicturePath =
'pictures/5.jpg';//relative or full path on
server
$standardDetailPicturePath =
'pictures/6.jpg';//relative or full path on
server

$arFields = [
        'NAME' => 'Example product 2',
        'CURRENCY_ID' => 'USD',
        'PRICE' => 4900,
        'SORT' => 500
];
```

```
$result = CRest::call(
        'crm.product.get',
        [
                'id' => $idProduct
        ]
);
if (!empty($result['result']))
{
        $arProduct = $result['result'];
        if ($propertyIDSelect > 0 &&
$propertySelectValueID > 0)
        {
                $arFields['PROPERTY_' .
$propertyIDSelect] = $propertySelectValueID;
        }

        if ($propertyIDMultiSelect > 0 &&
is_array($propertyMultiSelectValueID) &&
count($propertyMultiSelectValueID) > 0)
        {
                $arFields['PROPERTY_' .
$propertyIDMultiSelect] =
$propertyMultiSelectValueID;
        }

        if ($propertyIDFile > 0 &&
!empty($propertyFilePathToPicture) &&
file_exists($propertyFilePathToPicture))
        {
                $fileName = end(explode('/',
$propertyFilePathToPicture));
                $arFields['PROPERTY_' .
$propertyIDFile] = [
                        "fileData" => [
                                $fileName,
```

```
base64_encode(file_get_contents($propertyFil
ePathToPicture))
                        ]
                ];
        }
        if ($propertyIDMultiFile > 0 &&

is_array($propertyMultiFilePathToPicture) &&

count($propertyMultiFilePathToPicture) > 0)
        {
                foreach
($propertyMultiFilePathToPicture as $path)
                {
                        if
(file_exists($path))
                        {
                                $fileName =
end(explode('/', $path));

$arFields['PROPERTY_' .
$propertyIDMultiFile][] = [

"fileData" => [

$fileName,

base64_encode(file_get_contents($path))
                                        ]
                                ];
                        }
                }
        }
        if
(!empty($standardPreviewPicturePath) &&
file_exists($standardPreviewPicturePath))
        {
```

```
                $fileName = end(explode('/',
$standardPreviewPicturePath));
                $arFields['PREVIEW_PICTURE']
= [
                        "fileData" => [
                                $fileName,

base64_encode(file_get_contents($standardPre
viewPicturePath))
                        ]
                ];
        }
        if
(!empty($standardDetailPicturePath) &&
file_exists($standardDetailPicturePath))
        {
                $fileName = end(explode('/',
$standardDetailPicturePath));
                $arFields['DETAIL_PICTURE']
= [
                        "fileData" => [
                                $fileName,

base64_encode(file_get_contents($standardDet
ailPicturePath))
                        ]
                ];
        }

        //delete old files
        $arPropsFile = [
                'PREVIEW_PICTURE',
                'DETAIL_PICTURE',
        ];
        if ($propertyIDFile > 0)
        {
                $arPropsFile[] = 'PROPERTY_'
```

```
. $propertyIDFile;
        }
        if ($propertyIDMultiFile > 0)
        {
                $arPropsFile[] = 'PROPERTY_'
. $propertyIDMultiFile;
        }
        foreach ($arPropsFile as $prop)
        {
                if
(empty($arFields[$prop]))//if property does
not change file dont delete old file
                {
                        continue;
                }

                if (!empty($arProduct[$prop]
['id']))//for standard fields
PREVIEW_PICTURE and DETAIL_PICTURE
                {
                        $arFields[$prop][] =
[
                                'id' =>
$arProduct[$prop]['id'],
                                'remove' =>
'Y',
                        ];
                }
                elseif
(!empty($arProduct[$prop]['value']
['id']))//for property type file
                {
                        $arFields[$prop][] =
[
                                'valueId' =>
$arProduct[$prop]['valueId'],
                                'value' => [
```

```
                                                'id'
=> $arProduct[$prop]['value']['id'],

'remove' => 'Y',
                                ]
                        ];
                }
                elseif
(!isset($arProduct[$prop]['value']) &&
is_array($arProduct[$prop]))//for property
type multiple file
                {
                        foreach
($arProduct[$prop] as $file)
                        {
                                if
(!empty($file['value']['id']))
                                {

$arFields[$prop][] = [

'valueId' => $file['valueId'],

'value' => [

'id' => $file['value']['id'],

'remove' => 'Y',

]
                                                ];
                                }

                        }
                }
        }
```

```
        $result = CRest::call(
                'crm.product.update',
                [
                        'id' => $idProduct,
                        'fields' =>
$arFields
                ]
        );

}
```

Очистка свойств с файлами в товаре

```
$idProduct = 10339;

$propertyIDFile = 107;
$propertyIDMultiFile = 108;

$result = CRest::call(
        'crm.product.get',
        [
                'id' => $idProduct
        ]
);

if (!empty($result['result']))
{
        $arProduct = $result['result'];
        $arPropsFile = [
                'PREVIEW_PICTURE',
                'DETAIL_PICTURE',
        ];
        if ($propertyIDFile > 0)
        {
                $arPropsFile[] = 'PROPERTY_'
```

```
. $propertyIDFile;
        }
        if ($propertyIDMultiFile > 0)
        {
                $arPropsFile[] = 'PROPERTY_'
. $propertyIDMultiFile;
        }
        $arSaveData = [];
        foreach ($arPropsFile as $prop)
        {
                if (!empty($arProduct[$prop]
['id']))//for standard fields
PREVIEW_PICTURE and DETAIL_PICTURE
                {
                        $arSaveData[$prop] =
[
                                'id' =>
$arProduct[$prop]['id'],
                                'remove' =>
'Y',
                        ];
                }
                elseif
(!empty($arProduct[$prop]['value']
['id']))//for property type file
                {
                        $arSaveData[$prop] =
[
                                'valueId' =>
$arProduct[$prop]['valueId'],
                                'value' => [
                                        'id'
=> $arProduct[$prop]['value']['id'],

'remove' => 'Y',
                                ]
                        ];
```

```
                }
                elseif
(!isset($arProduct[$prop]['value']) &&
is_array($arProduct[$prop]))//for property
type multiple file
                {
                        foreach
($arProduct[$prop] as $file)
                        {
                                if
(!empty($file['value']['id']))
                                {

$arSaveData[$prop][] = [

'valueId' => $file['valueId'],

'value' => [

'id' => $file['value']['id'],

'remove' => 'Y',

]
                                                ];
                                }

                        }
                }
        }
}

$resultSave = CRest::call(
        'crm.product.update',
        [
                'id' => $idProduct,
                'fields' => $arSaveData
```

```
            ]
);
```

Поиск в CRM по телефону и e-mail

Пример отображает форму для ввода телефона и e-mail. Ниже формы выводится таблица с результатом поиска по столбцам:

- id сущности
- сущность
- заголовок
- телефоны сущности
- емаилы сущности

Код через поиск дубликатов находит все сущности (лид, контакт, компания) у которых есть указанный телефон/e-маил. Потом из списка всех ID получается информация о каждой сущности:

- заголовок или имя фамилия
- все телефоны
- e-mail

и выводится в таблице.

```
<?
   include('crest.php');
   $phone = ($_POST['PHONE'])?
htmlspecialchars($_POST['PHONE']):false;
   $email = ($_POST['EMAIL'])?
htmlspecialchars($_POST['EMAIL']):false;

   $entityIDs = [
      'LEAD' => [],
      'CONTACT' => [],
```

```
        'COMPANY' => []
    ];
    $resultEntity = [
        'lead' => [],
        'contact' => [],
        'company' => []
    ];

    if($phone)
    {
        $result =
CRest::call('crm.duplicate.findbycomm', [
            'type' => 'PHONE',
            'values' => [$phone]
        ]);
        if(is_array($result['result']
['LEAD']))
        {
            $entityIDs['LEAD'] =
array_merge($entityIDs['LEAD'],
$result['result']['LEAD']);
        }
        if(is_array($result['result']
['CONTACT']))
        {
            $entityIDs['CONTACT'] =
array_merge($entityIDs['CONTACT'],
$result['result']['CONTACT']);
        }
        if(is_array($result['result']
['COMPANY']))
        {
            $entityIDs['COMPANY'] =
array_merge($entityIDs['COMPANY'],
$result['result']['COMPANY']);
        }
    }
```

```
   if($email)
   {
      $result =
CRest::call('crm.duplicate.findbycomm', [
         'type' => 'EMAIL',
         'values' => [$email]
      ]);
      if(is_array($result['result']
['LEAD']))
      {
         $entityIDs['LEAD'] =
array_merge($entityIDs['LEAD'],
$result['result']['LEAD']);
      }
      if(is_array($result['result']
['CONTACT']))
      {
         $entityIDs['CONTACT'] =
array_merge($entityIDs['CONTACT'],
$result['result']['CONTACT']);
      }
      if(is_array($result['result']
['COMPANY']))
      {
         $entityIDs['COMPANY'] =
array_merge($entityIDs['COMPANY'],
$result['result']['COMPANY']);
      }
   }

   if(!empty($entityIDs['LEAD']))
   {
      $result = CRest::call(
         'crm.lead.list',
         [
            'filter' => [
               'ID' => $entityIDs['LEAD']
```

```
            ],
            'select' =>    [
               'ID', 'NAME', 'LAST_NAME',
'PHONE', 'EMAIL', 'TITLE'
            ]
         ]
      );
      if(!empty($result['result']))
      {
         $resultEntity['lead'] =
$result['result'];
      }
   }
   if(!empty($entityIDs['CONTACT']))
   {
      $result = CRest::call(
         'crm.contact.list',
         [
            'filter' => [
               'ID' => $entityIDs['CONTACT']
            ],
            'select' =>    [
               'ID', 'NAME', 'LAST_NAME',
'PHONE', 'EMAIL'
            ]
         ]
      );
      if(!empty($result['result']))
      {
         $resultEntity['contact'] =
$result['result'];
      }
   }
   if(!empty($entityIDs['COMPANY']))
   {
      $result = CRest::call(
         'crm.company.list',
```

```
         [
            'filter' => [
               'ID' => $entityIDs['COMPANY']
            ],
            'select' =>    [
               'ID', 'PHONE', 'EMAIL',
'TITLE'
            ]
         ]
      );
      if(!empty($result['result']))
      {
         $resultEntity['company'] =
$result['result'];
      }
   }
?>
<!DOCTYPE html>
<html lang="ru">
   <head>
      <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/boo
tstrap/4.3.1/css/bootstrap.min.css"
crossorigin="anonymous">
   </head>
   <body class="container">
      <form method="post" action="">
         <div class="row">
            <div class="col-4 mt-3">
               <label>E-mail*</label>
            </div>
            <div class="col-6 mt-3">
               <input type="text"
name="EMAIL" value="<?=$email?>">
            </div>
         </div>
         <div class="row">
```

```
            <div class="col-4 mt-3">
               <label>Phone*</label>
            </div>
            <div class="col-6 mt-3">
               <input type="text"
name="PHONE" value="<?=$phone?>">
            </div>
         </div>
         <div class="row">
            <div class="col-sm-10">
               <input type="submit"
name="SEARCH" class="btn btn-primary"
value="Search">
            </div>
         </div>
      </form>
      <table class="table mt-5">
         <thead>
            <tr>
               <th scope="col">ID</th>
               <th scope="col">Entity</th>
               <th scope="col">Title</th>
               <th scope="col">Phones</th>
               <th scope="col">Emails</th>
            </tr>
         </thead>
         <tbody>
            <? foreach($resultEntity as
$entity=>$items):?>
               <? foreach($items as $item):
                  $phones = '';
                  if(!empty($item['PHONE']))
                  {
                     $item['PHONE'] =
array_column($item['PHONE'],'VALUE');
                     $phones = implode(', ',
$item['PHONE']);
```

```
                    }

                    $emails = '';
                    if(!empty($item['EMAIL']))
                    {
                       $item['EMAIL'] =
array_column($item['EMAIL'],'VALUE');
                       $emails = implode(', ',
$item['EMAIL']);
                    }

                    $title = '';
                    if($item['TITLE'])
                    {
                       $title =
$item['TITLE'].(($item['NAME'] ||
$item['LAST_NAME'])?': ':'');
                    }
                    if($item['NAME'] ||
$item['LAST_NAME'])
                    {
                       $title .= implode(' ',
[$item['NAME'], $item['LAST_NAME']]);
                    }
                    ?>
                    <tr>
                       <th scope="row">
=$item['ID']?></th>
                       <td><?=$entity?></td>
                       <td><?=$title?></td>
                       <td><?=$phones?></td>
                       <td><?=$emails?></td>
                    </tr>
                 <? endforeach?>
              <? endforeach?>
           </tbody>
        </table>
```

```
    </body>
</html>
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Как получить список дел на примере контакта

Пример получает список грядущих дел контакта. Для получения дел других сущностей необходимо заменить поле `OWNER_TYPE_ID`, список возможных значений поля можно получить сделав запрос `CRest::call('crm.enum.ownertype');`.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

```
$contactID = 1;
$resultActivity = [];
$resultActivity = CRest::call(
        'crm.activity.list',
        [
                'filter' => [
                        'COMPLETED' =>
'N',//only new activity
                        'OWNER_ID' =>
$contactID,
                        'OWNER_TYPE_ID' =>
3, // CRest::call('crm.enum.ownertype');
                ],
                'select' => [
                        '*',
                        'COMMUNICATIONS'
                ]
        ]
);
```

```
echo '<pre>';
        print_r($resultActivity);
echo '</pre>';
```

Получение списка статусов лидов с семантикой

Пример получения всех статусов лида с семантикой.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

```
<?
$resultLeads =
CRest::call('crm.status.list', ['filter' =>
['ENTITY_ID' => 'STATUS']]);
if (!empty($resultLeads['result'])):?>
        <table>
                <thead>
                <tr>
                        <th>STATUS ID</th>
                        <th>NAME</th>
                        <th>SEMANTICS</th>
                </tr>
                </thead>
                <tbody>
                <? foreach
($resultLeads['result'] as $item): ?>
                <tr <?=
(!empty($item['EXTRA']['COLOR']) ? '
style="color:' . $item['EXTRA']['COLOR'] .
'"' : '');?>>
                        <td><?
```

```
=$item['STATUS_ID']?></td>
                        <td><?
=$item['NAME']?></td>
                        <td><?
=$item['EXTRA']['SEMANTICS']?></td>
                <tr>
                        <? endforeach;
                        ?>
                </tbody>
        </table>
<? endif; ?>
```

Получение списка статусов коммерческих предложений

Пример получения всех статусов коммерческих предложений с семантикой.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

```
<?
$resultQuote =
CRest::call('crm.status.list', ['filter' =>
['ENTITY_ID' => 'QUOTE_STATUS']]);
if (!empty($resultQuote['result'])):?>
        <table>
                <thead>
                <tr>
                        <th>STATUS ID</th>
                        <th>NAME</th>
                        <th>SEMANTICS</th>
                </tr>
                </thead>
                <tbody>
                <? foreach
($resultQuote['result'] as $item): ?>
                <tr <?=
(!empty($item['EXTRA']['COLOR']) ? '
```

```
style="color:' . $item['EXTRA']['COLOR'] .
'"' : '');?>>
                        <td><?
=$item['STATUS_ID']?></td>
                        <td><?
=$item['NAME']?></td>
                        <td><?
=$item['EXTRA']['SEMANTICS']?></td>
                <tr>
                        <? endforeach;
                        ?>
                </tbody>
        </table>
<? endif; ?>
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Использование сквозной аналитики при создании лида

Пример использования сквозной аналитики при создании лида. Предварительно создайте php-страницу с веб-формой обратной связи (Ф.И.О, телефон). На странице разместите код примера.

Что происходит во время выполнения кода?

1. Подключается стандартный js-код из сквозной аналитики Битрикс24.
2. После заполнения формы помимо полей формы в скрытом поле передаётся код для сквозной аналитики `b24Tracker.guest.getTrace()`.
3. Затем вызывается crm.lead.add, в котором в поле **TRACE** добавляется код из getTrace.

Скрипт сквозной аналитики устанавливается на вашем сайте перед закрывающим тегом `</body>` на всех страницах сайта включая страницу с формой.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

```
<!DOCTYPE html>
<html lang="ru">
   <head>
      <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/boo
tstrap/4.3.1/css/bootstrap.min.css"
```

```
crossorigin="anonymous">
   </head>
   <body class="container">
      <h1>Feedback</h1>
      <?
      include("crest.php");
      $message = '';
      if(!empty($_POST['SAVE']))
      {
         $fields = [
            'TRACE' => $_POST['TRACE'],
            'NAME' => $_POST['NAME'],
            'LAST_NAME' =>
$_POST['LAST_NAME'],
            'PHONE' => [ [
'value'=>$_POST['PHONE'] ] ],
         ];
         $result = CRest::call(
            'crm.lead.add',
            [
               'fields' => $fields
            ]
         );
         if (!empty($result['result']))
         {
            $message = 'Feedback saved';
         }
         elseif
(!empty($result['error_description']))
         {
            $message =  'Feedback has not
been saved: '.$result['error_description'];
         }
         else
         {
            $message = 'Feedback has not
been saved';
```

```
            }
        }
        ?>
        <div class="col-12">
            <p><?=$message?></p>
        </div>
        <form method="post" action="">
            <input type="hidden"
id="FORM_TRACE" name="TRACE">
            <div class="row">
                <div class="col-4 mt-3">
                    <label>Name*</label>
                </div>
                <div class="col-6 mt-3">
                    <input type="text"
name="NAME" required>
                </div>
            </div>
            <div class="row">
                <div class="col-4 mt-3">
                    <label>Last name*</label>
                </div>
                <div class="col-6 mt-3">
                    <input type="text"
name="LAST_NAME" required>
                </div>
            </div>
            <div class="row">
                <div class="col-4 mt-3">
                    <label>Phone*</label>
                </div>
                <div class="col-6 mt-3">
                    <input type="text"
name="PHONE" required>
                </div>
            </div>
            <div class="row">
```

```
            <div class="col-sm-10">
               <input type="submit"
name="SAVE" class="btn btn-primary"
value="Send">
            </div>
         </div>
      </form>
      <script>
         window.onload = function(e){
            var traceInput =
document.getElementById('FORM_TRACE');
            if(traceInput)
            {
               traceInput.value =
b24Tracker.guest.getTrace();
            }
         }
      </script>
   </body>
</html>
```

Использование сквозной аналитики при создании сделки и контакта

Пример использования сквозной аналитики при создании сделки и контакта. Предварительно создайте php-страницу с веб-формой обратной связи (Ф.И.О, телефон). На странице разместите код примера.

Что происходит во время выполнения кода?

1. Подключается стандартный js-код из сквозной аналитики Битрикс24.
2. После заполнения формы помимо полей формы в скрытом поле передаётся код для сквозной аналитики `b24Tracker.guest.getTrace()`.
3. Далее создаётся сделка и связанный контакт.
4. И затем регистрируется "след" аналитики для этих объектов, передачей их типов и идентификаторов вида: `/rest/crm.tracking.trace.add?ENTITIES[0][TYPE]=CONTACT&ENTITIES[0][ID]=3215&ENTITIES[1][TYPE]=LEAD&ENTITIES[1][ID]=1&TRACE=….`

Скрипт сквозной аналитики устанавливается на вашем сайте перед закрывающим тегом `</body>` на всех страницах сайта включая страницу с формой.

Внимание! Для использования примера настройте работу класса *CRest* и подключите файл **crest.php** в файлах, где используется данный класс. Подробнее.

```
<!DOCTYPE html>
<html lang="ru">
```

```
<head>
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/boo
tstrap/4.3.1/css/bootstrap.min.css"
crossorigin="anonymous">
</head>
<body class="container">
<h1>Feedback</h1>
<?
include("crest.php");
$message = '';
if(!empty($_POST['SAVE']))
{

   $fields = [
      'NAME' => $_POST['NAME'],
      'LAST_NAME' => $_POST['LAST_NAME'],
      'PHONE' => [ [
'value'=>$_POST['PHONE'] ] ],
   ];

   $resultContact = CRest::call(
      'crm.contact.add',
      [
         'fields' => $fields
      ]
   );
   if (!empty($resultContact['result']))
   {

      $arDealFields = [
         'TITLE' => 'Feedback page:
'.$_POST['NAME'].' '.$_POST['LAST_NAME'],
         'CONTACT_ID' =>
$resultContact['result']
      ];
```

```
        $resultDeal = CRest::call(
           'crm.deal.add',
           [
              'fields' => $arDealFields
           ]
        );

        if (!empty($resultDeal['result']))
        {
           if(!empty($_POST['TRACE']))
           {

              $resultTrace = CRest::call(
                 'crm.tracking.trace.add',
                 [
                    'ENTITIES' => [
                       [
                          'TYPE' =>
'CONTACT',//COMPANY, CONTACT, DEAL, LEAD,
QUOTE
                          'ID' =>
$resultContact['result']
                       ],
                       [
                          'TYPE' =>
'DEAL',//COMPANY, CONTACT, DEAL, LEAD, QUOTE
                          'ID' =>
$resultDeal['result']
                       ]
                    ],
                    'TRACE'
=>  $_POST['TRACE']
                 ]
              );

           }
```

```
         $message = 'Feedback saved';
      }
      elseif
(!empty($resultDeal['error_description']))
      {
         $message =  'Feedback has not been
saved: '.$resultDeal['error_description'];
      }
      else
      {
         $message = 'Feedback has not been
saved';
      }
   }
   elseif
(!empty($resultContact['error_description'])
)
   {
      $message =  'Feedback has not been
saved: '.
$resultContact['error_description'];
   }
   else
   {
      $message = 'Feedback has not been
saved';
   }
}
?>
<div class="col-12">
   <p><?=$message?></p>
</div>
<form method="post" action="">
   <input type="hidden" id="FORM_TRACE"
name="TRACE">
   <div class="row">
      <div class="col-4 mt-3">
```

```
         <label>Name*</label>
      </div>
      <div class="col-6 mt-3">
         <input type="text" name="NAME"
required>
      </div>
   </div>
   <div class="row">
      <div class="col-4 mt-3">
         <label>Last name*</label>
      </div>
      <div class="col-6 mt-3">
         <input type="text" name="LAST_NAME"
required>
      </div>
   </div>
   <div class="row">
      <div class="col-4 mt-3">
         <label>Phone*</label>
      </div>
      <div class="col-6 mt-3">
         <input type="text" name="PHONE"
required>
      </div>
   </div>
   <div class="row">
      <div class="col-sm-10">
         <input type="submit" name="SAVE"
class="btn btn-primary" value="Send">
      </div>
   </div>
</form>
<script>
   window.onload = function(e){
      var traceDom =
document.getElementById('FORM_TRACE');
      if(traceDom)
```

```
        {
            var trace =
b24Tracker.guest.getTrace();
            traceDom.value = trace;
        }
    }
</script>
</body>
</html>
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Как передать информацию в Сквозную аналитику

При создании сущностей через REST есть 3 способа передать информацию для сквозной аналитики.

Самый простой

Передать в полях создаваемой сущности поле `UTM_SOURCE`.

В этом случае при создании сущности, если будет найден настроенный источник в сквозной аналитике с таким же **utm_source**, сущности будет проставлен этот источник, будет выведена соответствующая иконка, сущность будет участвовать в отчете.

Полные данные

Передать в полях создаваемой сущности поле `TRACE`.

В этом случае будут учтены все данные - устройство, все каналы (в том числе и сайт), посещенные страницы.

Способ работает для методов: crm.lead.add, crm.deal.add, crm.contact.add, crm.company.add, crm.quote.add

```
{
   "fields": {
      "NAME": "test",
      "LAST_NAME": "",
```

```
        "TRACE": ...
    },
  }
```

Значение **M** поля `TRACE` должен быть или идентификатор сохраненной записи сквозной аналитики или JSON-строка с массивом определенного формата, для получения которого можно просто воспользоваться JS-кодом виджета сквозной аналитики Битрикс24:

```
//js
b24Tracker.guest.getTrace()
```

Значение поля `TRACE` может быть число - ID трейса, который получен rest-методом crm.tracking.trace.add.

Создание трейса и получение его ID

Метод создает трейс:

```
crm.tracking.trace.add
?ENTITIES[0][TYPE]=CONTACT&ENTITIES[0]
[ID]=3215&ENTITIES[1][TYPE]=LEAD&ENTITIES[1]
[ID]=1&TRACE=
```

Поле `TRACE` обязательное, значение - Строка, полученная методом b24Tracker.guest.getTrace, пример выше.

Поле ENTITIES не обязательное, в нем можно перечислить сущности, которые связываются с этим трейсом:

```
ENTITIES: [{TYPE: 'CONTACT', ID: 1}, {TYPE: 'LEAD', ID:
101}]
```

Один трейс для связанных сущностей

Если создаётся пакет связанных сущностей (сделка + контакт + компания), то можно создать единый трейс для них. Если контакт и компания существующие, а создается только сделка, то можно создать трейс и привязать к существующим сущностям.

Смотри так же

- Использование сквозной аналитики при создании лида
- Использование сквозной аналитики при создании сделки и контакта

crm.lead.add

```
crm.lead.add(fields, params)
```

Создаёт новый лид.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей лида. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.lead.fields и посмотрите формат пришедших значений этих полей. Для создания повторного лида установите значения полей `COMPANY_ID` или `CONTACT_ID`. После установки этих полей флаг IS_RETURN_CUSTOMER будет установлен автоматически.
params	Набор параметров. REGISTER_SONET_EVENT - произвести регистрацию события добавления лида в живой ленте. Дополнительно будет отправлено уведомление ответственному за лид.

Пример

```
        BX24.callMethod(
        "crm.lead.add",
        {
                fields:
                {
                        "TITLE": "ИП Титов",
                        "NAME": "Глеб",
                        "SECOND_NAME":
"Егорович",
                        "LAST_NAME":
"Титов",
                        "STATUS_ID": "NEW",
                        "OPENED": "Y",
                        "ASSIGNED_BY_ID": 1,
                        "CURRENCY_ID":
"USD",
                        "OPPORTUNITY":
12500,
                        "PHONE": [ {
"VALUE": "555888", "VALUE_TYPE": "WORK" } ]
                },
                params: {
"REGISTER_SONET_EVENT": "Y" }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                        console.info("Создан
лид с ID " + result.data());
        }
);
```

crm.lead.delete

```
crm.lead.delete(id)
```

Удаляет лид и все связанные с ним объекты.

Параметры

Параметр	Описание
id	Идентификатор лида.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(
                        "crm.lead.delete",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
```

```
console.info(result.data());
                            }
                    );
```

crm.lead.fields

```
crm.lead.fields()
```

Возвращает описание полей лида, в том числе пользовательских.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(
                        "crm.lead.fields",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else

console.dir(result.data());
                        }
                );
```

Поля

Поле	Описание	Тип	П
ADDRESS	Адрес контакта	string	
ADDRESS_2	Вторая страница адреса	string	В нек принят на 2 ч
ADDRESS_CITY	Город	string	
ADDRESS_COUNTRY	Страна	string	
ADDRESS_COUNTRY_CODE	Код страны	string	
ADDRESS_POSTAL_CODE	Почтовый индекс	string	
ADDRESS_PROVINCE	Область	string	
ADDRESS_REGION	Район	string	
ASSIGNED_BY_ID	Связано с пользователем по ID	user	
BIRTHDATE	Дата рождения	date	
COMMENTS	Комментарии	string	
COMPANY_ID	Привязка лида к компании	crm_company	
COMPANY_TITLE	Название компании, привязанной к лиду	crm_company	

CONTACT_ID	Привязка лида к контакту	crm_contact	
CREATED_BY_ID	Кем создана	user	Тольк
CURRENCY_ID	Идентификатор валюты	crm_currency	
DATE_CLOSED	Дата закрытия	datetime	Тольк
DATE_CREATE	Дата создания	datetime	Тольк
DATE_MODIFY	Дата изменения	datetime	Тольк
EMAIL	Адрес электронной почты	crm_multifield	Множ
HAS_EMAIL	Проверка заполненности поля электронной почты	char	Тольк
HAS_PHONE	Проверка заполненности поля телефон	char	Тольк
HONORIFIC	Вид обращения	crm_status	
ID	Идентификатор контакта	integer	Тольк
IM	Мессенджеры	crm_multifield	Множ
IS_RETURN_CUSTOMER	Признак повторного лида	char	Тольк
LAST_NAME	Фамилия	string	Обяза
MODIFY_BY_ID	Идентификатор автора	user	Тольк

	последнего изменения		
MOVED_BY_ID	Идентификатор автора перемещения элемента на текущую стадию	user	Тольк
MOVED_TIME	Дата перемещения элемента на текущую стадию	user	Тольк
NAME	Имя	string	Обяза
OPENED	Доступен для всех	char	
OPPORTUNITY	Предполагаемая сумма	double	
ORIGINATOR_ID	Идентификатор источника данных	string	Испол привя источ
ORIGIN_ID	Идентификатор элемента в источнике данных	string	Испол привя источ
ORIGIN_VERSION	Оригинальная версия	string	Испол данны перет систе были измен систе могут в CRM следу приве данны

PHONE	Телефон контакта	crm_multifield	Множ
POST	Должность	string	
SECOND_NAME	Отчество	string	Обяза
SOURCE_DESCRIPTION	Описание источника	string	
SOURCE_ID	Идентификатор источника	crm_status	Стату Списс идент получ crm.s filte
STATUS_DESCRIPTION		string	
STATUS_ID		string	
STATUS_SEMANTIC_ID		string	F (fail неусп S (suc успеш P (pro обраб
TITLE	Название лида	string	Обяза
UTM_CAMPAIGN	Обозначение рекламной кампании	string	
UTM_CONTENT	Содержание кампании	string	Напри конте
UTM_MEDIUM	Тип трафика	string	CPC ((банн
UTM_SOURCE	Рекламная система	string	Yande Adwo

UTM_TERM	Условие поиска кампании	string	Напри конте
WEB	URL ресурсов лида	crm_multifield	Множ

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.lead.get

```
crm.lead.get(id)
```

Возвращает лид по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор лида.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(
                        "crm.lead.get",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
```

```
console.dir(result.data());
                            }
                  );
```

Смотри также

- Как правильно выгружать большие объемы данных

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.lead.list

Возвращает список лидов по фильтру. Является реализацией списочного метода для лидов.

При выборке используйте маски:

- "*" - для выборки всех полей (без пользовательских и множественных)
- "UF_*"- для выборки всех пользовательских полей (без множественных)

Маски для выборки множественных полей нет. Для выборки множественных полей укажите нужные в списке выбора ("PHONE", "EMAIL" и так далее).

Внимание! Возможности добавить к фильтру логическое условие OR, если нужно выбрать по нескольким разным полям, нет.

Параметры

См. описание списочных методов.

Пример

```
        // Поиск несконвертированных лидов с
суммой больше нуля
                BX24.callMethod(
                        "crm.lead.list",
```

```
                        {
                                order: {
"STATUS_ID": "ASC" },
                                filter: {
">OPPORTUNITY": 0, "!STATUS_ID": "CONVERTED"
},
                                select: [
"ID", "TITLE", "STATUS_ID", "OPPORTUNITY",
"CURRENCY_ID" ]
                        },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
                                {

console.dir(result.data());

if(result.more())

result.next();
                                }
                        }
                );
                // Поиск лида по телефону
                BX24.callMethod(
                        "crm.lead.list",
                        {
                                filter: {
"PHONE": "555888" },
                                select: [
"ID", "TITLE" ]
                        },
                        function(result)
```

```
                        {

if(result.error())

console.error(result.error());
                                else
                                {

console.dir(result.data());

if(result.more())

result.next();
                                }
                        }
                );
```

Пример выборки лидов за месяц:

```
$result = CRest::call(
    'crm.lead.list',
    [
        'filter' => [
            '>DATE_CREATE' => '2019-10-
01T00:00:00',
            '<DATE_CREATE' => '2019-10-
31T23:59:59'
        ],
        'select' => [
            'ID',
            'DATE_CREATE'
        ]
    ]
);
```

Пример вывода следующего пакета лидов.

```
    BX24.callMethod(
            "crm.lead.list",
            {
                order: { "STATUS_ID": "ASC"
},
                filter: { ">OPPORTUNITY": 0,
"!STATUS_ID": "CONVERTED" },
                select: [ "ID", "TITLE",
"STATUS_ID", "OPPORTUNITY", "CURRENCY_ID" ],
                start: "50"
            },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else
                {

console.dir(result.data());
                    if(result.more())
                        result.next();
                }
            }
        );
```

crm.lead.productrows.get

```
crm.lead.productrows.get(id)
```

Возвращает товарные позиции лида.

Параметры

Параметр	Описание
id	Идентификатор лида.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(

"crm.lead.productrows.get",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
```

```
                                    else

console.dir(result.data());
                            }
                  );
```

crm.lead.productrows.set

```
crm.lead.productrows.set(id, rows)
```

Устанавливает (создаёт или обновляет) товарные позиции лида.

Параметры

Параметр	Описание
id	Идентификатор лида.
rows	Товарные позиции - массив вида array(array("поле"=>"значение"[, ...])[, ...]), где "поле" может принимать значения из возвращаемых методом crm.productrow.fields. Товарные позиции лида, существующие до момента вызова метода, будут заменены новыми. После сохранения будет произведён пересчёт суммы лида.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
```

```
"crm.lead.productrows.set",
                        {
                                id: id,
                                rows:
                                [
                                        {
"PRODUCT_ID": 689, "PRICE": 100.00,
"QUANTITY": 2 },
                                        {
"PRODUCT_ID": 690, "PRICE": 200.00,
"QUANTITY": 1 }
                                ]
                        },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else

console.info(result.data());
                        }
                );
```

crm.lead.update

```
crm.lead.update(id, fields, params)
```

Обновляет существующий лид.

Важно! Настоятельно рекомендуется при обновлении адреса передавать полный набор полей адреса в метод обновления. Особенности обновления полей адреса описаны здесь.

Параметры

Параметр	Описание
id	Идентификатор лида.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.lead.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.lead.fields и посмотрите формат пришедших значений этих полей.
params	Набор параметров. REGISTER_SONET_EVENT - произвести регистрацию события изменения лида в живой ленте. Дополнительно будет отправлено уведомление ответственному за лид.

Пример

```
        var id = prompt("Введите ID");
BX24.callMethod(
        "crm.lead.update",
        {
                id: id,
                fields:
                {
                        "STATUS_ID":
"IN_PROCESS",
                        "CURRENCY_ID":
"USD",
                        "OPPORTUNITY": 15500
                },
                params: {
"REGISTER_SONET_EVENT": "Y" }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                {

console.info(result.data());
                }
        }
);
```

crm.lead.userfield.add

```
crm.lead.userfield.add(fields)
```

Создаёт новое пользовательское поле для лидов.

Системное ограничение на название поля - 20 знаков. К названию пользовательского поля всегда добавляется префикс UF_CRM_, то есть реальная длина названия - 13 знаков.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий описание пользовательского поля.
LIST	Содержит набор значений списка для пользовательских полей типа Список. Указывается при создании/обновлении поля. Каждое значение представляет собой массив с полями: ▪ **VALUE** - значение элемента списка. Поле является обязательным в случае, когда создается новый элемент. ▪ **SORT** - сортировка. ▪ **DEF** - если равно Y, то элемент списка является значением по-умолчанию. Для множественного поля допустимо несколько DEF=Y. Для не

	множественного, дефолтным будет считаться первое. ▪ **XML_ID** - внешний код значения. Параметр учитывается только при обновлении уже существующих значений элемента списка. ▪ **ID** - идентификатор значения. Если он указан, то считается что это обновление существующего значения элемента списка, а не создание нового. Имеет смысл только при вызове методов `*.userfield.update`. ▪ **DEL** - если равно Y, то существующий элемент списка будет удален. Применяется, если заполнен параметр ID.

Полное описание полей можно получить вызовом метода crm.userfield.fields.

Пример #1

```
        BX24.callMethod(
        "crm.lead.userfield.add",
        {
                fields:
                {
                        "FIELD_NAME":
"MY_STRING",
                        "EDIT_FORM_LABEL":
"Моя строка",
                        "LIST_COLUMN_LABEL":
"Моя строка",
                        "USER_TYPE_ID":
"string",
                        "XML_ID":
"MY_STRING",
                        "SETTINGS": {
```

```
"DEFAULT_VALUE": "Привет, мир!" }
                }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

Пример #2

```
        BX24.callMethod(
        "crm.lead.userfield.add",
        {
                fields:
                {
                        "FIELD_NAME":
"MY_LIST",
                        "EDIT_FORM_LABEL":
"Мой список",
                        "LIST_COLUMN_LABEL":
"Мой список",
                        "USER_TYPE_ID":
"enumeration",
                        "LIST": [ { "VALUE":
"Элемент #1" }, { "VALUE": "Элемент #2" }, {
"VALUE": "Элемент #3" }, { "VALUE": "Элемент
#4" }, { "VALUE": "Элемент #5" } ],
                        "XML_ID": "MY_LIST",
```

```
                                "SETTINGS": {
"LIST_HEIGHT": 3 }
                }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

crm.lead.userfield.delete

```
crm.lead.userfield.delete(id)
```

Удаляет пользовательское поле лидов.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.lead.userfield.delete",
        {
                id: id
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
```

```
                    else

console.info(result.data());
        }
);
```

crm.lead.userfield.get

```
crm.lead.userfield.get(id)
```

Возвращает пользовательское поле лидов по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.

Пример

```
        var id = prompt("Введите ID");
BX24.callMethod(
        "crm.lead.userfield.get",
        {
                id: id
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
```

```
                    else

console.dir(result.data());
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.lead.userfield.list

```
crm.lead.userfield.list(order,filter)
```

Возвращает список пользовательских полей лидов по фильтру.

Параметры

Параметр	Описание
order	Поля сортировки.
filter	Поля фильтра.

Пример

```
BX24.callMethod(
        "crm.lead.userfield.list",
        {
                order: { "SORT": "ASC" },
                filter: { "MANDATORY": "N" }
        },
        function(result)
        {
                if(result.error())
```

```
console.error(result.error());
                else
                {

console.dir(result.data());
                        if(result.more())

result.next();
                }
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.lead.userfield.update

```
crm.lead.userfield.update(id,fields)
```

Обновляет существующее пользовательское поле лидов.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.userfield.fields.
LIST	Содержит набор значений списка для пользовательских полей типа Список. Указывается при создании/обновлении поля. Каждое значение представляет собой массив с полями: ▪ **VALUE** - значение элемента списка. Поле является обязательным в случае, когда создается новый элемент. ▪ **SORT** - сортировка. ▪ **DEF** - если равно Y, то элемент списка является значением по-умолчанию. Для множественного поля допустимо несколько DEF=Y. Для не

множественного, дефолтным будет считаться первое.

- **XML_ID** - внешний код значения. Параметр учитывается только при обновлении уже существующих значений элемента списка.
- **ID** - идентификатор значения. Если он указан, то считается что это обновление существующего значения элемента списка, а не создание нового. Имеет смысл только при вызове методов `*.userfield.update`.
- **DEL** - если равно Y, то существующий элемент списка будет удален. Применяется, если заполнен параметр ID.

Пример

```
        var id = prompt("Введите ID");
var label = prompt("Введите новое
название");
BX24.callMethod(
        "crm.lead.userfield.update",
        {
                id: id,
                fields:
                {
                        "EDIT_FORM_LABEL":
label,
                        "LIST_COLUMN_LABEL":
label
                }
        },
        function(result)
        {
                if(result.error())
```

```
console.error(result.error());
                else
                {

console.dir(result.data());
                        if(result.more())

result.next();
                }
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmLeadAdd

Событие, вызываемое при создании лида.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора созданного лида.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmLeadDelete

Событие, вызываемое при удалении лида.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора удаленного лида.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmLeadUpdate

Событие, вызываемое при обновлении лида.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора обновленного лида.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmLeadUserFieldAdd

Событие, вызываемое при добавлении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmLeadUserFieldDelete

Событие, вызываемое при удалении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmLeadUserFieldSetEnumV

Событие, вызываемое при изменении набора значений для пользовательского поля списочного типа.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmLeadUserFieldUpdate

Событие, вызываемое при изменении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Компании::crm.company.add

```
crm.company.add(fields, params)
```

Создаёт новую компанию.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей компании. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.company.fields и посмотрите формат пришедших значений этих полей.
params	Набор параметров. REGISTER_SONET_EVENT - произвести регистрацию события добавления компании в живой ленте. Дополнительно будет отправлено уведомление ответственному за компанию.

Пример

```
BX24.callMethod(
        "crm.company.add",
        {
                fields:
                {
                        "TITLE": "ИП Титов",
                        "COMPANY_TYPE":
"CUSTOMER",
                        "INDUSTRY":
"MANUFACTURING",
                        "EMPLOYEES":
"EMPLOYEES_2",
                        "CURRENCY_ID":
"RUB",
                        "REVENUE" : 3000000,
                        "LOGO": {
"fileData": document.getElementById('logo')
},
                        "OPENED": "Y",
                        "ASSIGNED_BY_ID": 1,
                        "PHONE": [ {
"VALUE": "555888", "VALUE_TYPE": "WORK" } ]
                },
                params: {
"REGISTER_SONET_EVENT": "Y" }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.info("Создана компания с ID " +
result.data());
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Компании::crm.company.conta

```
crm.company.contact.add(id, fields)
```

Добавляет контакт к указанной компании.

Параметры

Параметр	Описание
id	Идентификатор компании.
fields	Объект со следующими полями: ▪ CONTACT_ID - идентификатор контакта (обязательное поле) ▪ SORT - индекс сортировки ▪ IS_PRIMARY - флаг первичного контакта **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.company.fields и посмотрите формат пришедших значений этих полей.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Компании::crm.company.conta

```
crm.company.contact.delete(id, fields)
```

Удаляет контакт из указанной компании.

Параметры

Параметр	Описание
id	Идентификатор контакта.
fields	Объект со следующими полями: ▪ CONTACT_ID - идентификатор контакта (обязательное поле)

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Компании::crm.company.conta

```
crm.company.contact.fields()
```

Возвращает для связи компания-контакт описание полей, используемых методами семейства crm.company.contact.*, то есть crm.company.contact.items.get, crm.company.contact.items.set, crm.company.contact.add и т.д.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(

"crm.company.contact.fields",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                              else
```

```
console.dir(result.data());
                            }
                    );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Компании::crm.company.conta

```
crm.company.contact.items.delete(id)
```

Очищает набор контактов, связанных с указанной компанией.

Параметры

Параметр	Описание
id	Идентификатор компании.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.company.contact.items.delete",
        {
                id: id
        },
        function(result)
        {
                if(result.error())
```

```
console.error(result.error());
                else

console.info(result.data());
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Компании::crm.company.conta

```
crm.company.contact.items.get(id)
```

Возвращает набор контактов, связанных с указанной компанией.

Параметры

Параметр	Описание
id	Идентификатор компании.

Результат возвращается в виде массива объектов, каждый из которых содержит следующие поля:

Поле	Описание
CONTACT_ID	Идентификатор контакта
SORT	Индекс сортировки
ROLE_ID	Идентификатор роли (зарезервировано)
IS_PRIMARY	Флаг первичного контакта

Пример вызова

```
var id = prompt("Введите ID");
BX24.callMethod(
"crm.company.contact.items.get",
{
id: id
},
function(result)
{
if(result.error())
console.error(result.error());
else
console.dir(result.data());
}
);
```

Компании::crm.company.conta

```
crm.company.contact.items.set(id, items)
```

Устанавливает набор контактов, связанных с указанной компанией.

Параметры

Параметр	Описание
id	Идентификатор компании.
items	Набор контактов в виде массива объектов со следующими полями: ▪ CONTACT_ID - идентификатор контакта (обязательное поле) ▪ SORT - индекс сортировки ▪ IS_PRIMARY - флаг первичного контакта

© «Битрикс», 2001-2008, «1С-

Компании::crm.company.delet

```
crm.company.delete (id)
```

Удаляет компанию и все связанные с ней объекты.

Параметры

Параметр	Описание
id	Идентификатор компании.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(

"crm.company.delete",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
```

```
                                    else

console.info(result.data());
                            }
                    );
```

Компании::crm.company.fields

```
crm.company.fields()
```

Возвращает описание полей компании, в том числе пользовательских.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(
"crm.company.fields",
                        {},
                        function(result)
                        {
if(result.error())
console.error(result.error());
                                else
console.dir(result.data());
```

```
                                        }
                        );
```

Поля

Поле	Описание	Тип	П
ADDRESS	Адрес компании	string	
ADDRESS_2	Вторая страница адреса	string	В с р н
ADDRESS_CITY	Город	string	
ADDRESS_COUNTRY	Страна	string	
ADDRESS_COUNTRY_CODE	Код страны	string	
ADDRESS_LEGAL		string	
ADDRESS_POSTAL_CODE	Почтовый индекс	string	
ADDRESS_PROVINCE	Область	string	
ADDRESS_REGION	Район	string	
ASSIGNED_BY_ID	Связано с пользователем по ID	user	
BANKING_DETAILS	Банковские реквизиты	string	
COMMENTS	Комментарии	string	

COMPANY_TYPE	Тип компании	crm_status	
CREATED_BY_ID	Кем создана	user	Т ч
CURRENCY_ID	Валюта	crm_currency	
DATE_CREATE	Дата создания	datetime	Т ч
DATE_MODIFY	Дата изменения	datetime	Т ч
EMAIL	Адрес электронной почты	crm_multifield	М
EMPLOYESS	Количество сотрудников	crm_status	
HAS_EMAIL	Проверка заполненности поля электронной почты	char	Т ч
HAS_PHONE	Проверка заполненности поля телефон	char	Т ч
ID	Идентификатор компании	integer	Т ч
IM	Мессенджеры	crm_multifield	М
INDUSTRY	Сфера деятельности	crm_status	
IS_MY_COMPANY		char	
LEAD_ID	Идентификатор лида,	crm_lead	Т ч

	связанного с компанией		
LOGO	Логотип	file	
MODIFY_BY_ID	Идентификатор автора последнего изменения	user	Т ч
OPENED	Доступен для всех	char	
ORIGINATOR_ID	Идентификатор источника данных	string	И т п в и
ORIGIN_ID	Идентификатор элемента в источнике данных	string	И т п в и
ORIGIN_VERSION	Оригинальная версия	string	И з о п в с д и н в т м р С о с в п п д

PHONE	Телефон компании	crm_multifield	М
REG_ADDRESS	Юридический адрес компании	string	У и с
REG_ADDRESS_2	Вторая страница юридического адреса	string	В с р н У и с
REG_ADDRESS_CITY	Город юридического адреса	string	У и с
REG_ADDRESS_COUNTRY	Страна юридического адреса	string	У и с
REG_ADDRESS_COUNTRY_CODE	Код страны юридического адреса	string	У и с
REG_ADDRESS_LEGAL		string	У и с
REG_ADDRESS_POSTAL_CODE	Почтовый индекс юридического адреса	string	У и с
REG_ADDRESS_PROVINCE	Область юридического адреса	string	У и с
REG_ADDRESS_REGION	Район юридического адреса	string	У и с
REVENUE	Годовой оборот	double	

TITLE	Название	string	С
UTM_CAMPAIGN	Обозначение рекламной кампании	string	
UTM_CONTENT	Содержание кампании	string	Н к о
UTM_MEDIUM	Тип трафика	string	С (С
UTM_SOURCE	Рекламная система	string	Y G д
UTM_TERM	Условие поиска кампании	string	Н к к р
WEB	URL ресурсов компании	crm_multifield	М

Компании::crm.company.get

```
crm.company.get(id)
```

Возвращает компанию по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор компании.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(
                        "crm.company.get",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
```

```
console.dir(result.data());
                            }
                  );
```

Компании::crm.company.list

Возвращает список компаний по фильтру. Является реализацией списочного метода для компаний.

При выборке используйте маски:

- "*" - для выборки всех полей (без пользовательских и множественных)
- "UF_*"- для выборки всех пользовательских полей (без множественных)

Маски для выборки множественных полей нет. Для выборки множественных полей укажите нужные в списке выбора ("PHONE", "EMAIL" и так далее).

Параметры

См. описание списочных методов.

Пример

```
        //Поиск компаний по сфере
деятельности и типу
                BX24.callMethod(
                        "crm.company.list",
                        {
                                order: {
"DATE_CREATE": "ASC" },
                                filter: {
"INDUSTRY": "MANUFACTURING", "COMPANY_TYPE":
```

```
"CUSTOMER" },
                                select: [
"ID", "TITLE", "CURRENCY_ID", "REVENUE" ]
                        },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
                                {

console.dir(result.data());

if(result.more())

result.next();
                                }
                        }
                );
                //Поиск компании по телефону
                BX24.callMethod(
                        "crm.company.list",
                        {
                                filter: {
"PHONE": "555888" },
                                select: [
"ID", "TITLE" ]
                        },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
```

```
                                        {
console.dir(result.data());
if(result.more())
result.next();
                                        }
                                }
                        );
```

Компании::crm.company.upda

```
crm.company.update(id, fields, params)
```

Обновляет существующую компанию.

Важно! Настоятельно рекомендуется при обновлении адреса передавать полный набор полей адреса в метод обновления. Особенности обновления полей адреса описаны здесь.

Параметры

Параметр	Описание
id	Идентификатор компании.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.company.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.company.fields и посмотрите формат пришедших значений этих полей.
params	Набор параметров. REGISTER_SONET_EVENT - произвести регистрацию события изменения компании в живой ленте. Дополнительно будет отправлено уведомление ответственному за компанию.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.company.update",
        {
                id: id,
                fields:
                {
                        "CURRENCY_ID":
"RUB",
                        "REVENUE" : 500000,
                        "EMPLOYEES":
"EMPLOYEES_3"
                },
                params: {
"REGISTER_SONET_EVENT": "Y" }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                {

console.info(result.data());
                }
        }
);
```

Компании::crm.company.userf

```
crm.company.userfield.add(fields)
```

Создаёт новое пользовательское поле для компаний.

Системное ограничение на название поля - 20 знаков. К названию пользовательского поля всегда добавляется префикс UF_CRM_, то есть реальная длина названия - 13 знаков.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий описание пользовательского поля.
LIST	Содержит набор значений списка для пользовательских полей типа Список. Указывается при создании/обновлении поля. Каждое значение представляет собой массив с полями: ▪ **VALUE** - значение элемента списка. Поле является обязательным в случае, когда создается новый элемент. ▪ **SORT** - сортировка. ▪ **DEF** - если равно Y, то элемент списка является значением по-умолчанию. Для множественного поля допустимо несколько DEF=Y. Для не

	множественного, дефолтным будет считаться первое. ▪ **XML_ID** - внешний код значения. Параметр учитывается только при обновлении уже существующих значений элемента списка. ▪ **ID** - идентификатор значения. Если он указан, то считается что это обновление существующего значения элемента списка, а не создание нового. Имеет смысл только при вызове методов `*.userfield.update`. ▪ **DEL** - если равно Y, то существующий элемент списка будет удален. Применяется, если заполнен параметр ID.

Полное описание полей можно получить вызовом метода crm.userfield.fields.

Пример #1

```
BX24.callMethod(
        "crm.company.userfield.add",
        {
                fields:
                {
                        "FIELD_NAME":
"MY_STRING",
                        "EDIT_FORM_LABEL":
"Моя строка",
                        "LIST_COLUMN_LABEL":
"Моя строка",
                        "USER_TYPE_ID":
"string",
                        "XML_ID":
"MY_STRING",
                        "SETTINGS": {
```

```
"DEFAULT_VALUE": "Привет, мир!" }
                }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

Пример #2

```
BX24.callMethod(
        "crm.company.userfield.add",
        {
                fields:
                {
                        "FIELD_NAME":
"MY_LIST",
                        "EDIT_FORM_LABEL":
"Мой список",
                        "LIST_COLUMN_LABEL":
"Мой список",
                        "USER_TYPE_ID":
"enumeration",
                        "LIST": [ { "VALUE":
"Элемент #1" }, { "VALUE": "Элемент #2" }, {
"VALUE": "Элемент #3" }, { "VALUE": "Элемент
#4" }, { "VALUE": "Элемент #5" } ],
                        "XML_ID": "MY_LIST",
```

```
                                "SETTINGS": {
"LIST_HEIGHT": 3 }
                }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

Компании::crm.company.userf

```
crm.company.userfield.delete(id)
```

Удаляет пользовательское поле компаний.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.company.userfield.delete",
        {
                id: id
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
```

```
                    else

console.info(result.data());
        }
);
```

Компании::crm.company.userf

```
crm.company.userfield.get(id)
```

Возвращает пользовательское поле компаний по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.company.userfield.get",
        {
                id: id
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
```

```
                    else

console.dir(result.data());
        }
);
```

Компании::crm.company.userf

```
crm.company.userfield.list(order, filter)
```

Возвращает список пользовательских полей компаний по фильтру.

Параметры

Параметр	Описание
order	Поля сортировки.
filter	Поля фильтра.

Пример

```
BX24.callMethod(
        "crm.company.userfield.list",
        {
                order: { "SORT": "ASC" },
                filter: { "MANDATORY": "N" }
        },
        function(result)
        {
                if(result.error())
```

```
console.error(result.error());
                else
                {

console.dir(result.data());
                        if(result.more())

result.next();
                }
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Компании::crm.company.userf

```
crm.company.userfield.update(id, fields)
```

Обновляет существующее пользовательское поле компаний.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.userfield.fields.
LIST	Содержит набор значений списка для пользовательских полей типа Список. Указывается при создании/обновлении поля. Каждое значение представляет собой массив с полями: ▪ **VALUE** - значение элемента списка. Поле является обязательным в случае, когда создается новый элемент. ▪ **SORT** - сортировка. ▪ **DEF** - если равно Y, то элемент списка является значением по-умолчанию. Для множественного поля допустимо несколько DEF=Y. Для не

множественного, дефолтным будет считаться первое.

- **XML_ID** - внешний код значения. Параметр учитывается только при обновлении уже существующих значений элемента списка.
- **ID** - идентификатор значения. Если он указан, то считается что это обновление существующего значения элемента списка, а не создание нового. Имеет смысл только при вызове методов `*.userfield.update`.
- **DEL** - если равно Y, то существующий элемент списка будет удален. Применяется, если заполнен параметр ID.

Пример

```
var id = prompt("Введите ID");
var label = prompt("Введите новое
название");
BX24.callMethod(
        "crm.company.userfield.update",
        {
                id: id,
                fields:
                {
                        "EDIT_FORM_LABEL":
label,
                        "LIST_COLUMN_LABEL":
label
                }
        },
        function(result)
        {
                if(result.error())
```

```
console.error(result.error());
                else
                {

console.dir(result.data());
                        if(result.more())

result.next();
                }
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmCompanyAdd

Событие, вызываемое при создании компании.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора созданной компании.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmCompanyDelete

Событие, вызываемое при удалении компании.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора удаленной компании.

© «Битрикс», 2001-2008, «1С-Битрикс»

onCrmCompanyUpdate

Событие, вызываемое при обновлении компании.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора обновленной компании.

© «Битрикс», 2001-2008, «1С-

onCrmCompanyUserFieldAdd

Событие, вызываемое при добавлении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmCompanyUserFieldDelete

Событие, вызываемое при удалении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmCompanyUserFieldSetEn

Событие, вызываемое при изменении набора значений для пользовательского поля списочного типа.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmCompanyUserFieldUpdat

Событие, вызываемое при изменении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Контакты::crm.contact.add

```
crm.contact.add(fields, params)
```

Создаёт новый контакт.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей контакта. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.contact.fields и посмотрите формат пришедших значений этих полей.
params	Набор параметров. REGISTER_SONET_EVENT - произвести регистрацию события добавления контакта в живой ленте. Дополнительно будет отправлено уведомление ответственному за контакт.

Одно из полей: **NAME** или **LAST_NAME** обязательно должно быть заполнено.

Пример

```
BX24.callMethod(
        "crm.contact.add",
        {
                fields:
                {
                        "NAME": "Глеб",
                        "SECOND_NAME":
"Егорович",
                        "LAST_NAME":
"Титов",
                        "OPENED": "Y",
                        "ASSIGNED_BY_ID": 1,
                        "TYPE_ID": "CLIENT",
                        "SOURCE_ID": "SELF",
                        "PHOTO": {
"fileData": document.getElementById('photo')
},
                        "PHONE": [ {
"VALUE": "555888", "VALUE_TYPE": "WORK" } ]
                },
                params: {
"REGISTER_SONET_EVENT": "Y" }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                        console.info("Создан
контакт с ID " + result.data());
        }
);
```

Контакты::crm.contact.compar

```
crm.contact.company.add(id, fields)
```

Добавляет компанию к указанному контакту.

Параметры

Параметр	Описание
id	Идентификатор контакта.
fields	Объект со следующими полями: ▪ COMPANY_ID - идентификатор компании (обязательное поле) ▪ SORT - индекс сортировки ▪ IS_PRIMARY - флаг первичной компании **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.contact.fields и посмотрите формат пришедших значений этих полей.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Контакты::crm.contact.compar

```
crm.contact.company.delete(id, fields)
```

Удаляет компанию из указанного контакта.

Параметры

Параметр	Описание
id	Идентификатор контакта.
fields	Объект со следующими полями: ▪ COMPANY_ID - идентификатор компании (обязательное поле)

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Контакты::crm.contact.compar

```
crm.contact.company.fields()
```

Возвращает для связи контакт-компания описание полей, используемых методами семейства crm.contact.company.*, то есть crm.contact.company.items.get, crm.contact.company.items.set, crm.contact.company.add и т.д.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(

"crm.contact.company.fields",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                              else
```

```
console.dir(result.data());
                            }
                    );
```

Контакты::crm.contact.compar

```
crm.contact.company.items.delete(id)
```

Очищает набор компаний, связанных с указанным контактом.

Параметры

Параметр	Описание
id	Идентификатор контакта.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.contact.company.items.delete",
        {
                id: id
        },
        function(result)
        {
                if(result.error())
```

```
console.error(result.error());
                else

console.info(result.data());
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Контакты::crm.contact.compar

```
crm.contact.company.items.get(id)
```

Возвращает набор компаний, связанных с указанным контактом.

Параметры

Параметр	Описание
id	Идентификатор контакта.

Результат возвращается в виде массива объектов, каждый из которых содержит следующие поля:

Поле	Описание
COMPANY_ID	Идентификатор компании
SORT	Индекс сортировки
ROLE_ID	Идентификатор роли (зарезервировано)
IS_PRIMARY	Флаг первичной компании

Пример вызова

```
var id = prompt("Введите ID");
BX24.callMethod(
"crm.contact.company.items.get",
{
id: id
},
function(result)
{
if(result.error())
console.error(result.error());
else
console.dir(result.data());
}
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Контакты::crm.contact.compar

```
crm.contact.company.items.set(id, items)
```

Устанавливает набор компаний, связанных с указанным контактом.

Параметры

Параметр	Описание
id	Идентификатор контакта.
items	Набор компаний в виде массива объектов со следующими полями: ▪ COMPANY_ID - идентификатор компании (обязательное поле) ▪ SORT - индекс сортировки ▪ IS_PRIMARY - флаг первичной компании

© «Битрикс», 2001-2008, «1С-

Контакты::crm.contact.delete

```
crm.contact.delete(id)
```

Удаляет контакт и все связанные с ним объекты.

Параметры

Параметр	Описание
id	Идентификатор контакта.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(

"crm.contact.delete",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
```

```
                                    else

console.info(result.data());
                            }
                    );
```

Контакты::crm.contact.fields

```
crm.contact.fields()
```

Возвращает описание полей контакта, в том числе пользовательских.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(

"crm.contact.fields",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else

console.dir(result.data());
```

```
                                        }
                                );
```

Поля, возвращаемые методом

Поле	Описание	Тип	
ADDRESS	Адрес контакта	string	
ADDRESS_2	Вторая страница адреса	string	В неко разбив
ADDRESS_CITY	Город	string	
ADDRESS_COUNTRY	Страна	string	
ADDRESS_COUNTRY_CODE	Код страны	string	
ADDRESS_POSTAL_CODE	Почтовый индекс	string	
ADDRESS_PROVINCE	Область	string	
ADDRESS_REGION	Район	string	
ASSIGNED_BY_ID	Связано с пользователем по ID	user	
BIRTHDATE	Дата рождения	date	
COMMENTS	Комментарии	string	
COMPANY_ID	Привязка контакта к компании	crm_company	Устаре совмес

COMPANY_IDS	Привязка контакта к нескольким компаниям.	crm_company	Множе В мето и crm. исполь массив crm.co crm.co необхо crm.co для по компан
CREATED_DY_ID	Кем создана	user	Только
DATE_CREATE	Дата создания	datetime	Только
DATE_MODIFY	Дата изменения	datetime	Только
EMAIL	Адрес электронной почты	crm_multifield	Множе
EXPORT	Участвует ли контакт в экспорте. Если N, то выгрузить его невозможно.	char	Защит которы клиент
FACE_ID	Привязка к лицам из модуля faceid	integer	
HAS_EMAIL	Проверка заполненности поля электронной почты	char	Только
HAS_PHONE	Проверка заполненности поля телефон	char	Только
HONORIFIC	Вид обращения	crm_status	

ID	Идентификатор контакта	integer	Только
IM	Мессенджеры	crm_multifield	Множе
LAST_NAME	Фамилия	string	Обязат
LEAD_ID	Идентификатор лида, связанного с контактом	crm_lead	Только
MODIFY_BY_ID	Идентификатор автора последнего изменения	user	Только
NAME	Имя	string	Обязат
OPENED	Доступен для всех	char	
ORIGINATOR_ID	Идентификатор источника данных	string	Исполь привяз источн
ORIGIN_ID	Идентификатор элемента в источнике данных	string	Исполь привяз источн
ORIGIN_VERSION	Оригинальная версия	string	Исполь данны перети систем импор изменя систем могут CRM б следу приве данны

PHONE	Телефон контакта	crm_multifield	Множе
PHOTO	Фото контакта	file	
POST	Должность	string	
SECOND_NAME	Отчество	string	Обязат
SOURCE_DESCRIPTION	Описание источника	string	Тексто
SOURCE_ID	Идентификатор источника	crm_status	Статус Список идент получ crm.st filter
TYPE_ID	Идентификатор типа	crm_status	Статус
UTM_CAMPAIGN	Обозначение рекламной кампании	string	
UTM_CONTENT	Содержание кампании	string	Напри объявл
UTM_MEDIUM	Тип трафика	string	CPC (о (банне
UTM_SOURCE	Рекламная система	string	Yandex Adwor
UTM_TERM	Условие поиска кампании	string	Напри контек
WEB	URL ресурсов контакта	crm_multifield	Множе

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Контакты::crm.contact.get

```
crm.contact.get(id)
```

Возвращает контакт по идентификатору.

Для получения списка компаний, привязанных к контакту используйте метод crm.contact.company.items.get.

Параметры

Параметр	Описание
id	Идентификатор контакта.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
"crm.contact.get",
{ id: id },
function(result)
{

if(result.error())
```

```
console.error(result.error());
                                else

console.dir(result.data());
                        }
                );
```

Контакты::crm.contact.list

Возвращает список контактов по фильтру. Является реализацией списочного метода для контактов.

При выборке используйте маски:

- "*" - для выборки всех полей (без пользовательских и множественных)
- "UF_*"- для выборки всех пользовательских полей (без множественных)

Маски для выборки множественных полей нет. Для выборки множественных полей укажите нужные в списке выбора ("PHONE", "EMAIL" и так далее).

Для получения списка компаний, привязанных к контакту используйте метод crm.contact.company.items.get.

Внимание! Поля: телефон, почта, сайт, мессенджеры - множественные. По ним фильтры работают только на точное совпадение.

Параметры

См. описание списочных методов.

Пример

```
BX24.callMethod(
                "crm.contact.list",
```

```
                        {
                                order: {
"DATE_CREATE": "ASC" },
                                filter: {
"TYPE_ID": "CLIENT" },
                                select: [
"ID", "NAME", "LAST_NAME", "TYPE_ID",
"SOURCE_ID" ]
                        },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
                                {

console.dir(result.data());

if(result.more())

result.next();
                                }
                        }
                );

//Поиск контакта по телефону
                BX24.callMethod(
                        "crm.contact.list",
                        {
                                filter: {
"PHONE": "555888" },
                                select: [
"ID", "NAME", "LAST_NAME" ]
                        },
                        function(result)
```

```
                        {

if(result.error())

console.error(result.error());
                                else
                                {

console.dir(result.data());

if(result.more())

result.next();
                                }
                        }
                );
```

Контакты::crm.contact.update

```
crm.contact.update(id, fields, params)
```

Обновляет существующий контакт.

Важно! Настоятельно рекомендуется при обновлении адреса передавать полный набор полей адреса в метод обновления. Особенности обновления полей адреса описаны здесь.

Параметры

Параметр	Описание
id	Идентификатор контакта.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.contact.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.contact.fields и посмотрите формат пришедших значений этих полей.
params	Набор параметров. REGISTER_SONET_EVENT - произвести регистрацию события изменения контакта в живой ленте. Дополнительно будет отправлено уведомление ответственному за контакт.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.contact.update",
        {
                id: id,
                fields:
                {
                        "TYPE_ID":
"JOURNALIST",
                        "SOURCE_ID":
"CONFERENCE"
                },
                params: {
"REGISTER_SONET_EVENT": "Y" }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                {

console.info(result.data());
                }
        }
);
```

Контакты::crm.contact.userfiel

```
crm.contact.userfield.add(fields)
```

Создаёт новое пользовательское поле для контактов.

Системное ограничение на название поля - 20 знаков. К названию пользовательского поля всегда добавляется префикс UF_CRM_, то есть реальная длина названия - 13 знаков.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий описание пользовательского поля.
LIST	Содержит набор значений списка для пользовательских полей типа Список. Указывается при создании/обновлении поля. Каждое значение представляет собой массив с полями: ▪ **VALUE** - значение элемента списка. Поле является обязательным в случае, когда создается новый элемент. ▪ **SORT** - сортировка. ▪ **DEF** - если равно Y, то элемент списка является значением по-умолчанию. Для множественного поля допустимо несколько DEF=Y. Для не

множественного, дефолтным будет считаться первое.

- **XML_ID** - внешний код значения. Параметр учитывается только при обновлении уже существующих значений элемента списка.
- **ID** - идентификатор значения. Если он указан, то считается что это обновление существующего значения элемента списка, а не создание нового. Имеет смысл только при вызове методов `*.userfield.update`.
- **DEL** - если равно Y, то существующий элемент списка будет удален. Применяется, если заполнен параметр ID.

Полное описание полей можно получить вызовом метода crm.userfield.fields.

Пример #1

```
BX24.callMethod(
        "crm.contact.userfield.add",
        {
                fields:
                {
                        "FIELD_NAME":
"MY_STRING",
                        "EDIT_FORM_LABEL":
"Моя строка",
                        "LIST_COLUMN_LABEL":
"Моя строка",
                        "USER_TYPE_ID":
"string",
                        "XML_ID":
"MY_STRING",
                        "SETTINGS": {
```

```
"DEFAULT_VALUE": "Привет, мир!" }
                }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

Пример #2

```
BX24.callMethod(
        "crm.contact.userfield.add",
        {
                fields:
                {
                        "FIELD_NAME":
"MY_LIST",
                        "EDIT_FORM_LABEL":
"Мой список",
                        "LIST_COLUMN_LABEL":
"Мой список",
                        "USER_TYPE_ID":
"enumeration",
                        "LIST": [ { "VALUE":
"Элемент #1" }, { "VALUE": "Элемент #2" }, {
"VALUE": "Элемент #3" }, { "VALUE": "Элемент
#4" }, { "VALUE": "Элемент #5" } ],
                        "XML_ID": "MY_LIST",
```

```
                        "SETTINGS": {
"LIST_HEIGHT": 3 }
                }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

Контакты::crm.contact.userfiel

```
crm.contact.userfield.delete(id)
```

Удаляет пользовательское поле контактов.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.contact.userfield.delete",
        {
                id: id
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
```

```
                    else

console.info(result.data());
        }
);
```

Контакты::crm.contact.userfiel

```
crm.contact.userfield.get(id)
```

Возвращает пользовательское поле контактов по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.contact.userfield.get",
        {
                id: id
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
```

```
                    else

console.dir(result.data());
        }
);
```

Контакты::crm.contact.userfiel

```
crm.contact.userfield.list(order, filter)
```

Возвращает список пользовательских полей контактов по фильтру. Выводится и информация об этих полях, но без названия, которое присвоил полю пользователь, только внутренний идентификатор. Если нужно пользовательское название поля, удобней будет воспользоваться методом crm.contact.list, который выводит как стандартные поля, так и пользовательские.

Параметры

Параметр	Описание
order	Поля сортировки.
filter	Поля фильтра.

Пример

```
BX24.callMethod(
        "crm.contact.userfield.list",
        {
                order: { "SORT": "ASC" },
                filter: { "MANDATORY": "N" }
```

```
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                {

console.dir(result.data());
                        if(result.more())

result.next();
                }
        }
);
```

Контакты::crm.contact.userfiel

```
crm.contact.userfield.update(id, fields)
```

Обновляет существующее пользовательское поле контактов.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.userfield.fields.
LIST	Содержит набор значений списка для пользовательских полей типа Список. Указывается при создании/обновлении поля. Каждое значение представляет собой массив с полями: ▪ **VALUE** - значение элемента списка. Поле является обязательным в случае, когда создается новый элемент. ▪ **SORT** - сортировка. ▪ **DEF** - если равно Y, то элемент списка является значением по-умолчанию. Для множественного поля допустимо несколько DEF=Y. Для не

множественного, дефолтным будет считаться первое.

- **XML_ID** - внешний код значения. Параметр учитывается только при обновлении уже существующих значений элемента списка.
- **ID** - идентификатор значения. Если он указан, то считается что это обновление существующего значения элемента списка, а не создание нового. Имеет смысл только при вызове методов `*.userfield.update`.
- **DEL** - если равно Y, то существующий элемент списка будет удален. Применяется, если заполнен параметр ID.

Пример

```
var id = prompt("Введите ID");
var label = prompt("Введите новое
название");
BX24.callMethod(
        "crm.contact.userfield.update",
        {
                id: id,
                fields:
                {
                        "EDIT_FORM_LABEL":
label,
                        "LIST_COLUMN_LABEL":
label
                }
        },
        function(result)
        {
                if(result.error())
```

```
console.error(result.error());
                else
                {

console.dir(result.data());
                        if(result.more())

result.next();
                }
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmContactAdd

Событие, вызываемое при создании контакта.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора созданного контакта.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmContactDelete

Событие, вызываемое при удалении контакта.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора удалённого контакта.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmContactUpdate

Событие, вызываемое при обновлении контакта.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора обновленного контакта.

© «Битрикс», 2001-2008, «1С-Битрикс»

onCrmContactUserFieldAdd

Событие, вызываемое при добавлении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmContactUserFieldDelete

Событие, вызываемое при удалении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmContactUserFieldSetEnu

Событие, вызываемое при изменении набора значений для пользовательского поля списочного типа.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmContactUserFieldUpdate

Событие, вызываемое при изменении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Сделки::crm.deal.add

```
crm.deal.add(fields, params)
```

Создаёт новую сделку.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей сделки. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.deal.fields и посмотрите формат пришедших значений этих полей.
params	Набор параметров. REGISTER_SONET_EVENT - произвести регистрацию события добавления сделки в живой ленте. Дополнительно будет отправлено уведомление ответственному за сделку.

Пример

```
        var current = new Date();
var nextMonth = new Date();
nextMonth.setMonth(current.getMonth() + 1);
var date2str = function(d)
{
         return d.getFullYear() + '-' +
paddatepart(1 + d.getMonth()) + '-' +
paddatepart(d.getDate()) + 'T' +
paddatepart(d.getHours()) + ':' +
paddatepart(d.getMinutes()) + ':' +
paddatepart(d.getSeconds()) + '+03:00';
};
var paddatepart = function(part)
{
         return part >= 10 ? part.toString()
: '0' + part.toString();
};

BX24.callMethod(
        "crm.deal.add",
        {
                fields:
                {
                        "TITLE": "Плановая
продажа",
                        "TYPE_ID": "GOODS",
                        "STAGE_ID": "NEW",
                        "COMPANY_ID": 3,
                        "CONTACT_ID": 3,
                        "OPENED": "Y",
                        "ASSIGNED_BY_ID": 1,
                        "PROBABILITY": 30,
                        "CURRENCY_ID":
"USD",
                        "OPPORTUNITY": 5000,
                        "CATEGORY_ID": 5,
                        "BEGINDATE":
```

```
date2str(current),
                        "CLOSEDATE":
date2str(nextMonth)
                },
                params: {
"REGISTER_SONET_EVENT": "Y" }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.info("Создана сделка с ID " +
result.data());
        }
);
```

Сделки::crm.deal.contact.add

```
crm.deal.contact.add(id, fields)
```

Добавляет контакт к указанной сделке.

Параметры

Параметр	Описание
id	Идентификатор сделки.
fields	Объект со следующими полями: ▪ CONTACT_ID - идентификатор контакта (обязательное поле) ▪ SORT - индекс сортировки ▪ IS_PRIMARY - флаг первичного контакта **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.deal.contact.fields и посмотрите формат пришедших значений этих полей.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Сделки::crm.deal.contact.delet

```
crm.deal.contact.delete(id, fields)
```

Удаляет контакт из указанной сделки.

Параметры

Параметр	Описание
id	Идентификатор сделки.
fields	Объект со следующими полями: ▪ CONTACT_ID - идентификатор контакта (обязательное поле)

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Сделки::crm.deal.contact.fields

```
crm.deal.contact.fields()
```

Возвращает для связи сделка-контакт описание полей, используемых методами семейства crm.deal.contact.*, то есть crm.deal.contact.items.get, crm.deal.contact.items.set, crm.deal.contact.add и т.д.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(

"crm.deal.contact.fields",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                              else
```

```
console.dir(result.data());
                            }
                    );
```

Сделки::crm.deal.contact.items

```
crm.deal.contact.items.delete(id)
```

Очищает набор контактов, связанных с указанной сделкой.

Параметры

Параметр	Описание
id	Идентификатор сделки.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.deal.contact.items.delete",
        {
                id: id
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
```

```
                    else

console.info(result.data());
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Сделки::crm.deal.contact.items

```
crm.deal.contact.items.get(id)
```

Возвращает набор контактов, связанных с указанной сделкой.

Параметры

Параметр	Описание
id	Идентификатор сделки.

Результат возвращается в виде массива объектов, каждый из которых содержит следующие поля:

Поле	Описание
CONTACT_ID	Идентификатор контакта
SORT	Индекс сортировки
ROLE_ID	Идентификатор роли (зарезервировано)
IS_PRIMARY	Флаг первичного контакта

Пример вызова

```
var id = prompt("Введите ID");
BX24.callMethod(
"crm.deal.contact.items.get",
{
id: id
},
function(result)
{
if(result.error())
console.error(result.error());
else
console.dir(result.data());
}
);
```

Сделки::crm.deal.contact.items

```
crm.deal.contact.items.set(id, items)
```

Устанавливает набор контактов, связанных с указанной сделкой.

Параметры

Параметр	Описание
id	Идентификатор сделки.
items	Набор контактов в виде массива объектов со следующими полями: ▪ CONTACT_ID - идентификатор контакта (обязательное поле) ▪ SORT - индекс сортировки ▪ IS_PRIMARY - флаг первичного контакта

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Сделки::crm.deal.delete

```
crm.deal.delete(id)
```

Удаляет сделку и все связанные с ней объекты.

Параметры

Параметр	Описание
id	Идентификатор сделки.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(
                        "crm.deal.delete",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
```

```
console.info(result.data());
                            }
                    );
```

Сделки::crm.deal.fields

Описание и пример

```
crm.deal.fields()
```

Возвращает описание полей сделки, в том числе пользовательских.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(
                        "crm.deal.fields",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
```

```
console.dir(result.data());
                            }
                    );
```

Поля

Поле	Описание	Тип	Пр
ADDITIONAL_INFO	Дополнительная информация	string	
ASSIGNED_BY_ID	Связано с пользователем по ID	user	
BANK_DETAIL_ID	ID банковского реквизита	integer	Принима возвраща Параметр функцию crm.requ автомати успешно обновлен идентиф сделки
BEGINDATE	Дата начала	date	
CATEGORY_ID	Идентификатор направления	crm_category	Неизмен передава создании создастс направле
CLOSED	Завершена	char	
CLOSEDATE	Дата завершения	date	
COMMENTS	Коментарии	string	

COMPANY_ID	Идентификатор привязанной компании	crm_company	
CONTACT_ID	Идентификатор привязанного контакта	crm_contact	Устаревш для совм
CONTACT_IDS	Идентификатор привязанного контакта	crm_contact	Множест При испо crm.deal. crm.deal. массив к crm.deal. поля нет использо crm.deal. для полу контакто Для очис использу crm.deal. для заме использу crm.deal.
CREATED_BY_ID	Создано пользователем	user	Только д
CURRENCY_ID	Идентификатор валюты сделки	crm_currency	
DATE_CREATE	Дата создания	datetime	Только д
DATE_MODIFY	Дата изменения	datetime	Только д
ID	Идентификатор сделки	integer	Только д
IS_NEW	Флаг новой сделки (т. е. сделка в первой стадии)	char	
IS_RECURRING	Флаг шаблона	char	

	регулярной сделки (если стоит Y, то это не сделка, а шаблон)		
IS_RETURN_CUSTOMER	Признак повторного лида	char	
LEAD_ID	Идентификатор привязанного лида	crm_lead	Только д
LOCATION_ID	Местоположение клиента	location	Служебн рекоменд использо
MODIFY_BY_ID	Идентификатор автора последнего изменения	user	Только д
MOVED_BY_ID	Идентификатор автора перемещения элемента на текущую стадию	user	Только д
MOVED_TIME	Дата перемещения элемента на текущую стадию	user	Только д
OPENED	Доступен для всех	char	
OPPORTUNITY	Сумма	double	
ORIGINATOR_ID	Идентификатор источника данных	string	Использу привязки источник
ORIGIN_ID	Идентификатор элемента в	string	Использу привязки источник

	источнике данных		
PROBABILITY	Вероятность	integer	
QUOTE_ID	Идентификатор квоты	crm_quote	Только д Устаревш метод crm фильром
REQUISITE_ID	Идентификатор реквизита		Принима возвраща Параметр функцию crm.requ автомати успешно обновлен идентиф сделки.
STAGE_ID	Идентификатор стадии	crm_status	
STAGE_SEMANTIC_ID	Имя	string	
SOURCE_ID	Идентификатор источника. Определяет источник сделки (обратный звонок, реклама, электронная почта итд).	string	Список в идентиф вытащит crm.statu filter[E
SOURCE_DESCRIPTION	Дополнительно об источнике.	string	Текстово
TAX_VALUE	Ставка налога	double	
TITLE	Название	string	Обязател
TYPE_ID	Тип сделки	crm_status	Использу привязки

			источник
UTM_CAMPAIGN	Обозначение рекламной кампании	string	
UTM_CONTENT	Содержание кампании	string	Наприме объявлен
UTM_MEDIUM	Тип трафика	string	CPC (объ (баннерь
UTM_SOURCE	Рекламная система	string	Yandex-D Adwords
UTM_TERM	Условие поиска кампании	string	Наприме контекст

Сделки::crm.deal.get

```
crm.deal.get(id)
```

Возвращает сделку по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор сделки.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(
                        "crm.deal.get",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
```

```
console.dir(result.data());
                            }
                  );
```

Сделки::crm.deal.list

Возвращает список сделок по фильтру. Является реализацией списочного метода для сделок.

При выборке используйте маски:

- "*" - для выборки всех полей (без пользовательских и множественных)
- "UF_*"- для выборки всех пользовательских полей (без множественных)

Параметры

См. описание списочных методов.

Пример

Пример выводит данные в консоль. Если нужно вывести данные по другому, то реализуйте свою обработку данных возвращенных вызовами **result.data()** и **result.error()**.

```
        BX24.callMethod(
        "crm.deal.list",
        {
                order: { "STAGE_ID": "ASC"
},
                filter: { ">PROBABILITY": 50
},
                select: [ "ID", "TITLE",
"STAGE_ID", "PROBABILITY", "OPPORTUNITY",
```

```
"CURRENCY_ID" ]
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                {

console.dir(result.data());
                        if(result.more())

result.next();
                }
        }
);
```

Сделки::crm.deal.productrows

```
crm.deal.productrows.get(id)
```

Возвращает товарные позиции сделки.

Параметры

Параметр	Описание
id	Идентификатор сделки.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(

"crm.deal.productrows.get",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
```

```
                                        else

console.dir(result.data());
                                }
                        );
```

Сделки::crm.deal.productrows

```
crm.deal.productrows.set(id, rows)
```

Устанавливает (создаёт или обновляет) товарные позиции сделки.

Параметры

Параметр	Описание
id	Идентификатор сделки.
rows	Товарные позиции - массив вида array(array("поле"=>"значение"[, ...])[, ...]), где "поле" может принимать значения из возвращаемых методом crm.productrow.fields. Товарные позиции сделки, существующие до момента вызова метода, будут заменены новыми. После сохранения будет произведён пересчёт суммы сделки.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
```

```
"crm.deal.productrows.set",
                        {
                                id: id,
                                rows:
                                [
                                        {
"PRODUCT_ID": 689, "PRICE": 100.00,
"QUANTITY": 4 },
                                        {
"PRODUCT_ID": 690, "PRICE": 400.00,
"QUANTITY": 1 }
                                ]
                        },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else

console.info(result.data());
                        }
                );
```

Сделки::crm.deal.recurring.add

```
crm.deal.recurring.add(fields)
```

Добавляет новую настройку для регулярной сделки.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array ("поле"=>"значение"[, ...]), содержащий значения полей сделки. Обязательное поле - поле DEAL_ID [ID сделки, у которой задан параметр IS_RECURRING=Y] **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.deal.recurring.fields и посмотрите формат пришедших значений этих полей.

Пример

```
var current = new Date();
var nextMonth = new Date();
var nextYear = new Date();
nextMonth.setMonth(current.getMonth() +
```

```
1);
   nextYear.setYear(current.getFullYear() +
1);
   var date2str = function(d)
   {
        return d.getFullYear() + '-' +
paddatepart(1 + d.getMonth()) + '-' +
paddatepart(d.getDate()) + 'T' +
paddatepart(d.getHours()) + ':' +
paddatepart(d.getMinutes()) + ':' +
paddatepart(d.getSeconds()) + '+03:00';
   };
   var paddatepart = function(part)
   {
        return part >= 10 ? part.toString()
: '0' + part.toString();
   };

   BX24.callMethod(
       "crm.deal.recurring.add",
       {
           fields:
           {
               "DEAL_ID": "45",
               "CATEGORY_ID": "1",
               "IS_LIMIT": "D",
               "LIMIT_DATE":
date2str(nextYear),
               "START_DATE":
date2str(nextMonth),
               "PARAMS": {
                   "MODE": "multiple",
                   "MULTIPLE_TYPE": "month",
                   "MULTIPLE_INTERVAL": 1,
                   "OFFSET_BEGINDATE_TYPE":
"day",
                   "OFFSET_BEGINDATE_VALUE":
```

```
1,
                    "OFFSET_CLOSEDATE_TYPE":
"month",
                    "OFFSET_CLOSEDATE_VALUE":
2,
                }
            }
        },
        function(result)
        {
            if(result.error())

console.error(result.error());
            else
                console.info("Добавлены
настройки регулярной сделки. ID записи - " +
result.data());
        }
    );
```

Сделки::crm.deal.recurring.del

```
crm.deal.recurring.delete(id)
```

Удаляет существующую настройку для шаблона регулярной сделки.

Параметры

Параметр	Описание
id	Идентификатор настройки шаблона регулярной сделки.

Пример:

```
    var id = prompt("Введите ID");
    BX24.callMethod(
        "crm.deal.recurring.delete",
        { id: id },
        function(result)
        {
            if(result.error())

console.error(result.error());
```

```
            else
                console.info(result.data());
        }
    );
```

Сделки::crm.deal.recurring.exp

```
crm.deal.recurring.expose(id)
```

Создает новую сделку из шаблона.

Параметры

Параметр	Описание
id	Идентификатор настройки шаблона регулярной сделки.

Пример:

```
var id = prompt("Введите ID");
BX24.callMethod(
    "crm.deal.recurring.expose",
    {
        id: id,
    },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error());
            else
            {
                console.info(result.data());
            }
        }
    );
```

Сделки::crm.deal.recurring.fiel

Описание и пример

```
crm.deal.recurring.fields()
```

Возвращает список полей настройки шаблона регулярной сделки с описанием.

Параметры

Без параметров.

Пример

```
    BX24.callMethod(
       "crm.deal.recurring.fields",
       {},
       function(result)
       {
           if(result.error())

console.error(result.error());
           else
               console.dir(result.data());
```

```
            }
    );
```

Поля

Поле	Описание	T
ID	Идентификатор записи в таблице настроек регулярной сделки	in
DEAL_ID	ID шаблона сделки	in
BASED_ID	ID сделки, на основании которой был создан шаблон	in
ACTIVE	Флаг активности. Значения: Y/N	c
NEXT_EXECUTION	Дата следующего создания новой сделки из шаблона. Рассчитывается системой по указанным параметрам. Если значение пустое, новые сделки не создаются	d
LAST_EXECUTION	Дата последнего создания новой сделки из шаблона	d
COUNTER_REPEAT	Количество созданных из шаблона сделок	in
START_DATE	Дата начала отсчета при расчете даты следующего создания новой сделки. Если значение пустое, рассчитывается от текущей даты	d
CATEGORY_ID	Категория, которая будет задана у сделки, созданной из шаблона	c
IS_LIMIT	Есть ли ограничения по созданию новых сделок. Значения: N - без ограничений, D - установлено ограничение по дате, T -	c

	установлено ограничение по количеству новых сделок	
LIMIT_REPEAT	Максимальное число сделок, которое можно создать из этого шаблона	i
LIMIT_DATE	Дата, до достижения которой можно создавать сделки из этого шаблона	d
PARAMS	Набор параметров для расчета - recurring_params: ▪ **MODE** - режим повторения: ▪ single - единичный (будет создана одна сделка из шаблона, смещение рассчитывается до значения START_DATE); ▪ multiple - множественный ▪ **MULTIPLE_TYPE** - тип повторения в множественном режиме [MODE равен multiple]: ▪ day - день ▪ week - неделя ▪ month - месяц ▪ year - год ▪ **MULTIPLE_INTERVAL** - значение смещения [MODE равен multiple] ▪ **SINGLE_BEFORE_START_DATE_TYPE** - тип смещения до даты начала [MODE равен single]: ▪ day - день ▪ week - неделя ▪ month - месяц ▪ year - год ▪ **SINGLE_BEFORE_START_DATE_VALUE** - значение смещения до даты начала, если не установлено - смещения нет [MODE равен single]	

- **OFFSET_BEGINDATE_TYPE** - тип смещения для расчета поля "даты начала сделки", расчет ведется от момента создания новой сделки из шаблона:
  - day - день
  - week - неделя
  - month - месяц
  - year - год
- **OFFSET_BEGINDATE_VALUE** - значение смещения для расчета поля "даты начала сделки", расчет ведется от момента создания новой сделки из шаблона
- **OFFSET_CLOSEDATE_TYPE** - значение смещения для расчета поля "даты завершения сделки", расчет ведется от момента создания новой сделки из шаблона:
  - day - день
  - week - неделя
  - month - месяц
  - year - год
- **OFFSET_CLOSEDATE_VALUE** - значение смещения для расчета поля "даты завершения сделки", расчет ведется от момента создания новой сделки из шаблона

Сделки::crm.deal.recurring.get

```
crm.deal.recurring.get(id)
```

Возвращает поля настройки шаблона регулярной сделки по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор настройки шаблона регулярной сделки.

Пример:

```
    var id = prompt("Введите ID");
    BX24.callMethod(
        "crm.deal.recurring.get",
        { id: id },
        function(result)
        {
            if(result.error())

console.error(result.error());
```

```
            else
                console.dir(result.data());
        }
    );
```

Сделки::crm.deal.recurring.list

```
crm.deal.recurring.list ()
```

Возвращает список настроек шаблонов регулярных сделок по фильтру.

При выборке используйте маску "*" для выборки всех полей (без пользовательских и множественных).

Параметры

См. описание списочных методов.

Пример:

```
   BX24.callMethod(
       "crm.deal.recurring.list",
       {
           order: { "DEAL_ID": "ASC" },
           filter: { ">COUNTER_REPEAT": 5 },
           select: [ "ID", "DEAL_ID ",
"NEXT_EXECUTION", "LAST_EXECUTION",
"CATEGORY_ID", "IS_LIMIT" ]
       },
       function(result)
       {
```

```
            if(result.error())

console.error(result.error());
            else
            {
                console.dir(result.data());
                if(result.more())
                    result.next();
            }
        }
    );
```

Сделки::crm.deal.recurring.up

```
crm.deal.recurring.update(id, fields)
```

Обновляет существующую настройку для шаблона регулярной сделки.

Параметры

Параметр	Описание
id	Идентификатор настройки шаблона регулярной сделки.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.deal.recurring.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.deal.recurring.fields и посмотрите формат пришедших значений этих полей.

Пример:

```
   var current = new Date();
   var nextYear = new Date();
   nextYear.setYear(current.getFullYear() +
1);
   var date2str = function(d)
   {
        return d.getFullYear() + '-' +
paddatepart(1 + d.getMonth()) + '-' +
paddatepart(d.getDate()) + 'T' +
paddatepart(d.getHours()) + ':' +
paddatepart(d.getMinutes()) + ':' +
paddatepart(d.getSeconds()) + '+03:00';
   };
   var paddatepart = function(part)
   {
        return part >= 10 ? part.toString()
: '0' + part.toString();
   };

   var id = prompt("Введите ID");
   BX24.callMethod(
       "crm.deal.recurring.update",
       {
           id: id,
           fields:
           {
               "CATEGORY_ID": "2",
               "START_DATE":
date2str(nextMonth),
               "PARAMS": {
                   "MODE": "single",

"SINGLE_BEFORE_START_DATE_TYPE": "day",

"SINGLE_BEFORE_START_DATE_VALUE": 5,
                   "OFFSET_BEGINDATE_TYPE":
"day",
```

```
                    "OFFSET_BEGINDATE_VALUE":
1,
                    "OFFSET_CLOSEDATE_TYPE":
"month",
                    "OFFSET_CLOSEDATE_VALUE":
2,
                }
            },
        },
        function(result)
        {
            if(result.error())

console.error(result.error());
            else
            {
                console.info(result.data());
            }
        }
    );
```

Сделки::crm.deal.update

Описание

```
crm.deal.update(id, fields, params)
```

Обновляет существующую сделку.

Параметры

Параметр	Описание
id	Идентификатор сделки.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.deal.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.deal.fields и посмотрите формат пришедших значений этих полей.
params	Набор параметров. REGISTER_SONET_EVENT - произвести регистрацию события изменения сделки в живой ленте. Дополнительно будет отправлено уведомление ответственному за сделку.

Пример

```
        var id = prompt("Введите ID");
BX24.callMethod(
        "crm.deal.update",
        {
                id: id,
                fields:
                {
                        "STAGE_ID":
"NEGOTIATION",
                        "PROBABILITY": 70
                },
                params: {
"REGISTER_SONET_EVENT": "Y" }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                {

console.info(result.data());
                }
        }
);
```

Пояснения к методу

Для управления контактами сделки рекомендуется использовать множественное поле CONTACT_IDS:

Пример

```
BX24.callMethod("crm.deal.update", { id: 1,
fields: { "CONTACT_IDS": [ 1, 2, 3 ] } });
```

В результате сделка будет связана с тремя указанными контактами.

Поле CONTACT_ID является устаревшим и поддерживается для обеспечения обратной совместимости.

Пример

```
BX24.callMethod("crm.deal.update", { id: 1,
fields: { "CONTACT_ID": 4 } });
```

В результате этого вызова в сделку будет добавлена связь с указанным контактом.

Пожалуйста, обратите **внимание**, что уже существующие связи связи с контактами при этом удалены не будут. То есть если сделка до этого была связана с контактами 1, 2 и 3, то в результате она будет связана с контактами 1, 2, 3 и 4.

Сделки::crm.deal.userfield.add

```
crm.deal.userfield.add(fields)
```

Создаёт новое пользовательское поле для сделок.

Системное ограничение на название поля - 20 знаков. К названию пользовательского поля всегда добавляется префикс UF_CRM_, то есть реальная длина названия - 13 знаков.

Параметры

Параметр	Описание
fields	Набор [dw]полей[/dw][di]На данный момент: ENTITY_ID USER_TYPE_ID FIELD_NAME LIST_FILTER_LABEL LIST_COLUMN_LABEL EDIT_FORM_LABEL ERROR_MESSAGE HELP_MESSAGE MULTIPLE MANDATORY SHOW_FILTER SETTINGS LIST[/di] - массив вида array("поле"=>"значение"[, ...]), содержащий описание пользовательского поля. В том числе содержит поле `LIST`, которое содержит набор значений списка для

	пользовательских полей типа Список. Указывается при создании/обновлении поля. Каждое значение представляет собой массив с полями: ▪ **VALUE** - значение элемента списка. Поле является обязательным в случае, когда создается новый элемент. ▪ **SORT** - сортировка. ▪ **DEF** - если равно Y, то элемент списка является значением по-умолчанию. Для множественного поля допустимо несколько DEF=Y. Для не множественного, дефолтным будет считаться первое. ▪ **XML_ID** - внешний код значения. Параметр учитывается только при обновлении уже существующих значений элемента списка. ▪ **ID** - идентификатор значения. Если он указан, то считается что это обновление существующего значения элемента списка, а не создание нового. Имеет смысл только при вызове методов `*.userfield.update`. ▪ **DEL** - если равно Y, то существующий элемент списка будет удален. Применяется, если заполнен параметр ID.

Полное описание полей можно получить вызовом метода crm.userfield.fields.

Пример #1

```
BX24.callMethod(
"crm.deal.userfield.add",
{
        fields:
        {
```

```
                        "FIELD_NAME":
"MY_STRING",
                        "EDIT_FORM_LABEL":
"Моя строка",
                        "LIST_COLUMN_LABEL":
"Моя строка",
                        "USER_TYPE_ID":
"string",
                        "XML_ID":
"MY_STRING",
                        "SETTINGS": {
"DEFAULT_VALUE": "Привет, мир!" }
                }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

Пример #2

```
BX24.callMethod(
        "crm.deal.userfield.add",
        {
                fields:
                {
                        "FIELD_NAME":
"MY_LIST",
```

```
                            "EDIT_FORM_LABEL":
"Мой список",
                            "LIST_COLUMN_LABEL":
"Мой список",
                            "USER_TYPE_ID":
"enumeration",
                            "LIST": [ { "VALUE":
"Элемент #1" },
                          { "VALUE": "Элемент
#2" },
                          { "VALUE": "Элемент
#3" },
                          { "VALUE": "Элемент
#4" },
                          { "VALUE": "Элемент
#5" } ],
                            "XML_ID": "MY_LIST",
                            "SETTINGS": {
"LIST_HEIGHT": 3 }
                  }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс»

Сделки::crm.deal.userfield.del

```
crm.deal.userfield.delete(id)
```

Удаляет пользовательское поле сделок.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.deal.userfield.delete",
        {
                id: id
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
```

```
                    else

console.info(result.data());
        }
);
```

Сделки::crm.deal.userfield.get

```
crm.deal.userfield.get(id)
```

Возвращает пользовательское поле сделок по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор поля.

Пример

```
        var id = prompt("Введите ID");
BX24.callMethod(
        "crm.deal.userfield.get",
        {
                id: id
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
```

```
                    else

console.dir(result.data());
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Сделки::crm.deal.userfield.list

```
crm.deal.userfield.list(order, filter)
```

Возвращает список пользовательских полей сделок по фильтру.

Параметры

Параметр	Описание
order	Поля сортировки.
filter	Поля фильтра.

Пример

```
BX24.callMethod(
        "crm.deal.userfield.list",
        {
                order: { "SORT": "ASC" },
                filter: { "MANDATORY": "N" }
        },
        function(result)
        {
                if(result.error())
```

```
console.error(result.error());
                else
                {

console.dir(result.data());
                        if(result.more())

result.next();
                }
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Сделки::crm.deal.userfield.upd

```
crm.deal.userfield.update(id, fields)
```

Обновляет существующее пользовательское поле сделок.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.userfield.fields.
LIST	Содержит набор значений списка для пользовательских полей типа Список. Указывается при создании/обновлении поля. Каждое значение представляет собой массив с полями: ▪ **VALUE** - значение элемента списка. Поле является обязательным в случае, когда создается новый элемент. ▪ **SORT** - сортировка. ▪ **DEF** - если равно Y, то элемент списка является значением по-умолчанию. Для множественного поля допустимо несколько DEF=Y. Для не

множественного, дефолтным будет считаться первое.

- **XML_ID** - внешний код значения. Параметр учитывается только при обновлении уже существующих значений элемента списка.
- **ID** - идентификатор значения. Если он указан, то считается что это обновление существующего значения элемента списка, а не создание нового. Имеет смысл только при вызове методов `*.userfield.update`.
- **DEL** - если равно Y, то существующий элемент списка будет удален. Применяется, если заполнен параметр ID.

Пример

```
var id = prompt("Введите ID");
var label = prompt("Введите новое
название");
BX24.callMethod(
        "crm.deal.userfield.update",
        {
                id: id,
                fields:
                {
                        "EDIT_FORM_LABEL":
label,
                        "LIST_COLUMN_LABEL":
label
                }
        },
        function(result)
        {
                if(result.error())
```

```
console.error(result.error());
                else
                {

console.dir(result.data());
                        if(result.more())

result.next();
                }
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmDealAdd

Событие, вызываемое при создании сделки.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора созданной сделки.

Пример

Подписаться на метод через рест можно так:

```
CRest::call('event.bind',
    [
        'event' => 'onCrmDealAdd',
        'handler' =>
'https://example.com/handler.php'
    ]);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmDealDelete

Событие, вызываемое при удалении сделки.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора удаленной сделки.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmDealUpdate

Событие, вызываемое при обновлении сделки.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора обновленной сделки.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmDealUserFieldAdd

Событие, вызываемое при добавлении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmDealUserFieldDelete

Событие, вызываемое при удалении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmDealUserFieldSetEnumVa

Событие, вызываемое при изменении набора значений для пользовательского поля списочного типа.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmDealUserFieldUpdate

Событие, вызываемое при изменении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmDealRecurringAdd

Событие, вызываемое при создании новой регулярной сделки.

Параметры

Параметр	Описание
FIELDS	Массив содержит следующие поля: ▪ **ID** - значение идентификатора записи в таблице настроек регулярной сделки; ▪ **RECURRING_DEAL_ID** - значение идентификатора шаблона регулярной сделки.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmDealRecurringDelete

Событие, вызываемое при удалении регулярной сделки.

Параметры

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора записи в таблице настроек регулярной сделки.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmDealRecurringExpose

Событие, вызываемое при создании новой сделки из регулярной сделки.

Параметры

Параметр	Описание
FIELDS	Массив содержит следующие поля: ▪ **ID** - значение идентификатора записи в таблице настроек регулярной сделки; ▪ **RECURRING_DEAL_ID** - значение идентификатора шаблона регулярной сделки; ▪ **DEAL_ID** - значение ID новой сделки, созданной на основе регулярной.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmDealRecurringUpdate

Событие, вызываемое при обновлении регулярной сделки.

Параметры

Параметр	Описание
FIELDS	Массив содержит следующие поля: ▪ **ID** - значение идентификатора записи в таблице настроек регулярной сделки; ▪ **RECURRING_DEAL_ID** - значение идентификатора шаблона регулярной сделки.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Направления

Методы из семейства `crm.category.*` позволяют универсально управлять направлениями любых сущностей CRM, которые их поддерживают (на данный момент — сделки и смарт-процессы).

Во все методы необходимо передавать как параметр идентификатор типа сущности CRM (`entityTypeId`). Так как у каждого типа сущностей могут иметься свои особенности по работе с направлениями, данный параметр необходим для определения, с каким конкретно типом мы работаем.

Права доступа

При обращении к методам REST учитываются права доступа пользователя, от которого осуществляется вызов методов.

Булевы значения

Значения некоторых полей имеют булевый тип (например, поле `isDefault`).

В этом случае для изменения значения надо передавать `"Y"` или `"N"`. В ответах на запросы они будут отображаться аналогично.

Особенности работы с различными типами сущности

Как отмечалось выше, в зависимости от типа сущности может незначительно отличаться работа методов.

При работе с направлениями **сделки невозможно изменить**, какое направление будет **направлением по умолчанию**. Соответственно, поле `isDefault` для направлений сделки - read-only (о чем можно узнать через crm.category.fields). Удалить направление сделки по умолчанию тоже нельзя.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.category.add

```
crm.category.add({entityTypeId: number,
fields: {}})
```

Метод создаст у типа сущности с идентификатором `entityTypeId` новое направление с полями `fields`.

Метод вернет результат аналогичный вызову метода crm.category.get на только что созданном направлении.

Параметры

Параметр	Описание
entityTypeId	Идентификатор типа сущности CRM.
fields	Значение полей направления.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.category.delete

```
crm.category.delete({entityTypeId: number,
id: number})
```

Метод удалит направление с идентификатором `id`, которое относится к типу сущности с идентификатором `entityTypeId`.

Если это направление по умолчанию, то будет возвращена ошибка.

Если на этом направлении есть хотя бы один элемент, то оно не может быть удалено.

Параметры

Параметр	Описание
entityTypeId	Идентификатор типа сущности CRM.
id	Идентификатор направления.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.category.fields

```
crm.category.fields({entityTypeId: number})
```

Отдает информацию о полях направлений. Названия полей для входных данных следует брать из этого метода.

Параметры

Параметр	Описание
entityTypeId	Идентификатор типа сущности CRM.

Ответ:

```
{
  "fields": {
    "id": {
      "type": "integer",
      "isRequired": false,
      "isReadOnly": true,
      "isImmutable": false,
      "isMultiple": false,
      "isDynamic": false,
      "title": "ID"
```

```
    },
    "name": {
      "type": "string",
      "isRequired": false,
      "isReadOnly": false,
      "isImmutable": false,
      "isMultiple": false,
      "isDynamic": false,
      "title": "NAME"
    },
    "sort": {
      "type": "integer",
      "isRequired": false,
      "isReadOnly": false,
      "isImmutable": false,
      "isMultiple": false,
      "isDynamic": false,
      "title": "SORT"
    },
    "entityTypeId": {
      "type": "integer",
      "isRequired": true,
      "isReadOnly": false,
      "isImmutable": false,
      "isMultiple": false,
      "isDynamic": false,
      "title": "ENTITY_TYPE_ID"
    },
    "isDefault": {
      "type": "boolean",
      "isRequired": false,
      "isReadOnly": false,
      "isImmutable": false,
      "isMultiple": false,
      "isDynamic": false,
      "title": "IS_DEFAULT"
    }
```

```
    }
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.category.get

```
crm.category.get({entityTypeId: number, id:
number})
```

Отдает информацию о направлении с идентификатором `id`.

Параметры

Параметр	Описание
entityTypeId	Идентификатор типа сущности CRM.
id	Идентификатор направления.

Ответ:


```
{
        "category": {
                "id": 53,
                "name": "Общее",
                "sort": 500,
                "entityTypeId": 170,
```

```
                    "isDefault": "N"
            }
}
```

Здесь:

- `id` - идентификатор направления.
- `name` - название направления.
- `sort` - коэффициент сортировки.
- `entityTypeId` - идентификатор типа сущности CRM, к которому относится данное направление.
- `isDefault` - является ли данное направление направлением по умолчанию.

У некоторых сущностей ответ может быть расширен дополнительными полями.

crm.category.list

```
crm.category.list({entityTypeId: number})
```

Метод вернет массив направлений, которые относятся к типу сущности с идентификатором `entityTypeId`.

Параметры

Параметр	Описание
entityTypeId	Идентификатор типа сущности CRM.

```
{
        "categories": []
}
```

где каждый элемент массива — это структура, аналогичная результату метода crm.category.get.

crm.category.update

```
crm.category.update({entityTypeId: number,
id: number, fields: {}})
```

Метод обновит направление с идентификатором `id`, задав ему новые значения полей из `fields`. Если какое-то поле будет отсутствовать в `fields`, то его значение останется неизменным.

Метод вернет результат аналогичный вызову метода crm.category.get для изменённого направления.

Параметры

Параметр	Описание
entityTypeId	Идентификатор типа сущности CRM.
id	Идентификатор направления.
fields	Значение полей направления.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Направления сделок, устаревшее::crm.dealcategory

```
crm.dealcategory.add(fields)
```

Создаёт новое направление сделок.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.dealcategory.fields (кроме полей, помеченных атрибутом isReadOnly). **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.dealcategory.fields и посмотрите формат пришедших значений этих полей.

Пример

```
    "crm.dealcategory.add",
    {
        fields:
        {
            "NAME": "Новое направление",
            "SORT": "20"
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.info("Создано
направление с ID " + result.data());
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Направления сделок, устаревшее::crm.dealcategory

```
crm.dealcategory.default.get(
)
```

Метод чтения настроек общего направления сделок.

Параметры

Без параметров

Примеры

```
BX24.callMethod(
           "crm.dealcategory.default.get",
           {},
           function(result)
           {
               if(result.error())

console.error(result.error());
               else
```

```
console.dir(result.data());
            }
        );
```

Результат:

```
{ID: 0, NAME: "Общее"}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Направления сделок, устаревшее::crm.dealcategory

```
crm.dealcategory.default.set(
   name
)
```

Метод записи настроек общего направления сделок.

Параметры

Параметр	Описание	С версии
name	Название общего направления	

Примеры

```
BX24.callMethod(
          "crm.dealcategory.default.set",
        { "name": "" },
        function(result)
        {
```

```
                if(result.error())

console.error(result.error());
                else
                    console.dir(result.data());
            }
        );
```

Направления сделок, устаревшее::crm.dealcategory

```
crm.dealcategory.delete(id)
```

Удаляет направление сделок.

Параметры

Параметр	Описание
id	Идентификатор направления.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.dealcategory.delete",
        { id: id },
        function(result)
        {
                if(result.error())
```

```
console.error(result.error());
                else

console.info(result.data());
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Направления сделок, устаревшее::crm.dealcategory

```
crm.dealcategory.fields
```

Возвращает описание полей направления сделок.

Параметры

Без параметров

Пример

```
        BX24.callMethod(
        "crm.dealcategory.fields",
        {},
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
```

```
            }
);
```

Поля

Поле	Описание	Тип	Примечание
CREATED_DATE	Дата создания	datetime	Только для чтения
ID	Идентификатор направления сделки	integer	Только для чтения
IS_LOCKED	Заблокировано	char	Только для чтения
NAME	Название направления	string	Обязательное
SORT	Сортировка	integer	

Направления сделок, устаревшее::crm.dealcategory

```
crm.dealcategory.get(id)
```

Возвращает направление сделок по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор направления.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.dealcategory.get",
        { id: id },
        function(result)
        {
                if(result.error())
```

```
console.error(result.error());
                else

console.dir(result.data());
        }
);
```

Направления сделок, устаревшее::crm.dealcategory

```
crm.dealcategory.list
```

Возвращает список направлений сделок по фильтру. Является реализацией списочного метода для направлений сделок.

Параметры

Без параметров

Пример

```
BX24.callMethod(
        "crm.dealcategory.list",
        {
                order: { "SORT": "ASC" },
                filter: { "IS_LOCKED": "N"
},
                select: [ "ID", "NAME",
"SORT" ]
        },
        function(result)
        {
```

```
                if(result.error())

console.error(result.error());
                else
                {

console.dir(result.data());
                        if(result.more())

result.next();
                }
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Направления сделок, устаревшее::crm.dealcategory

```
crm.dealcategory.stage.list(id)
```

Возвращает список стадий сделок для направления по идентификатору. Равносилен вызову метода crm.status.list с параметром ENTITY_ID равным результату вызова crm.dealcategory.status.

Параметры

Параметр	Описание
id	Идентификатор направления. Если указать id = 0 или ничего [dw]не указывать[/dw] [di]Пустые кавычки или совсем не передавать параметр.[/di], то вернёт статусы "дефолтного" направления. Если указать id > 0 [dw]не существующего направления[/dw][di]Например, указываем id = 10, но направления с id=10 нет в системе. [/di], ничего не возвращает.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.dealcategory.stage.list",
        { id: id },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Направления сделок, устаревшее::crm.dealcategory

```
crm.dealcategory.status(id)
```

Возвращает идентификатор типа справочника для хранения стадий по идентификатору направления сделок.

Это строка вида DEAL_STAGE_[Идентификатор направления]. Например, для направления с идентификатором 1 будет возвращена строка "DEAL_STAGE_1".
Идентификатор предназначен для работы с семейством методов crm.status.*. Например, чтобы создать новую стадию для направления, нужно передать его в метод crm.status.add в качестве параметра ENTITY_ID

Параметры

Параметр	Описание
id	Идентификатор направления.

Пример

```
BX24.callMethod(
        "crm.dealcategory.status",
        { id: id },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

Направления сделок, устаревшее::crm.dealcategory

```
crm.dealcategory.update(id, fields)
```

Обновляет существующее направление.

Параметры

Параметр	Описание
id	Идентификатор направления.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.dealcategory.fields (кроме полей помеченных атрибутами isReadOnly и isImmutable). **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.dealcategory.fields и посмотрите формат пришедших значений этих полей.

Пример

```
var id = prompt("Введите ID");
var sort = prompt("Введите сортировку");
sort = parseInt(sort);
if(isNaN(sort) || sort < 0)
{
        sort = 0;
}

BX24.callMethod(
    "crm.dealcategory.update",
    {
        id: id,
        fields: { "SORT": sort }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
        {
            console.info(result.data());
        }
    }
);
```

crm.invoice.status.add

С версии 19.0.0 рекомендуется использовать метод crm.status.add

```
crm.invoice.status.add(fields)
```

Создаёт новый статус счёта.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей статуса счёта. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.invoice.status.fields и посмотрите формат пришедших значений этих полей.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.invoice.status.delete

С версии 19.0.0 рекомендуется использовать метод crm.status.delete

```
crm.invoice.status.delete(id)
```

Удаляет статус счёта.

Параметры

Параметр	Описание
id	Идентификатор статуса счёта.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.invoice.status.fields

С версии 19.0.0 рекомендуется использовать метод crm.status.fields

```
crm.invoice.status.fields()
```

Возвращает описание полей статуса счёта.

Параметры

Без параметров.

Поля

Поле	Описание	Тип	Примечание
ENTITY_ID	Идентификатор сущности, связанной с счётом	string	Только для чтения
ID	Идентификатор счёта	integer	Только для чтения
NAME	Название раздела	string	Обязательное

NAME_INIT		string	Только для чтения
SORT	Сортировка	integer	Обязательный
STATUS_ID	Статус	string	Только для чтения
SYSTEM	Системный или нет	char	Только для чтения

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.invoice.status.get

С версии 19.0.0 рекомендуется использовать метод crm.status.get

```
crm.invoice.status.get(id)
```

Возвращает статус счёта по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор статуса счёта.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.invoice.status.list

С версии 19.0.0 рекомендуется использовать метод crm.status.list

```
crm.invoice.status.list()
```

Возвращает список статусов счёта по фильтру. Является реализацией списочного метода для статусов счёта.

Параметры

См. описание списочных методов.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.invoice.status.update

С версии 19.0.0 рекомендуется использовать метод crm.status.update

```
crm.invoice.status.update(id, fields)
```

Обновляет существующий статус счёта.

Параметры

Параметр	Описание
id	Идентификатор статуса счёта.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.invoice.status.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.invoice.status.fields и посмотрите формат пришедших значений этих полей.

crm.quote.add

```
crm.quote.add(fields)
```

Создаёт новое коммерческое предложение. Если необходимо в предложении указать какие реквизиты покупателя/продавца (т.к. их может быть несколько у компании), то используйте метод crm.requisite.link.register.

В создаваемом предложении обязательно должны быть указаны компании продавца и покупателя:
`COMPANY_ID`, если покупатель - компания или `CONTACT_ID`, если покупатель - контакт.
`MYCOMPANY_ID` - продавец.

Идентификаторы указанные в **crm.requisite.link.register** и в создаваемом предложении должны соответствовать покупателю и продавцу.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей предложения, где "поле" может принимать значения из возвращаемых методом crm.quote.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод

	crm.quote.fields и посмотрите формат пришедших значений этих полей.

Пример

```
BX24.callMethod(
        "crm.quote.add",
        {
                fields:
                {
                        "TITLE": "Черновик",
                        "STATUS_ID":
"DRAFT",
                        "OPENED": "Y",
                        "ASSIGNED_BY_ID": 1,
                        "CURRENCY_ID":
"USD",
                        "OPPORTUNITY": 5000,
                        "COMPANY_ID": 1,
                        "COMMENTS": "Новое
коммерческое предложение.",
                        "BEGINDATE": "2016-
03-01T12:00:00",
                        "CLOSEDATE": "2016-
04-01T12:00:00"
                }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
```

```
console.info("Создано предложение с ID " +
result.data());
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.quote.delete

```
crm.quote.delete(id)
```

Удаляет предложение и все связанные с ним объекты.

Параметры

Параметр	Описание
id	Идентификатор предложения.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.quote.delete",
        { id: id },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
```

```
console.info(result.data());
        }
);
```

crm.quote.fields

```
crm.quote.fields()
```

Возвращает описание полей коммерческого предложения, в том числе пользовательских.

Параметры

Без параметров.

Пример

```
BX24.callMethod(
        "crm.quote.fields",
        {},
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
```

```
            }
);
```

Поля

Поле	Описание	Тип	Примеч
ASSIGNED_BY_ID	Связано с пользователем по ID	user	
BEGINDATA	Дата выставления	date	
CLIENT_ADDR	Адрес контакта	string	
CLIENT_CONTACT	Контакт	string	Устаревший Сохраняетс совместимо
CLIENT_EMAIL	Адрес электронной почты контакта	string	Устаревший Сохраняетс совместимо
CLIENT_PHONE	Проверка заполненности поля телефон	char	Только для чтения.Уста Сохраняетс совместимо
CLIENT_TITLE	Название клиента	string	Устаревший Сохраняетс совместимо
CLIENT_TPA_ID	КПП клиента	string	Устаревший Сохраняетс совместимо
CLIENT_TP_ID	ИНН клиента	string	Устаревший Сохраняетс совместимо

CLOSED	Завершено	char	
CLOSEDATA	Дата завершения	date	
COMMENTS	Комментарий менеджера	string	
COMPANY_ID	Привязка к компании	crm_company	
CONTACT_ID	Привязка к контакту	crm_contact	Устаревший Сохраняетс совместимо
CONTACT_IDS	Привязка к нескольким контактам	crm_contact	Множестве
CONTENT	Содержание предложения	string	
CREATED_DY_ID	Кем создана	user	Только для
CURRENCY_ID	Валюта предложения	crm_currency	
DATE_CREATE	Дата создания	datetime	Только для
DATE_MODIFY	Дата изменения	datetime	Только для
DEAL_ID	Сделка, привязанная к предложению.	crm_deal	
ID	Идентификатор предложения	integer	Только для
LEAD_ID	Лид, привязанное к предложению	crm_lead	
LOCATION_ID	Месторасположение	location	
MODIFY_BY_ID	Идентификатор	user	Только для

	автора последнего изменения		
MYCOMPANY_ID	Идентификатор компании, от которой делается предложение	crm_company	
OPENED	Доступен для всех	char	
OPPORTUNITY	Сумма	double	
PERSON_TYPE_ID	Идентификатор типа плательщика	integer	
QUOTE_NUMBER	Номер предложения	string	Только для
STATUS_ID	Статус	crm_status	Статус из справочник
TAX_VALUE	Сумма	double	
TERMS	Условия	string	
TITLE	Название	string	Обязательн
UTM_CAMPAIGN	Обозначение рекламной кампании	string	
UTM_CONTENT	Содержание кампании	string	Например, контекстны объявлений
UTM_MEDIUM	Тип трафика	string	CPC (объяв CPM (банне
UTM_SOURSE	Рекламная система	string	Yandex-Dire Google-Adw другие.
UTM_TERM	Условие поиска кампании	string	Например, ключевые с

			контекстно рекламы.

crm.quote.get

```
crm.quote.get(id)
```

Возвращает коммерческое предложение по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор предложения.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.quote.get",
        { id: id },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
```

```
console.dir(result.data());
        }
);
```

crm.quote.list

Возвращает список предложений по фильтру. Является реализацией списочного метода для предложений.

Параметры

См. описание списочных методов.

Пример

```
BX24.callMethod(
        "crm.quote.list",
        {
                order: { "STATUS_ID": "ASC"
},
                filter: { "=COMPANY_ID": 1
},
                select: [ "ID", "TITLE",
"STATUS_ID", "OPPORTUNITY", "CURRENCY_ID" ]
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                {
```

```
console.dir(result.data());
                        if(result.more())

result.next();
                }
        }
);
```

crm.quote.productrows.get

```
crm.quote.productrows.get(id)
```

Возвращает товарные позиции предложения.

Параметры

Параметр	Описание
id	Идентификатор предложения.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.quote.productrows.get",
        { id: id },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
```

```
console.dir(result.data());
        }
);
```

crm.quote.productrows.set

```
crm.quote.productrows.set(id, rows)
```

Устанавливает (создаёт или обновляет) товарные позиции предложения.

Параметры

Параметр	Описание
id	Идентификатор предложения.
rows	Товарные позиции - массив вида array(array("поле"=>"значение"[, ...])[, ...]), где "поле" может принимать значения из возвращаемых методом crm.productrow.fields. Товарные позиции предложения, существующие до момента вызова метода, будут заменены новыми. После сохранения будет произведён пересчёт суммы предложения.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.quote.productrows.set",
        {
                id: id,
                rows:
                [
                        { "PRODUCT_ID": 1,
"PRICE": 100.00, "QUANTITY": 2 },
                        { "PRODUCT_ID": 2,
"PRICE": 200.00, "QUANTITY": 1 }
                ]
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.info(result.data());
        }
);
```

crm.quote.update

```
crm.quote.update(id, fields, params)
```

Обновляет существующее предложение.

Параметры

Параметр	Описание
id	Идентификатор предложения.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.quote.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.quote.fields и посмотрите формат пришедших значений этих полей.
params	Набор параметров. REGISTER_HISTORY_EVENT - создать запись в истории, значение по умолчанию: "Y". Дополнительно будет отправлено уведомление ответственному за предложение.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.quote.update",
        {
                id: id,
                fields: { "STATUS_ID":
"SENT" }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.info(result.data());
        }
);
```

crm.quote.userfield.add

```
crm.quote.userfield.add(fields)
```

Создаёт новое пользовательское поле для предложений.

Системное ограничение на название поля - 20 знаков. К названию пользовательского поля всегда добавляется префикс UF_CRM_, то есть реальная длина названия - 13 знаков.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий описание пользовательского поля. Полное описание полей можно получить вызовом метода crm.userfield.fields.

Пример

```
BX24.callMethod(
"crm.quote.userfield.add",
{
```

```
                fields:
                {
                        "FIELD_NAME":
"MY_STRING",
                        "EDIT_FORM_LABEL":
"Моя строка",
                        "LIST_COLUMN_LABEL":
"Моя строка",
                        "USER_TYPE_ID":
"string",
                        "XML_ID":
"MY_STRING",
                        "SETTINGS": {
"DEFAULT_VALUE": "Привет, мир!" }
                }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

crm.quote.userfield.delete

```
crm.quote.userfield.delete(id)
```

Удаляет пользовательское поле предложений.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.quote.userfield.delete",
        { id: id },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
```

```
console.info(result.data());
        }
);
```

crm.quote.userfield.get

```
crm.quote.userfield.get(id)
```

Возвращает пользовательское поле предложений по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.quote.userfield.get",
        { id: id },
        function(result)
        {
                if(result.error())

console.error(result.error());
```

```
                    else

console.dir(result.data());
        }
);
```

crm.quote.userfield.list

```
crm.quote.userfield.list(order, filter)
```

Возвращает список пользовательских полей предложений по фильтру.

Параметры

Параметр	Описание
order	Поля сортировки.
filter	Поля фильтра.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.quote.userfield.list",
        {
                order: { "SORT": "ASC" },
                filter: { "MANDATORY": "N" }
        },
```

```
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                {

console.dir(result.data());
                        if(result.more())

result.next();
                }
        }
);
```

crm.quote.userfield.update

```
crm.quote.userfield.update(id, fields)
```

Обновляет существующее пользовательское поле предложений.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.userfield.fields.

Пример

```
var id = prompt("Введите ID");
var label = prompt("Введите новое
название");
BX24.callMethod(
        "crm.quote.userfield.update",
```

```
        {
                id: id,
                fields:
                {
                        "EDIT_FORM_LABEL":
label,
                        "LIST_COLUMN_LABEL":
label
                }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                {

console.dir(result.data());
                        if(result.more())

result.next();
                }
        }
);
```

onCrmQuoteAdd

Событие, вызываемое при создании коммерческого предложения.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора созданного предложения.

© «Битрикс», 2001-2008, «1С-Битрикс» 2008-2022

onCrmQuoteDelete

Событие, вызываемое при удалении коммерческого предложения.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора удаляемого предложения.

© «Битрикс», 2001-2008, «1С-Битрикс» 2008-2022

onCrmQuoteUpdate

Событие, вызываемое при изменении коммерческого предложения.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора изменяемого предложения.

© «Битрикс», 2001-2008, «1С-Битрикс»

onCrmQuoteUserFieldAdd

Событие, вызываемое при добавлении пользовательского поля.

Параметры события

Параметр	Описание	С версии
id	Идентификатор пользовательского поля.	
entityId	Символьный идентификатор сущности, для которой создано поле	
fieldName	Имя созданного пользовательского поля	

© «Битрикс», 2001-2008, «1С-Битрикс» 2008-2022

onCrmQuoteUserFieldDelete

Событие, вызываемое при удалении пользовательского поля.

Параметры события

Параметр	Описание	С версии
id	Идентификатор пользовательского поля.	
entityId	Символьный идентификатор сущности, которой принадлежит поле	
fieldName	Имя удаляемого пользовательского поля	

© «Битрикс», 2001-2008, «1С-Битрикс», 2009-2022

onCrmQuoteUserFieldSetEnum

Событие, вызываемое при изменении набора значений для пользовательского поля списочного типа.

Параметры события

Параметр	Описание	С версии
id	Идентификатор пользовательского поля.	
entityId	Символьный идентификатор сущности, которой принадлежит поле	
fieldName	Имя пользовательского поля, для которого изменился набор значений	

© «Битрикс», 2001-2008, «1С-

onCrmQuoteUserFieldUpdate

Событие, вызываемое при изменении пользовательского поля.

Параметры события

Параметр	Описание	С версии
id	Идентификатор пользовательского поля.	
entityId	Символьный идентификатор сущности, которой принадлежит поле	
fieldName	Имя изменяемого пользовательского поля	

© «Битрикс», 2001-2008, «1С-Битрикс», 2009-2022

crm.invoice.add

```
crm.invoice.add(fields)
```

Создаёт новый счет. Если необходимо в счёте указать какие реквизиты покупателя/продавца (т.к. их может быть несколько у компании), то используйте метод crm.requisite.link.register.

В создаваемом счёте обязательно должны быть указаны компании продавца и покупателя:
`UF_COMPANY_ID`, если покупатель - компания или `UF_CONTACT_ID`, если покупатель - контакт.
`UF_MYCOMPANY_ID` - продавец.

Идентификаторы указанные в **crm.requisite.link.register** и в создаваемом счёте должны соответствовать покупателю и продавцу.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей счёта. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.invoice.fields и посмотрите формат пришедших значений этих полей.

Пример

Пример 1

```
        var current = new Date();
        var nextMonth = new Date();

nextMonth.setMonth(current.getMonth() + 1);

        var date2str = function(d)
        {
                return d.getFullYear() + '-'
+ paddatepart(1 + d.getMonth()) + '-' +
paddatepart(d.getDate()) + 'T' +
paddatepart(d.getHours())
                        + ':' +
paddatepart(d.getMinutes()) + ':' +
paddatepart(d.getSeconds()) + '+03:00';
        };

        var paddatepart = function(part)
        {
                return part >= 10 ?
part.toString() : '0' + part.toString();
        };

        BX24.callMethod(
                "crm.invoice.add",
                {
                        "fields": {

"ORDER_TOPIC": "Счёт для юр. лица",
                                "STATUS_ID":
"P",

"DATE_INSERT": date2str(current),
```

```
"PAY_VOUCHER_DATE": date2str(current),

"PAY_VOUCHER_NUM": "876",

"DATE_MARKED": date2str(current),

"REASON_MARKED": "Счёт оплачен сразу.",
                                "COMMENTS":
"комментарий менеджера",

"USER_DESCRIPTION": "комментарий для
клиента",
                                "DATE_BILL":
date2str(current),

"DATE_PAY_BEFORE": date2str(nextMonth),

"RESPONSIBLE_ID": 1,

"UF_DEAL_ID": 10,

"UF_COMPANY_ID": 5,

"UF_CONTACT_ID": 2,

"PERSON_TYPE_ID": 2,

"PAY_SYSTEM_ID": 6,

"INVOICE_PROPERTIES": {

"COMPANY": "ООО \"Новые технологии\"",
// Название компании

"COMPANY_ADR": "543000 Москва, ул. Песчаная,
д. 15, оф. 55 (юр)",   // Юридический адрес
```

"INN": "",
// ИНН

"KPP": "",
// КПП

"CONTACT_PERSON": "Борис Соколов",
// Контактное лицо

"EMAIL": "pr@logistics-north.com",
// E-Mail

"PHONE": "8 (495) 234-54-32",
// Телефон

"FAX": "",
// Факс

"ZIP": "",
// Индекс

"CITY": "",
// Город

"LOCATION": "",
// Местоположение

"ADDRESS": ""
// Адрес доставки
},

"PRODUCT_ROWS": [

{"ID": 0, "PRODUCT_ID": 438, "PRODUCT_NAME": "Товар 01", "QUANTITY": 1, "PRICE": 100},

{"ID": 0, "PRODUCT_ID": 515, "PRODUCT_NAME":

```
"Товар 77", "QUANTITY": 1, "PRICE": 118}
                                ]
                        }
                },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.info("Создан счёт с ID " +
result.data());
                }
        );
```

Пример 2

```
var current = new Date();
        var nextMonth = new Date();

nextMonth.setMonth(current.getMonth() + 1);

        var date2str = function(d)
        {
                return d.getFullYear() + '-'
+ paddatepart(1 + d.getMonth()) + '-' +
paddatepart(d.getDate()) + 'T' +
paddatepart(d.getHours())
                        + ':' +
paddatepart(d.getMinutes()) + ':' +
paddatepart(d.getSeconds()) + '+03:00';
        };

        var paddatepart = function(part)
```

```
        {
                return part >= 10 ?
part.toString() : '0' + part.toString();
        };

        BX24.callMethod(
                "crm.invoice.add",
                {
                        "fields": {

"ORDER_TOPIC": "Счёт для физ. лица)",
                                "STATUS_ID":
"P",

"DATE_INSERT": date2str(current),

"PAY_VOUCHER_DATE": date2str(current),

"PAY_VOUCHER_NUM": "876",

"DATE_MARKED": date2str(current),

"REASON_MARKED": "оплатили",
                                "COMMENTS":
"комментарий",

"USER_DESCRIPTION": "комментарий клиенту
",
                                "DATE_BILL":
date2str(current),

"DATE_PAY_BEFORE": date2str(nextMonth),

"RESPONSIBLE_ID": 1,

"UF_DEAL_ID": 8,
```

"UF_COMPANY_ID": 0,

"UF_CONTACT_ID": 3,

"PERSON_TYPE_ID": 1,

"PAY_SYSTEM_ID": 6,

"INVOICE_PROPERTIES": {

"FIO": "Глеб Титов",
// Ф.И.О.

"EMAIL": "boss@yt-soft.net",
// E-Mail

"PHONE": "",
// Телефон

"ZIP": "",
// Индекс

"CITY": "",
// Город

"LOCATION": "",
// Местоположение

"ADDRESS": ""
// Адрес доставки
},

"PRODUCT_ROWS": [

{"ID": 0, "PRODUCT_ID": 438, "PRODUCT_NAME":
"Товар 01", "QUANTITY": 1, "PRICE": 100},

```
{"ID": 0, "PRODUCT_ID": 515, "PRODUCT_NAME":
"Товар 77", "QUANTITY": 1, "PRICE": 118}
                                ]
                        }
                },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.info("Создан счёт с ID " +
result.data());
                }
        );
```

crm.invoice.delete

```
crm.invoice.delete(id)
```

Удаляет счёт.

Параметры

Параметр	Описание
id	Идентификатор счета.

Пример

```
        var id = prompt("Введите ID");
        BX24.callMethod(
                "crm.invoice.delete",
                { "id": id },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else
```

```
console.info(result.data());
                    }
          );
```

crm.invoice.fields

```
crm.invoice.fields()
```

Возвращает описание полей счёта, в том числе пользовательских.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(
                "crm.invoice.fields",
                {},
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
        );
```

Поля, возвращаемые методом

Поле	Описание	Тип
ID	Идентификатор	interger
ACCOUNT_NUMBER	Номер	string
COMMENTS	Комментарий менеджера	text
CREATED_BY	Создано пользователем	integer
CURRENCY	Идентификатор валюты	crm_currency
DATE_BILL	Дата выставления	date
DATE_INSERT	Дата создания	datetime
DATE_MARKED	Дата отклонения	datetime
DATE_PAY_BEFORE	Срок оплаты	date
DATE_PAYED	Дата перевода в состояние оплаты	datetime
DATE_STATUS	Дата изменения статуса	datetime
DATE_UPDATE	Дата изменения	datetime

EMP_PAYED_ID	Идентификатор пользователя, который последним перевёл счёт в состояние "оплачен"	integer
EMP_STATUS_ID	Идентификатор пользователя, который последним поменял статус счёта	integer
LID	Идентификатор сайта	integer
XML_ID	Внешний код	string
ORDER_TOPIC	Тема	string
PAY_SYSTEM_ID	Идентификатор печатной формы	integer
PAY_VOUCHER_DATE	Дата оплаты	date
PAY_VOUCHER_NUM	Номер документа оплаты	string
PAYED	Признак оплаченности	char
PERSON_TYPE_ID	Идентификатор типа плательщика	integer
PRICE	Сумма	double
REASON_MARKED	Комментарий статуса	string

RESPONSIBLE_EMAIL	E-mail ответственного	string
RESPONSIBLE_ID	Идентификатор ответственного	integer
RESPONSIBLE_LAST_NAME	Фамилия ответственного	string
RESPONSIBLE_LOGIN	Логин ответственного	string
RESPONSIBLE_NAME	Имя ответственного	string
RESPONSIBLE_PERSONAL_PHOTO	Идентификатор фото ответственного	integer
RESPONSIBLE_SECOND_NAME	Отчество ответственного	string
RESPONSIBLE_WORK_POSITION	Должность ответственного	string
STATUS_ID	Идентификатор статуса	crm_status
TAX_VALUE	Сумма налога	double
UF_COMPANY_ID	Идентификатор компании	integer
UF_CONTACT_ID	Идентификатор контакта	integer
UF_MYCOMPANY_ID	Идентификатор	integer

	своей компании	
UF_DEAL_ID	Идентификатор связанной сделки	integer
USER_DESCRIPTION	Комментарий	string
PR_LOCATION	Идентификатор местоположения	integer
INVOICE_PROPERTIES	Список свойств	array

PRODUCT_ROWS	Список товарных позиций	array

crm.invoice.get

```
crm.invoice.get(id)
```

Возвращает счёт по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор счета.

Пример

```
        var id = prompt("Введите ID");
        BX24.callMethod(
                "crm.invoice.get",
                { "id": id },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else
```

```
console.dir(result.data());
                    }
          );
```

crm.invoice.getexternallink

```
crm.invoice.getexternallink(id)
```

Метод возращает публичную ссылку для онлайн-счета.

Параметры

Параметр	Описание
id	Идентификатор счета.

Примеры

```
var id = prompt("Введите ID");
BX24.callMethod(
    "crm.invoice.getexternallink",
    { "id": id },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.log(result.data());
```

```
    }
);
```

Запрос

```
https://******.bitrix24.ru/rest/crm.invoice.
getexternallink.json?auth=******&ID=2
```

Ответ

```
https://******.bitrix24.ru/pub/pay/Mg==/0072
4392abf8a2da3bc85d7382f5753c/
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.invoice.list

Описание

Возвращает список счетов. Является реализацией списочного метода для счетов.

При выборке используйте маски:

- "*" - для выборки всех полей (без пользовательских)
- "UF_*"- для выборки всех пользовательских полей.

Свойства и товарные позиции счёта метод crm.invoice.list не возвращает.

Для получения свойств и товарных позиций нужно использовать метод crm.invoice.get.

Параметры

См. описание списочных методов.

Параметр	Описание
filter	Фильтр записей. По умолчанию отдаются все записи, без фильтрации.
order	Сортировка записей. Поддерживается сортировка по тем же полям, что и в фильтре.

Пример

Пример выводит данные в консоль. Если нужно вывести данные по другому, то реализуйте свою обработку данных, возвращенных вызовами **result.data()** и **result.error()**.

```
        BX24.callMethod(
                "crm.invoice.list",
                {
                        "order": {
"DATE_INSERT": "ASC" },
                        "filter": {
">PRICE": 100 },
                        "select": [ "ID",
"ACCOUNT_NUMBER", "ORDER_TOPIC",
"DATE_INSERT", "STATUS_ID", "PRICE",
"CURRENCY_ID" ]
                },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else
                        {

console.dir(result.data());

if(result.more())

result.next();
                        }
                }
        );
```

Поля, возвращаемые методом

Поле	Описание	Тип	
ID	Идентификатор	interger	Т
ACCOUNT_NUMBER	Номер	string	С
COMMENTS	Комментарий менеджера	text	
CREATED_BY	Создано пользователем	integer	Т
CURRENCY	Идентификатор валюты	crm_currency	Т
DATE_BILL	Дата выставления	date	
DATE_INSERT	Дата создания	datetime	
DATE_MARKED	Дата отклонения	datetime	У с
DATE_PAY_BEFORE	Срок оплаты	date	
DATE_PAYED	Дата перевода в состояние оплаты	datetime	Т
DATE_STATUS	Дата изменения статуса	datetime	Т
DATE_UPDATE	Дата изменения	datetime	Т
EMP_PAYED_ID	Идентификатор пользователя, который последним	integer	Т

	перевёл счёт в состояние "оплачен"		
EMP_STATUS_ID	Идентификатор пользователя, который последним поменял статус счёта	integer	Т
LID	Идентификатор сайта	integer	Т
IS_RECURRING	Флаг шаблона регулярной сделки (если стоит Y, то это не сделка, а шаблон)	char	
XML_ID	Внешний код	string	
ORDER_TOPIC	Тема	string	С
PAY_SYSTEM_ID	Идентификатор печатной формы	integer	С
PAY_VOUCHER_DATE	Дата оплаты	date	У с
PAY_VOUCHER_NUM	Номер документа оплаты	string	У с
PAYED	Признак оплаченности	char	Т
PERSON_TYPE_ID	Идентификатор типа плательщика	integer	С
PRICE	Сумма	double	Т

REASON_MARKED	Комментарий статуса	string	У с с
RESPONSIBLE_EMAIL	E-mail ответственного	string	Т
RESPONSIBLE_ID	Идентификатор ответственного	integer	
RESPONSIBLE_LAST_NAME	Фамилия ответственного	string	Т
RESPONSIBLE_LOGIN	Логин ответственного	string	Т
RESPONSIBLE_NAME	Имя ответственного	string	Т
RESPONSIBLE_PERSONAL_PHOTO	Идентификатор фото ответственного	integer	Т
RESPONSIBLE_SECOND_NAME	Отчество ответственного	string	Т
RESPONSIBLE_WORK_POSITION	Должность ответственного	string	Т
STATUS_ID	Идентификатор статуса	crm_status	И с "
TAX_VALUE	Сумма налога	double	Т
UF_COMPANY_ID	Идентификатор компании	integer	У п " л
UF_CONTACT_ID	Идентификатор контакта	integer	У п " л

			к к к
UF_MYCOMPANY_ID	Идентификатор своей компании	integer	У к к р (р У с
UF_DEAL_ID	Идентификатор связанной сделки	integer	
UF_QUOTE_ID	Идентификатор связанного коммерческого предложения	integer	
USER_DESCRIPTION	Комментарий	string	

crm.invoice.recurring.add

```
crm.invoice.recurring.add(fields)
```

Добавляет новую настройку для регулярного счета.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array ("поле"=>"значение"[, ...]), содержащий значения полей настройки регулярного счета. Обязательное поле - поле INVOICE_ID [ID счета, у которой задан параметр IS_RECURRING=Y]. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.invoice.recurring.fields и посмотрите формат пришедших значений этих полей.

Пример:

```
   var current = new Date();
   var nextMonth = new Date();
   nextMonth.setMonth(current.getMonth() +
1);
```

```
   var date2str = function(d)
   {
        return d.getFullYear() + '-' +
paddatepart(1 + d.getMonth()) + '-' +
paddatepart(d.getDate()) + 'T' +
paddatepart(d.getHours()) + ':' +
paddatepart(d.getMinutes()) + ':' +
paddatepart(d.getSeconds()) + '+03:00';
   };
   var paddatepart = function(part)
   {
        return part >= 10 ? part.toString()
: '0' + part.toString();
   };

   BX24.callMethod(
       "crm.invoice.recurring.add",
       {
           fields:
           {
               "INVOICE_ID": "10",
               "IS_LIMIT": "N",
               "START_DATE":
date2str(nextMonth),
               "PARAMS": {
                   "PERIOD": "day",
                   "IS_WORKING_ONLY": "N",
                   "INTERVAL": 30,

"DATE_PAY_BEFORE_OFFSET_TYPE": "month",

"DATE_PAY_BEFORE_OFFSET_VALUE": 1,
               }
           }
       },
       function(result)
       {
```

```
            if(result.error())

console.error(result.error());
            else
                console.info("Добавлены
настройки регулярного счета. ID записи - " +
result.data());
        }
    );
```

crm.invoice.recurring.delete

```
crm.invoice.recurring.delete(id)
```

Удаляет существующую настройку для шаблона регулярного счета.

Параметры

Параметр	Описание
id	Идентификатор настройки шаблона регулярного счета.

Пример:

```
    var id = prompt("Введите ID");
    BX24.callMethod(
        "crm.invoice.recurring.delete",
        { id: id },
        function(result)
        {
            if(result.error())

console.error(result.error());
```

```
            else
                console.info(result.data());
        }
    );
```

crm.invoice.recurring.expose

```
crm.invoice.recurring.expose(id)
```

Создает новый счет из шаблона.

Параметры

Параметр	Описание
id	Идентификатор настройки шаблона регулярного счета.

Пример:

```
var id = prompt("Введите ID");
BX24.callMethod(
    "crm.invoice.recurring.expose",
    {
        id: id,
    },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error());
            else
            {
                console.info(result.data());
            }
        }
    );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.invoice.recurring.fields

Описание и пример

```
crm.invoice.recurring.fields()
```

Возвращает список полей настройки шаблона регулярного счета с описанием.

Параметры

Без параметров.

Пример

```
   BX24.callMethod(
       "crm.invoice.recurring.fields",
       {},
       function(result)
       {
           if(result.error())

console.error(result.error());
           else
               console.dir(result.data());
```

```
        }
    );
```

Поля

Поле	Описание
ID	Идентификатор записи в таблице настроек регулярного счета
INVOICE_ID	ID шаблона счета
ACTIVE	Флаг активности. Значения: Y/N
NEXT_EXECUTION	Дата следующего создания нового счета из шаблона. Рассчитывается системой по указанным параметрам. Если значение пустое, новые счета не создаются
LAST_EXECUTION	Дата последнего создания нового счета из шаблона
COUNTER_REPEAT	Количество созданных из шаблона счетов
START_DATE	Дата начала отсчета при расчете даты следующего создания нового счета - date - Если значние пустое, рассчитывается от текущей даты
SEND_BILL	Отправлять счет на почту, привязанную к плательщику. Значения: Y/N
EMAIL_ID	ID поля, содержащего email плательщика
IS_LIMIT	Есть ли ограничения по созданию новых счетов. Значения: N - без ограничений, D - установлено ограничение по дате, T - установлено ограничение по количеству новых счетов
LIMIT_REPEAT	Максимальное число счетов, которое можно

	создать из этого шаблона
LIMIT_DATE	Дата, до достижения которой можно создавать счета из этого шаблона
PARAMS	Набор параметров для расчета - recurring_params: ▪ **PERIOD** - период повторения: ▪ day - день ▪ week - неделя ▪ month - месяц ▪ year - год ▪ **TYPE** - тип повторения для месяца и года: ▪ если PERIOD равен month ▪ 1 - расчет по порядковому номеру дня в месяце ▪ 2 - расчет по номерам дней недели в месяце ▪ если PERIOD равен year ▪ 1 - расчет по порядковому номеру дня в заданном месяце ▪ 2 - расчет по номерам дней недели в заданном месяце ▪ **INTERVAL** - смещение при расчете ▪ **IS_WORKING_ONLY** - учитываются только рабочие дни (Y/N) ▪ **WEEKDAY** - полное наименование дня недели (согласно форматирования PHP метода date()) ▪ **NUM_DAY_IN_MONTH** - порядковый номер даты в месяце (PERIOD равен month или year) ▪ **NUM_WEEKDAY_IN_MONTH** - номер дня недели в месяце (PERIOD равен month или year)

- **FIELD_YEARLY_INTERVAL_MONTH_NAME** - номер дня недели в месяце (PERIOD равен month или year)
- **DATE_PAY_BEFORE_OFFSET_TYPE** - значение смещения для расчета срока оплаты, расчет ведется от момента создания нового счета из шаблона:
  - day - день
  - week - неделя
  - month - месяц
  - year - год
- **DATE_PAY_BEFORE_OFFSET_VALUE** - значение смещения для расчета срока оплаты, расчет ведется от момента создания нового счета из шаблона

crm.invoice.recurring.get

```
crm.invoice.recurring.get(id)
```

Возвращает поля настройки шаблона регулярного счета по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор настройки шаблона регулярной счета.

Пример:

```
    var id = prompt("Введите ID");
    BX24.callMethod(
        "crm.invoice.recurring.get",
        { id: id },
        function(result)
        {
            if(result.error())

console.error(result.error());
```

```
            else
                console.dir(result.data());
        }
    );
```

crm.invoice.recurring.list

```
crm.invoice.recurring.list ()
```

Возвращает список настроек шаблонов регулярных счетов по фильтру.

При выборке используйте маску "*" для выборки всех полей (без пользовательских и множественных).

Параметры

См. описание списочных методов.

Пример:

```
   BX24.callMethod(
       "crm.invoice.recurring.list",
       {
           order: { "INVOICE_ID": "ASC" },
           filter: { ">COUNTER_REPEAT": 5 },
           select: [ "ID", "INVOICE_ID ",
"NEXT_EXECUTION", "LAST_EXECUTION",
"SEND_BILL", "IS_LIMIT" ]
       },
       function(result)
       {
```

```
            if(result.error())

console.error(result.error());
            else
            {
                console.dir(result.data());
                if(result.more())
                    result.next();
            }
        }
    );
```

crm.invoice.recurring.update

```
crm.invoice.recurring.update(id, fields)
```

Обновляет существующую настройку для шаблона регулярного счета.

Параметры

Параметр	Описание
id	Идентификатор настройки шаблона регулярного счета.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.invoice.recurring.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.invoice.recurring.fields и посмотрите формат пришедших значений этих полей.

Пример:

```
  var id = prompt("Введите ID");
   BX24.callMethod(
       "crm.invoice.recurring.update",
       {
           id: id,
           fields:
           {
               "SEND_BILL": "Y",
               "EMAIL_ID": 136,
               "PARAMS": {
                   "MODE": "month",
                   "TYPE": 2,
                   "INTERVAL": 3,
                   "WEEKDAY": "Monday",
                   "NUM_WEEKDAY_IN_MONTH":
4,

"DATE_PAY_BEFORE_OFFSET_TYPE": "day",

"DATE_PAY_BEFORE_OFFSET_VALUE": 15,
               }
           },
       },
       function(result)
       {
           if(result.error())

console.error(result.error());
           else
           {
               console.info(result.data());
           }
       }
   );
```

crm.invoice.update

```
crm.invoice.update(id, fields)
```

Обновляет существующий счёт.

Параметры

Параметр	Описание
id	Идентификатор счета.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.invoice.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.invoice.fields и посмотрите формат пришедших значений этих полей.

Пример

```
// Добавление или обновление товара в счёте.
```

```
var current = new Date();

var date2str = function(d)
{
        return d.getFullYear() + '-' +
paddatepart(1 + d.getMonth()) + '-' +
paddatepart(d.getDate()) + 'T' +
paddatepart(d.getHours())
                + ':' +
paddatepart(d.getMinutes()) + ':' +
paddatepart(d.getSeconds()) + '+03:00';
};

var paddatepart = function(part)
{
        return part >= 10 ? part.toString()
: '0' + part.toString();
};

var id = prompt("Введите ID");
BX24.callMethod('crm.invoice.get', {"id":
id}, addProduct);
function addProduct(result)
{
        if(result.error())

console.error(result.error());
        else
        {
                var fields =
clone(result.data());
                var n =
fields['PRODUCT_ROWS'].length;
                var productUpdated = false;

                // Изменение поля "Дата
выставления"
```

```
                fields["DATE_BILL"] =
date2str(current);
                // Изменение поля
"Комментарий (отобразится в счёте)"
                fields["USER_DESCRIPTION"] =
"Комментарий для клиента (обновлённый).";

                // Если товар с ID 703 есть
в счёте, то обновляем его поля.
                // Если товара с ID 703 в
счёте нет, то добавляем его в счёт.
                // Если используется НДС, то
читается что цена его включает, а сам
признак включения НДС в цену будет
                // взят из каталога.
                for (var i in
fields['PRODUCT_ROWS'])
                {
                        if
(fields['PRODUCT_ROWS'][i]["PRODUCT_ID"] ==
703)
                        {
                                var rowId =
fields['PRODUCT_ROWS'][i]["ID"]

fields['PRODUCT_ROWS'][i] = {

"ID": rowId, "PRODUCT_ID": 703, "QUANTITY":
4, "PRICE": 779.60
                                };

productUpdated = true;
                                break;
                        }
                }
                if (!productUpdated && n >
0)
```

```
                {

fields['PRODUCT_ROWS'][n] = {
                                "ID": 0,
"PRODUCT_ID": 703, "QUANTITY": 5, "PRICE":
779.60
                        };
                }

BX24.callMethod('crm.invoice.update', {"id":
id, "fields": fields},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
                                {

console.info("Обновлён счёт с ID " +
result.data());
                                }
                        }
                );
        }
}

function clone(src)
{
        var dst;
        if (src instanceof Object)
        {
                dst = {};
                for (var i in src)
                {
                        if (src[i]
```

```
instanceof Object)
                                        dst[i] =
clone(src[i]);
                                else
                                        dst[i] =
src[i];
                        }
                }
                else dst = src;
                return dst;
}
```

crm.invoice.userfield.add

```
crm.invoice.userfield.add(fields)
```

Создаёт новое пользовательское поле для счетов.

Системное ограничение на название поля - 20 знаков. К названию пользовательского поля всегда добавляется префикс UF_CRM_, то есть реальная длина названия - 13 знаков.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий описание пользовательского поля. Полное описание полей можно получить вызовом метода crm.userfield.fields.
LIST	Содержит набор значений списка для пользовательских полей типа Список. Указывается при создании/обновлении поля. Каждое значение представляет собой массив с полями: ▪ **VALUE** - значение элемента списка. Поле является обязательным в случае, когда создается новый элемент. ▪ **SORT** - сортировка. ▪ **DEF** - если равно Y, то элемент списка является значением по-умолчанию. Для

множественного поля допустимо несколько DEF=Y. Для не множественного, дефолтным будет считаться первое.

- **XML_ID** - внешний код значения. Параметр учитывается только при обновлении уже существующих значений элемента списка.
- **ID** - идентификатор значения. Если он указан, то считается что это обновление существующего значения элемента списка, а не создание нового. Имеет смысл только при вызове методов `*.userfield.update`.
- **DEL** - если равно Y, то существующий элемент списка будет удален. Применяется, если заполнен параметр ID.

Пример

```
        BX24.callMethod(
                "crm.paysystem.list", {
                        order: {"SORT":
"ASC"},
                        filter: {
                                "%NAME":
"Предложение",
                        }
                },
                function (result)
                {
                        if (result.error())
                        {
console.error(result.error());
                        }
                        else
```

```
                        {
console.dir(result.data());
                                if
(result.more())
result.next();
                        }
                }
        );
```

crm.invoice.userfield.delete

```
crm.invoice.userfield.delete(id)
```

Удаляет пользовательское поле счетов.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.

Пример

```
        var id = prompt("Введите ID");
        BX24.callMethod(

"crm.invoice.userfield.delete",
                {id: id},
                function (result)
                {
                        if (result.error())

console.error(result.error());
```

```
                                else

console.info(result.data());
                    }
          );
```

crm.invoice.userfield.get

```
crm.invoice.userfield.get(id)
```

Возвращает пользовательское поле счетов по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.

Пример

```
        var id = prompt("Введите ID");
        BX24.callMethod(

"crm.invoice.userfield.get",
                        {id: id},
                        function (result)
                        {
                                if
(result.error())

console.error(result.error());
```

```
                                        else

console.dir(result.data());
                                }
        );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.invoice.userfield.list

```
crm.invoice.userfield.list(order, filter)
```

Возвращает список пользовательских полей счетов по фильтру.

Параметры

Параметр	Описание
order	Поля сортировки.
filter	Поля фильтра.

Пример

```
        BX24.callMethod(

"crm.invoice.userfield.list",
                {
                        order: {"SORT":
"ASC"},
                        filter:
{"MANDATORY": "N"}
                },
```

```
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else
                        {

console.dir(result.data());
                                if
(result.more())

result.next();
                        }
                }
        );
```

crm.invoice.userfield.update

```
crm.invoice.userfield.update(id, fields)
```

Обновляет существующее пользовательское поле счетов.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.userfield.fields.
LIST	Содержит набор значений списка для пользовательских полей типа Список. Указывается при создании/обновлении поля. Каждое значение представляет собой массив с полями: ▪ **VALUE** - значение элемента списка. Поле является обязательным в случае, когда создается новый элемент. ▪ **SORT** - сортировка. ▪ **DEF** - если равно Y, то элемент списка является значением по-умолчанию. Для

	множественного поля допустимо несколько DEF=Y. Для не множественного, дефолтным будет считаться первое. ▪ **XML_ID** - внешний код значения. Параметр учитывается только при обновлении уже существующих значений элемента списка. ▪ **ID** - идентификатор значения. Если он указан, то считается что это обновление существующего значения элемента списка, а не создание нового. Имеет смысл только при вызове методов `*.userfield.update`. ▪ **DEL** - если равно Y, то существующий элемент списка будет удален. Применяется, если заполнен параметр ID.

Пример

```
        var id = prompt("Введите ID");
        var label = prompt("Введите новое
название");
        BX24.callMethod(

"crm.invoice.userfield.update",
                {
                        id: id,
                        fields: {

"EDIT_FORM_LABEL": label,

"LIST_COLUMN_LABEL": label
                        }
                },
                function (result)
                {
```

```
                        if (result.error())

console.error(result.error());
                        else
                        {

console.dir(result.data());
                                if
(result.more())

result.next();
                        }
                }
        );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.paysystem.fields

```
crm.paysystem.fields()
```

Возвращает описание полей для способов оплаты.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(
"crm.paysystem.fields",
                        {},
                        function(result)
                        {
if(result.error())
console.error(result.error());
                                else
console.dir(result.data());
```

```
                                        }
            );
```

crm.paysystem.list

```
crm.paysystem.list(order, filter)
```

Возвращает список способов оплаты, применимых к предложениям либо счетам.

Параметры

Параметр	Описание
order	Поля сортировки.
filter	Поля фильтра.

Пример

```
        BX24.callMethod(
                "crm.paysystem.list", {
                        order: {"SORT":
"ASC"},
                        filter: {
                                "%NAME":
"Предложение",
                        }
```

```
                },
                function (result)
                {
                        if (result.error())
                        {
console.error(result.error());
                        }
                        else
                        {
console.dir(result.data());
                                if
(result.more())
result.next();
                        }
                }
        );
```

crm.persontype.fields

```
crm.persontype.fields()
```

Возвращает описание полей для типов плательщиков.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(
"crm.persontype.fields",
                        {},
                        function(result)
                        {
if(result.error())
console.error(result.error());
                                else
console.dir(result.data());
```

```
                                        }
            );
```

crm.persontype.list

```
crm.persontype.list(order, filter)
```

Возвращает список типов плательщиков.

Для платёжных систем, которые используются в CRM (для счетов, сделок), типы плательщиков нужно получать через метод **crm.persontype.list**, в случае если создаётся платёжная система для заказов, то нужно использовать sale.persontype.list.

Параметры

Параметр	Описание
order	Поля сортировки.
filter	Поля фильтра.

Пример

```
BX24.callMethod(
        "crm.persontype.list", {
                order: {"ID":
```

```
"ASC"},
                        filter: {
                                "NAME":
"CRM_COMPANY",
                        }
                },
                function (result)
                {
                        if (result.error())
                        {

console.error(result.error());
                        }
                        else
                        {

console.dir(result.data());
                                if
(result.more())

result.next();
                        }
                }
        );
```

onCrmInvoiceAdd

Событие, вызываемое при создании счёта.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора созданного счёта.

© «Битрикс», 2001-2008, «1С-Битрикс» 2008-2022

onCrmInvoiceDelete

Событие, вызываемое при удалении счёта.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора удаленного счёта.

© «Битрикс», 2001-2008, «1С-Битрикс» 2008-2022

onCrmInvoiceSetStatus

Событие, вызываемое при изменении статуса счёта.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора счёта с изменённым статусом.

© «Битрикс», 2001-2008, «1С-Битрикс»

onCrmInvoiceUpdate

Событие, вызываемое при обновлении счёта.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора обновленного счёта.

© «Битрикс», 2001-2008, «1С-Битрикс»

onCrmInvoiceUserFieldAdd

Событие, вызываемое при добавлении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmInvoiceUserFieldDelete

Событие, вызываемое при удалении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmInvoiceUserFieldSetEnum

Событие, вызываемое при изменении набора значений для пользовательского поля списочного типа.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmInvoiceUserFieldUpdate

Событие, вызываемое при изменении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmInvoiceRecurringAdd

Событие, вызываемое при создании нового регулярного счета.

Параметры

Параметр	Описание
FIELDS	Массив содержит следующие поля: ▪ **ID** - значение идентификатора записи в таблице настроек регулярного счета; ▪ **RECURRING_INVOICE_ID** - значение идентификатора шаблона регулярного счета.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmInvoiceRecurringDelete

Событие, вызываемое при удалении регулярного счета.

Параметры

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора записи в таблице настроек регулярного счета.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmInvoiceRecurringExpose

Событие, вызываемое при выставлении нового счета из регулярного счета.

Параметры

Параметр	Описание
FIELDS	Массив содержит следующие поля: ▪ **ID** - значение идентификатора записи в таблице настроек регулярного счета; ▪ **RECURRING_INVOICE_ID** - значение идентификатора шаблона регулярного счета; ▪ **INVOICE_ID** - значение ID нового счета, созданного на основе регулярного.

© «Битрикс», 2001-2008, «1С-Битрикс», 2009-2022

onCrmInvoiceRecurringUpdate

Событие, вызываемое при обновлении регулярного счета.

Параметры

Параметр	Описание
FIELDS	Массив содержит следующие поля: ▪ **ID** - значение идентификатора записи в таблице настроек регулярного счета; ▪ **RECURRING_INVOICE_ID** - значение идентификатора шаблона регулярного счета.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Настройки смарт-процессов

Начальные настройки

Настройки пользовательских полей

Невозможно представить рабочий смарт-процесс без собственного набора пользовательских полей.

Управление ими вынесено в отдельный scope userfieldconfig.

Права доступа

Методы доступны только тем приложениям/вебхукам, владельцы которых являются администраторами CRM.

Идентификаторы

При работе с этими методами в качестве идентификатора id необходимо передавать идентификатор настроек смарт-процесса (первичный ключ), а не идентификатор типа entityTypeId.

Связанные данные

Эти методы позволяют управлять не только полями самих настроек смарт-процессов, но и связанными данными - настройками связей, привязками к пользовательским полям, а также к настройкам показа смарт-процесса вне CRM.

Настройки связей

Настройки связей отдаются по ключу `relations` в следующем виде:

```
{
  "parent": [],
  "child": []
}
```

- `parent` - настройки привязок к этому смарт-процессу;
- `child` - настройки привязок этого смарт-процесса к другим разделам.

где каждый элемент массива имеет следующую структуру с описанием связи:

```
{
  "entityTypeId": number,
  "isChildrenListEnabled": boolean,
  "isPredefined": boolean
}
```

Здесь

- `entityTypeId` - идентификатор типа связанного раздела;
- `isChildrenListEnabled` - включить отображение отдельного таба со списком связанных элементов в карточке родителя;
- `isPredefined` - если здесь стоит true, то связь является "предустановленной". **Настройки таких связей изменить нельзя!**

При изменении настройки связей должны передаваться в таком же виде.

При изменении настроек связей смарт-процесса необходимо передавать набор настроек целиком, либо

опустить ключ `relations` вообще. Настройки переписываются целиком.

Привязки к пользовательским полям

Если поле `isUseInUserfieldEnabled` установлено в `true`, то можно передать по ключу `linkedUserFields` набор полей, в которых должен отображаться этот смарт-процесс.

- `'CALENDAR_EVENT|UF_CRM_CAL_EVENT'` - событие в календаре.
- `'TASKS_TASK|UF_CRM_TASK'` - задачи.
- `'TASKS_TASK_TEMPLATE|UF_CRM_TASK'` - шаблон задачи.

Если в поле `isUseInUserfieldEnabled` передать `false`, то все настроенные привязки будут отключены.

При изменении привязки к пользовательским полям передавать набор настроек целиком, либо опустить ключ `linkedUserFields` вообще. Настройки переписываются.

Показ вне CRM

По ключу `customSections` можно передать массив с описанием дополнительных разделов. Каждый элемент массива имеет следующую структуру:

```
{
  "id": number,
  "title": string,
  "isSelected": boolean
}
```

Ключ `isSelected` можно игнорировать, он используется для отображения диалога в настройках. За фактическую привязку смарт-процесса к разделу отвечает параметр `customSectionId`.

Примеры запросов есть в описании метода `crm.type.update`.

При изменении списка дополнительных разделов необходимо передавать его целиком, либо опустить ключ `customSections` вообще. Настройки переписываются.

Направления смарт-процессов

Описание

Обязательный параметр entityTypeId

На вход всех методов обязательно должен передаваться параметр `entityTypeId`, где должен быть записан идентификатор типа сущности CRM.

Если тип сущности не поддерживает работу с направлениями (например, лиды или предложения), то методы работать не будут.

Направления по умолчанию

У каждого типа сущности одновременно может быть только одно направление по умолчанию. В связи с этим есть ряд ограничений:

- нельзя удалить направление по умолчанию;
- при создании нового направления и передаче ему флага `"isDefault": "Y"` старое направление по умолчанию перестанет быть направлением по умолчанию;
- при изменении направления по умолчанию нельзя сделать его направлением не по умолчанию;
- при изменении направления направления не по умолчанию с передачей ему флага `"isDefault": "Y"` старое направление по умолчанию перестанет быть направлением по умолчанию.

Если для имеющегося смарт-процесса отключен показ направлений в интерфейсе, то работа с направлениями через rest всё равно возможна.

Методы

Методы работы с направлениями сделок и смарт-процессов описаны в отдельном разделе.

Методы	Описание
crm.category.fields	Отдает информацию о полях направлений.
crm.category.get	Отдает информацию о направлении с идентификатором `id`.
crm.category.list	Возвращает массив направлений, которые относятся к типу сущности с идентификатором `entityTypeId`.
crm.category.add	Создаёт у типа сущности с идентификатором `entityTypeId` новое направление с полями `fields`.
crm.category.update	Обновляет направление с идентификатором `id`, задав ему новые значения полей из `fields`.
crm.category.delete	Удаляет направление с идентификатором `id`, которое относится к типу сущности с идентификатором `entityTypeId`.

Стадии смарт-процессов

Работа со стадиями смарт-процессов осуществляется через общий набор методов crm.status.*

ENTITY_ID

Поле `ENTITY_ID` для стадий смарт-процессов имеет следующий вид: `DYNAMIC_{entityTypeId}_STAGE_{categoryId}`,

где:

- `{entityTypeId}` - идентификатор типа CRM смарт-процесса;
- `{categoryId}` - идентификатор направления, к которому относится стадия.

STATUS_ID

Поле `STATUS_ID` для стадий смарт-процессов должно имет префикс `DT{entityTypeId}_{categoryId}`,

где

- `{entityTypeId}` - идентификатор типа CRM смарт-процесса
- `{categoryId}` - идентификатор направления, к которому относится стадия

Пример

`crm.status.add({fields})`

Для создания новой стадии смарт-процесса с идентификатором 135 в направлении с идентификатором 20 параметр `fields` должен иметь следующий вид:

```
{
  "fields": {
    "COLOR": "#1111AA",
    "NAME": "My new stage",
    "SORT": 250,
    "ENTITY_ID": "DYNAMIC_135_STAGE_20",
    "STATUS_ID": "DT135_20:MY_STAGE_REST",
  }
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Настройки карточки элемента смарт-процесса

Методы для управления настройками карточки элемента смарт-процесса.

Работают абсолютно идентично существующим методам crm.deal.details.configuration.*, crm.contact.details.configuration.* и т.д.

Есть пара отличий:

- Обязательно надо указать параметр `entityTypeId`, где должен быть указан идентификатор типа смарт-процесса.
- В параметре `extras` по ключу `categoryId` можно передать идентификатор направления смарт-процесса. Если он не передан, то будет использовано направление по умолчанию.

В остальном входные и выходные данные идентичны.

Методы носят названия:

- `crm.item.details.configuration.get`
- `crm.item.details.configuration.set`
- `crm.item.details.configuration.reset`
- `crm.item.details.configuration.forceCommonScopeForAll`

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События над элементами смарт-процессов

Есть три типа события - на добавление, изменение и удаление элемента смарт-процесса.

События срабатывают при действиях над элементами ВСЕХ смарт-процессов. Фильтровать по нужному типу придется уже на стороне обработки события.

В списке доступных событий будет следующий набор:

- `onCrmDynamicItemAdd` - добавление элемента любого смарт-процесса.
- `onCrmDynamicItemUpdate` - изменение элемента любого смарт-процесса.
- `onCrmDynamicItemDelete` - удаление элемента любого смарт-процесса.

Существует теоретическая возможность подписаться на события элементов конкретного смарт-процесса. Имена таких событий выглядят как `onCrmDynamicItemAdd_{entityTypeId}`. В интерфейсе создания вебхуков они не выводятся, но могут работать при подписке через приложения

В обработчик события придут данные в следующем виде:

```
[
    'FIELDS' => [
        'ID' => $itemId,
        'ENTITY_TYPE_ID' => $entityTypeId,
```

```
        ],
    ]
```

где `$itemId` - идентификатор элемента, а `$entityTypeId` - идентификатор типа CRM.

События над смарт-процессами

События:

- `onCrmTypeAdd` - при создании нового смарт-процесса.
- `onCrmTypeUpdate` - при изменении настроек смарт-процесса.
- `onCrmTypeDelete` - при удалении смарт-процесса.

В обработчик события придут данные в следующем виде:

```
[
    'FIELDS' => [
        'ID' => $id,
    ],
]
```

где `$id` - идентификатор смарт-процесса (первичный ключ, а не идентификатор типа)..

crm.type.add

```
crm.type.add({fields: {}})
```

Метод создаст новый смарт-процесс.

`fields` может содержать `entityTypeId` - это поле можно указать вручную при создании. Если оно не будет передано, то идентификатор будет сгенерирован автоматически.

Подробный состав `fields` указан в описании метода crm.type.update.

Метод вернет результат аналогичный вызову метода crm.type.get на только что созданном смарт-процессе.

Параметры

Параметр	Описание
fields	Состоит из полей настроек смарт-процесса, а также настроек связей, привязок к пользовательскому полю и настройки показа вне CRM.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.type.delete

```
crm.type.delete({id: number})
```

Метод удалит существующие настройки смарт-процесса с идентификатором `id`.

Удаление смарт-процесса возможно, только если к нему нет ни одного привязанного элемента.

Если есть элементы, необходимо сначала их удалить, а уже потом удалять смарт-процесс.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.type.fields

```
crm.type.fields()
```

Отдает информацию о собственных полях настроек смарт-процесса.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.type.get

```
crm.type.get({id: number})
```

Отдает информацию о смарт-процессе с идентификатором типа `id`.

Ответ:


```
{
    "type": {
      "id":  24,
      "title": "Смарт-процесс",
      "code": "",
      "createdBy": 1,
      "entityTypeId": 132,
      "isCategoriesEnabled": "Y",
      "isStagesEnabled": "Y",
      "isBeginCloseDatesEnabled": "Y",
      "isClientEnabled": "Y",
      "isUseInUserfieldEnabled": "Y",
      "isLinkWithProductsEnabled": "Y",
      "isCrmTrackingEnabled": "N",
      "isMycompanyEnabled": "Y",
      "isDocumentsEnabled": "Y",
      "isSourceEnabled": "Y",
      "isObserversEnabled": "Y",
      "isRecyclebinEnabled": "Y",
      "isAutomationEnabled": "Y",
```

```
        "isBizProcEnabled": "Y",
        "isSetOpenPermissions": "Y",
        "relations":  {
          "parent": [
            {
                "entityTypeId": 3,
                "isChildrenListEnabled": "Y",
                "isPredefined": "Y"
            }
          ],
          "child": []
        },
        "linkedUserFields": {
          "CALENDAR_EVENT|UF_CRM_CAL_EVENT":
"N",
          "TASKS_TASK|UF_CRM_TASK": "N",
          "TASKS_TASK_TEMPLATE|UF_CRM_TASK":
"N",
        },
        "customSections": [
          {
            "id": 18,
            "title": "Новый раздел",
            "isSelected": "N"
          }
        ]
      }
}
```

crm.type.list

```
crm.type.list({order: ?{} = null, filter: ?{} = null, start: ?number = 0})
```

Метод вернет массив настроек смарт-процессов

```
{
    "types": []
}
```

где каждый элемент массива - это структура, аналогичная ответу на запрос crm.type.get, за исключением связанных данных.

> Этот метод возвращает только собственные поля настроек смарт-процессов.

Параметры

Параметр	Описание
order	Список для сортировки, где ключ - поле, а значение - `ASC` или `DESC`.

filter	Список для фильтрации. Примеры фильтров ниже.
start	Сдвиг для постраничной навигации.

Примеры запросов

Найти все элементы, у которых включено поле "Клиент" с сортировкой по названию.

```
{
    "order": {
        "title": "ASC"
    },
        "filter": {
        "isClientEnabled": "Y"
    }
}
```

Фильтр поддерживает сложную логику.

crm.type.update

Описание и параметры

```
crm.type.update({id: number, fields: {}})
```

Метод обновит существующие настройки смарт-процесса с идентификатором `id`.

Параметры

Параметр	Описание
id	Идентификатор смарт-процесса.
fields	Состоит из полей настроек смарт-процесса, а также настроек связей, привязок к пользовательскому полю и настройки показа вне CRM.

Изменить только собственные поля настроек смарт-процесса

Чтобы изменить только поля настроек смарт-процесса необходимо передать только изменяемые значения полей. Например, необходимо отключить функционал печати документов.

```
{
  "id": 128,
  "fields": {
    "isDocumentsEnabled": "N"
  }
}
```

Изменить настройки связей

Подробнее о связях можно прочитать тут.

Чтобы изменить настройки связей смарт-процесса надо передать данные по ключу `relations`.

- Настройки необходимо передавать целиком, они полностью перезаписываются.
- Нельзя изменить настройки предустановленных связей (`iPredefined: true`). Эти настройки можно не передавать в запросе.

Включение комплексного поля "Клиент" создает предустановленные привязки к Компании и Контактам.

```
{
  "id": 128,
  "fields": {
    "relations": {
      "parent": [
        {
          "entityTypeId": 130
        }
      ],
      "child": [
        {
          "entityTypeId": 130,
          "isChildrenListEnabled": "N"
        }
```

```
        ]
      }
    }
  }
```

Если не передать настройку `isChildrenListEnabled`, то по умолчанию туда запишется `false`. Ключ `iPredefined` можно не передавать.

Если при попытке сохранить переданные настройки связей возникнет ошибка, она не будет выведена. Настройки просто не сохранятся.

Изменить настройки показа вне CRM

Про показ вне CRM можно прочитать тут.

Набор дополнительных разделов необходимо передавать целиком.

Допустим, на текущий момент набор разделов выглядит следующим образом (в ответ на `crm.type.get`):

```
{
  "type": {
    "customSections": [
      {
        "id": 18,
        "title": "Manufacturing",
        "iSelected": "N"
      },
      {
        "id": 20,
        "title": "HR",
        "iSelected": "N"
      }
    ]
```

```
    }
  }
```

Необходимо раздел "HR" поставить на первое место, раздел "Manufacturing" удалить, а вместо него создать новый "Consuming", к которому сразу привязать смарт-процесс.

В этом случае запрос будет выглядеть следующим образом:

```
{
  "id": 128,
  "fields": {
    "customSections": [
      {
        "id": 20,
        "title": "HR"
      },
      {
        "id": "new_1",
        "title": "Consuming"
      }
    ],
    "customSectionId": "new_1"
  }
}
```

crm.item.add

```
crm.item.add({entityTypeId: number, fields:
?{})
```

Метод создает новый элемент смарт-процесса с идентификатором `entityTypeId`.

При создании элемента производится стандартный ряд проверок, модификаций и автоматических действий:

- проверяются права доступа;
- проверяется заполненность обязательных полей;
- проверяется заполненность зависимых от стадий обязательных полей;
- проверяется корректность заполнения полей;
- полям присваиваются значения по умолчанию;
- после сохранения запускаются роботы.

Метод вернет результат аналогичный вызову метода crm.item.get на только что созданном элементе.

Чтобы загрузить файл, в качестве значения пользовательского поля необходимо передать массив, где первый элемент - это имя файла, а второй - это закодированный в base64 контент файла.

Примеры `fields` для разных запросов можно посмотреть здесь.

Параметры

Параметр	Описание
entityTypeId	Идентификатор смарт-процесса.
fields	Значение полей элемента. Необязательный параметр.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.item.delete

```
crm.item.delete({entityTypeId: number, id:
number})
```

Удаляет элемент с идентификатором `id` смарт-процесса с идентификатором `entityTypeId`.

Параметры

Параметр	Описание
entityTypeId	Идентификатор смарт-процесса.
id	Идентификатор элемента.

© «Битрикс», 2001-2008, «1С-

crm.item.fields

```
crm.item.fields({entityTypeId: number})
```

Отдает информацию о полях смарт-процесса с идентификатором `entityTypeId`.

В этом методе учитываются настройки смарт-процесса.

Названия полей для входных данных следует брать из этого метода.

Параметры

Параметр	Описание
entityTypeId	Идентификатор смарт-процесса.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.item.get

```
crm.item.get({entityTypeId: number, id:
number})
```

Отдает информацию об элементе с идентификатором `id` смарт-процесса с идентификатором `entityTypeId`.

Параметры

Параметр	Описание
entityTypeId	Идентификатор смарт-процесса.
id	Идентификатор элемента.

Ответ

```
{
    "item": {
        "id": 28,
        "xmlId": "",
        "title": "Название элемента",
        "createdBy": 1,
```

```
        "updatedBy": 1,
        "movedBy": 1,
        "createdTime": "2021-03-
09T02:00:00+02:00",
        "updatedTime": "2021-03-
29T02:00:00+02:00",
        "movedTime": "2021-03-
25T02:00:00+02:00",
        "categoryId": 17,
        "opened": "Y",
        "stageId": "DT132_17:CLIENT",
        "previousStageId":
"DT132_17:PREPARATION",
        "begindate": "2021-03-
09T02:00:00+02:00",
        "closedate": "2021-03-
16T02:00:00+02:00",
        "companyId": 36,
        "contactId": 1,
        "opportunity": 120,
        "isManualOpportunity": "N",
        "taxValue": 20,
        "currencyId": "RUB",
        "mycompanyId": 7,
        "sourceId": "RC_GENERATOR",
        "sourceDescription": "",
        "assignedById": 2,
        "ufCrm24_1612951135": 123
    }
}
```


Здесь

- `xmlId` - внешний код элемента;
- `title` - название элемента;
- `stageId` - идентификатор стадии, на которой находится элемент;

- `previousStageId` - идентификатор предыдущей стадии элемента;
- `createdBy` - идентификатор пользователя, создавшего элемент;
- `updatedBy` - идентификатор пользователя, изменившего элемент;
- `movedBy` - идентификатор пользователя, изменившего стадию элемента;
- `createdTime` - время создания элемента;
- `updatedTime` - время изменения элемента;
- `movedTime` - время изменения стадии элемента;
- `categoryId` - идентификатор направления смарт-процесса;
- `ufCrm...` - значения пользовательских полей:
    - значения множественных полей отдаются в виде массива;
    - значение поля типа "файл" отдаются в виде списка:
        - `id` - идентификатор;
        - `url` - ссылка на файл на портале;
        - `urlMachine` - ссылка на файл для приложения.

crm.item.list

Описание и параметры

```
crm.item.list({entityTypeId: number,
select:?[] = ['*'], order: ?{} = null,
filter: ?{} = null, start: ?number = 0})
```

Метод вернет массив элементов смарт-процесса с идентификатором `entityTypeId`:

```
{
    "items": []
}
```

где каждый элемент массива - это структура, аналогичная ответу на запрос crm.item.get.

Параметры

Параметр	Описание
entityTypeId	Идентификатор смарт-процесса.
select	Массив имен полей для выборки.

order	Список для сортировки, где ключ - поле, а значение - `ASC` или `DESC`.
filter	Список для фильтрации. Примеры фильтров ниже. **Примечание:** Фильтр не поддерживает фильтрацию по родительским полям. Для поиска необходимо использовать методы работы со связями.
start	Сдвиг для постраничной навигации.

Обработка select

Параметр **select** может содержать в себе только названия полей для этого смарт-процесса или `'*'`.

По умолчанию будет произведена выборка всех полей. Аналогичное поведение будет, если в списке полей присутствует `'*'`.

Примеры:

- Будут выбраны все поля.

```
{
    "select": ["*", "title", "id"],
}
```

- Будет выбрано только поле **id**. Названия полей надо передавать в явном виде или `'*'`.

```
{
    "select": ["id", "uf_*"],
}
```

- Будут выбраны поля **id**, **title** и **categoryId**.

```
{
    "select": ["id", "title",
"categoryId"],
}
```

Примеры фильтра

1. Найти элементы, у которых ответственным является пользователь с идентификатором 4

```
{
    "filter": {
        "=assignedById": "4"
    }
}
```

2. Найти элементы, находящиеся на финальных стадиях, доступные для всех, либо у которых ответственным является пользователь с идентификатором 4

```
{
    "filter": {
        "@stageId": ["DT132_17:SUCCESS",
"DT132_17:FAIL"],
        "0": {
            "logic": "OR",
            "0": {
                "=assignedById": 4
            },
```

```
            "1": {
                "=opened": "Y"
            }
        }
    }
}
```

3. Найти элементы, у которых заполнено пользовательское поле с кодом `ufCrm24_1616150749`

```
{
    "filter": {
        "!=ufCrm24_1616150749": ""
    }
}
```

4. Найти элементы, которые были созданы, изменены и сдвинуты в период с 19.03 по 22.03

```
{
    "filter": {
        ">createdTime":"2020-03-
19T02:00:00+02:00",
        ">movedTime":"2020-03-
19T02:00:00+02:00",
        ">updatedTime":"2020-03-
19T02:00:00+02:00",
        "<createdTime":"2020-03-
22T02:00:00+02:00",
        "<movedTime":"2020-03-
22T02:00:00+02:00",
        "<updatedTime":"2020-03-
22T02:00:00+02:00"
```

```
    }
}
```

5. Найти элементы, которые были или созданы, или изменены или сдвинуты в период с 19.03 по 22.03

```
{
    "filter": {
        "logic":"OR",
        "0":{
              ">createdTime":"2020-03-
19T02:00:00+02:00",
              "<createdTime":"2020-03-
22T02:00:00+02:00"
        },
        "1":{
              ">movedTime":"2020-03-
19T02:00:00+02:00",
              "<movedTime":"2020-03-
22T02:00:00+02:00"
        },
        "2":{
              ">updatedTime":"2020-03-
19T02:00:00+02:00",
              "<updatedTime":"2020-03-
22T02:00:00+02:00"
        }
    }
}
```

crm.item.update

Описание и параметры

```
crm.item.update({entityTypeId: number, id:
number, fields: {})
```

Метод обновит элемент с идентификатором `id` смарт-процесса с идентификатором `entityTypeId`.

При обновлении элемента производится стандартный ряд проверок, модификаций и автоматических действий:

- проверяются права доступа;
- проверяется заполненность обязательных полей, если изменена стадия элемента в рамках того же направления;
- проверяется заполненность зависимых от стадий обязательных полей, если изменена стадия элемента в рамках того же направления;
- проверяется корректность заполнения полей;
- полям присваиваются значения по умолчанию;
- **если перед сохранением оказывается, что никакие значения полей не были изменены, то сохранение не производится**;
- **после сохранения запускаются роботы**.

Метод вернет результат аналогичный вызову метода crm.item.get для обновленного элемента.

Параметры

Параметр	Описание
entityTypeId	Идентификатор смарт-процесса.
id	Идентификатор элемента.
fields	Значение полей элемента.

Загрузить новый файл вместо старого (не множественное поле)

Чтобы заменить файл в не множественном поле, просто загрузите новый файл. Старый будет удален автоматически.

```
{
    "fields": {
        "ufCrm1617027453943": [
             "myfile.pdf",

"...base64_encoded_file_content..."
        ]
    }
}
```

Удалить значение пользовательского поля типа файл

Для этого достаточно передать пустую строку (`''`) вместо значения

Оставить значение не множественного поля типа файл без изменений

Самый простой вариант - не добавлять в `fields` ключ с этим полем. Но если надо и передать, и не изменить, то в качестве значения надо передать список, где по ключу `id` будет идентификатор файла

```
{
    "fields": {
        "ufCrm1617027453943": {
            "id": 433
        }
    }
}
```

Если в `id` передать отличное от текущего значения, то значение поля обнулится и файл будет стёрт.

Работа с множественным полем типа файл

Значение множественного поля - это массив. Каждый элемент массива подчиняется тем же правилам, что и для не множественных значений.

Частичная перезапись значения множественного поля типа файл

Например, сейчас в множественном поле типа файл находится значение `[12, 255, 44]`.

Необходимо оставить файлы `12` и `44`, а вместо `255` загрузить новый

Запрос должен выглядеть следующим образом:

```
{
    "fields": {
```

```
        "ufCrm1617027453943": [
                {
                    "id": 12
                },
                {
                    "id": 44
                },
                [
                     "myNewFile.pdf",

"...base64_encoded_file_content..."
                ]
        ]
    }
}
```

crm.item.import

```
crm.item.import({entityTypeId: number,
fields: ?{})
```

Импорт одной записи.

Параметры

Параметр	Описание
entityTypeId	идентификатор типа элемента (Какие сущности поддерживаются?)
fields	значения полей элемента. Узнать набор этих полей можно используя REST метод crm.item.fields, либо использовать методы для конкретных типов: crm.lead.fields, crm.deal.fields, crm.contact.fields, crm.company.fields, crm.quote.fields.

Метод вернет массив `item` с идентификатором созданного элемента в случае успеха, либо сообщение об ошибке.

Чтобы загрузить файл, в качестве значения пользовательского поля необходимо передать массив, где первый элемент - это имя файла, а второй - это закодированный в base64 контент файла.

Более подробно об отличиях логики импорта от логики обычного добавления элементов написано тут.

Пример

Импорт сделки:

```
{
        "entityTypeId": 2,
        "fields": {
                "title": "Моя сделка",
                "categoryId": 0
        }
}
```

Успешный результат:

```
{
"result": {
        "item": {
                "id": 15
        }
}
```

crm.item.batchImport

```
crm.item.batchImport({entityTypeId: number,
$data: ?{})
```

Групповой импорт записей.

Параметры

Параметр	Описание
entityTypeId	идентификатор типа элемента (Какие сущности поддерживаются?)
data	массив значений полей элементов. Можно рассматривать его как массив, каждый элемент которого содержит набор полей `fields`, описанный в методе crm.item.import.

Метод вернет массив `data`, содержащий те же ключи, которые были в массиве `data` запроса. Каждый элемент этого массива будет содержать результат импорта конкретного элемента: массив `item` с идентификатором созданного элемента в случае успеха, либо сообщение об ошибке.

Логика добавления элементов работает по аналогии с методом crm.item.import.

Внимание! В одном запросе допустимо импортировать максимум 20 элементов.

Пример

Импорт сделок:

```
{
        "entityTypeId": 2,
        "data": [
                {
                        "title": "Моя
сделка",
                        "categoryId": 0
                },
                {
                        "title": ""
                }
        ]
}
```

Пример результата:

```
{
        "result": {
                "items": [
                        {
                                "item": {
"id": 15
                                }
                        },
                        {
                                "error":
"CRM_FIELD_ERROR_REQUIRED",
"error_description": "Поле \"Название\"
```

```
обязательно для заполнения"
                                }
                        ]
                }
}
```

crm.timeline.bindings.bind

Описание

```
crm.timeline.bindings.bind()
```

Привязывает запись в таймлайне к элементу crm.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array ("поле"=>"значение"[, ...]), содержащий значения полей. Обязательные поля - `OWNER_ID`, `ENTITY_ID` и `ENTITY_TYPE`.

Поля

Поле	Описание	Тип	Примечание
OWNER_ID	ID записи в таймлайне.	integer	Неизменяемое\Обязательно
ENTITY_ID	ID элемента, к которому привязана записи в таймлайне.	integer	Неизменяемое\Обязательно

ENTITY_TYPE	Тип элемента, к которому привязан комментарий. Значения: ▪ `lead` - лид; ▪ `deal` - сделка; ▪ `contact` - контакт; ▪ `company` - компания; ▪ `order` - заказ.	string	Неизменяемое\Обязательно

Пример

```
    BX24.callMethod(
        "crm.timeline.bindings.bind",
        {
            fields:
            {
                "ENTITY_ID": 10,
                "ENTITY_TYPE": "deal",
                "OWNER_ID": 1110
            }
        },
        function(result)
        {
            if(result.error())

console.error(result.error());
            else
                console.info(result.data());
```

```
            }
        );
```

crm.timeline.bindings.fields

Описание, параметры и пример

```
crm.timeline.bindings.fields()
```

Возвращает список полей привязки элементов crm к записям в таймлайне.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array ("поле"=>"значение"[, ...]), содержащий значения полей. Обязательные поля - `OWNER_ID`, `ENTITY_ID` и `ENTITY_TYPE`.

Пример

```
BX24.callMethod(
    "crm.timeline.bindings.fields",
    {},
```

```
        function(result)
        {
            if(result.error())

console.error(result.error());
            else
                console.dir(result.data());
        }
    );
```

Поля

Поле	Описание	Тип	Примечание
OWNER_ID	ID записи в таймлайне.	integer	Неизменяемое\Обязательно
ENTITY_ID	ID элемента, к которому привязана записи в таймлайне.	integer	Неизменяемое\Обязательно
ENTITY_TYPE	Тип элемента, к которому привязан комментарий. Значения: ▪ `lead` - лид; ▪ `deal` - сделка; ▪ `contact` - контакт; ▪ `company` - компания; ▪ `order` - заказ.	string	Неизменяемое\Обязательно

crm.timeline.bindings.list

```
crm.timeline.bindings.list()
```

Возвращает список привязок к записи в таймлайне.

При фильтрации обязательное поле - [dw]OWNER_ID[/dw][di]

Поле	Описание	Тип	Примечание
OWNER_ID	ID записи в таймлайне.	integer	Неизменяемое\Обязательное

[/di].

Параметры

См. описание списочных методов.

Пример:

```
BX24.callMethod(
    "crm.timeline.bindings.list",
    {
        filter: {
            "OWNER_ID": 1110
        }
```

```
        },
        function(result)
        {
            if(result.error())

console.error(result.error());
            else
            {
                console.dir(result.data());
                if(result.more())
                   result.next();
            }
        }
    );
```

crm.timeline.bindings.unbind

Описание

```
crm.timeline.bindings.unbind()
```

Снимает привязку записи таймлайна с элемента crm.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array ("поле"=>"значение"[, ...]), содержащий значения полей. Обязательные поля - `OWNER_ID`, `ENTITY_ID` и `ENTITY_TYPE`.

Поля

Поле	Описание	Тип	Примечание
OWNER_ID	ID записи в таймлайне.	integer	Неизменяемое\Обязательно
ENTITY_ID	ID элемента, к которому	integer	Неизменяемое\Обязательно

	привязана записи в таймлайне.		
ENTITY_TYPE	Тип элемента, к которому привязан комментарий. Значения: ▪ `lead` - лид; ▪ `deal` - сделка; ▪ `contact` - контакт; ▪ `company` - компания; ▪ `order` - заказ.	string	Неизменяемое\Обязательно

Пример

```
    BX24.callMethod(
        "crm.timeline.bindings.unbind",
        {
            fields:
            {
                "ENTITY_ID": 10,
                "ENTITY_TYPE": "deal",
                "OWNER_ID": 1110
            }
        },
        function(result)
        {
            if(result.error())

console.error(result.error());
```

```
            else
                console.info(result.data());
        }
    );
```

crm.timeline.comment.add

```
crm.timeline.comment.add(fields)
```

Добавляет новый комментарий в таймлайн.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array ("поле"=>"значение" значения полей. Обязательные поля - поле [dw]ENTIT

Поле	Описание	Тип	Пр
ENTITY_ID	ID элемента, к которому привязана записи в таймлайне.	integer	Неизменя

[/di] и [dw]ENTITY_TYPE[/dw][di]

Поле	Описание	Тип	Пр
ENTITY_TYPE	Тип элемента, к которому привязан комментарий. Значения:	string	Неизменя

			▪ `lead` - лид; ▪ `deal` - сделка; ▪ `contact` - контакт; ▪ `company` - компания; ▪ `order` - заказ.		

[/di].
Примечание: чтобы узнать требуемый формат полей crm.timeline.comment.fields и посмотрите формат приш этих полей.

Пример

```
   BX24.callMethod(
       "crm.timeline.comment.add",
       {
           fields:
           {
               "ENTITY_ID": 10,
               "ENTITY_TYPE": "deal",
               "COMMENT": "New comment was
added"
           }
       },
       function(result)
       {
           if(result.error())

console.error(result.error());
           else
               console.info("Добавлен новый
```

```
комментарий. ID - " + result.data());
        }
    );
```

crm.timeline.comment.delete

```
crm.timeline.comment.delete(id)
```

Удаляет сообщение.

Ограничение: если сообщение имеет привязки к нескольким элементам, необходимо сначала снять лишние привязки через crm.timeline.bindings.unbind.

Параметры

Параметр	Описание
id	Идентификатор сообщения в таймлайне.

Пример:

```
var id = prompt("Введите ID");
BX24.callMethod(
    "crm.timeline.comment.delete",
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error());
            else
                console.info(result.data());
        }
    );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.timeline.comment.fields

Описание и пример

```
crm.timeline.comment.fields()
```

Возвращает список полей комментария таймлайна.

Параметры

Без параметров.

Пример

```
   BX24.callMethod(
       "crm.timeline.comment.fields",
       {},
       function(result)
       {
           if(result.error())

console.error(result.error());
           else
               console.dir(result.data());
```

```
            }
    );
```

Поля

Поле	Описание	Тип	Примеча
ID	Идентификатор.	integer	Только для чтени
CREATED	Дата добавления.	datetime	Только для чтени
ENTITY_ID	ID элемента, к которому привязан комментарий.	integer	Неизменяемое\О
ENTITY_TYPE	Тип элемента, к которому привязан комментарий. Значения: ▪ `lead` - лид; ▪ `deal` - сделка; ▪ `contact` - контакт; ▪ `company` - компания; ▪ `order` - заказ.	string	Неизменяемое\О
AUTHOR_ID	Автор.	integer	Неизменяемое
COMMENT	Текст комментария.	string	

FILES	Список файлов.	attached_diskfile	Массив значений описанный по пр приведенным тут

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.timeline.comment.get

```
crm.timeline.comment.get(id)
```

Возвращает информацию о сообщении.

Параметры

Параметр	Описание
id	Идентификатор сообщения в таймлайне.

Пример:

```
    var id = prompt("Введите ID");
    BX24.callMethod(
        "crm.timeline.comment.get ",
        { id: id },
        function(result)
        {
            if(result.error())

console.error(result.error());
            else
                console.dir(result.data());
```

```
            }
    );
```

crm.timeline.comment.list

```
crm.timeline.comment.list()
```

Возвращает список всех сообщений для определенного crm элемента.

При фильтрации обязательные поля - [dw]ENTITY_ID[/dw][di]

Поле	Описание	Тип	Примечание
ENTITY_ID	ID элемента, к которому привязана записи в таймлайне.	integer	Неизменяемое\Обязательное

[/di] и [dw]ENTITY_TYPE[/dw][di]

Поле	Описание	Тип	Примечание
ENTITY_TYPE	Тип элемента, к которому привязан комментарий. Значения: ▪ `lead` - лид; ▪ `deal` - сделка; ▪ `contact` - контакт;	string	Неизменяемое\Обязательное

	▪ `company` - компания; ▪ `order` - заказ.		

[/di]..

При выборке используйте маску "*" для выборки всех полей (без пользовательских и множественных).

Параметры

См. описание списочных методов.

Пример:

```
   BX24.callMethod(
       "crm.timeline.comment.list",
       {
           filter: {
               "ENTITY_ID": 10,
               "ENTITY_TYPE": "deal",
           },
           select: [ "ID", "COMMENT ",
"FILES"]
       },
       function(result)
       {
           if(result.error())

console.error(result.error());
           else
           {
               console.dir(result.data());
               if(result.more())
```

```
                        result.next();
                }
            }
    );
```

crm.timeline.comment.update

```
crm.timeline.comment.update(id)
```

Обновляет сообщение.

Параметры

Параметр	Описание
id	Идентификатор сообщения в таймлайне.

Пример

```
    var id = prompt("Введите ID");
    BX24.callMethod(
        "crm.timeline.comment.update",
        {
            id: id,
            fields:
            {
                "COMMENT": "Comment was
changed"
            }
        },
```

```
        function(result)
        {
            if(result.error())

console.error(result.error());
            else
                console.info(result.data());
        }
    );
```

onCrmTimelineCommentAdd

Событие, вызываемое при добавлении нового комментария в таймлайне.

Параметры

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора комментария.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmTimelineCommentDelete

Событие, вызываемое при удалении нового комментария в таймлайне.

Параметры

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора комментария.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmTimelineCommentUpdate

Событие, вызываемое при обновлении нового комментария в таймлайне.

Параметры

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора комментария.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.activity.add

```
crm.activity.add(fields)
```

Создаёт новое дело.

Параметры

Параметр	Описание

fields	Массив вида array("поле"=>"значение"[, ...]), содержащий значения полей дел.
	Примечание: чтобы узнать требуемый формат полей, выполните метод crm.activity.fields и посмотрите формат пришедших значений этих полей.
	
	Имеется дополнительное поле DISABLE_SENDING_MESSAGE_COPY. Оно предназначено для принудительного отключения отправки копии сообщения адресату из MESSAGE_FROM. Если параметр не заполнен или указано любое значение отличное от 'Y' - копия отправлена будет. Пример:
	
	`[`
	
	`'fields'=>`
	`array (`
	`'SETTINGS'=>array`
	`('DISABLE_SENDING_MESSAGE_COPY'=>'Y')`

```
)
]
```

Варианты использования значений полей

Для дел типа e-mail:

- если письмо не должно быть отправлено, то следует установить `DIRECTION=2` и `COMPLETED='N'`;
- если необходимо пометить письма как завершенные, то следует выполнить обновление с выставлением флага завершения.

Пример

```
var paddatepart = function(part)
{
     return part >= 10 ? part.toString() :
'0' + part.toString();
};

var d = new Date();
d.setDate(d.getDate() + 7);
d.setSeconds(0);
var dateStr = d.getFullYear() + '-' +
paddatepart(1 + d.getMonth()) + '-' +
paddatepart(d.getDate()) + 'T' +
paddatepart(d.getHours()) + ':'
             + paddatepart(d.getMinutes()) +
':' + paddatepart(d.getSeconds()) +
'+00:00';

BX24.callMethod(
```

```
    "crm.activity.add",
    {
        fields:
        {
            "OWNER_TYPE_ID": 2, //из метода
crm.enum.ownertype: 2 - тип "сделка"
            "OWNER_ID": 102, //id сделки
            "TYPE_ID": 2, //из метода
crm.enum.activitytype
            "COMMUNICATIONS": [  {
VALUE:"+79832322323",
ENTITY_ID:134,ENTITY_TYPE_ID:3 } ], //где
134 - id контакта, 3 - тип "контакт"
            "SUBJECT": "Новый звонок",
            "START_TIME": dateStr,
            "END_TIME": dateStr,
            "COMPLETED": "N",
            "PRIORITY": 3, //из метода
crm.enum.activitypriority
            "RESPONSIBLE_ID": 1,
            "DESCRIPTION": "Важный звонок",
            "DESCRIPTION_TYPE": 3, //из
метода crm.enum.contenttype
            "DIRECTION": 2, // из метода
crm.enum.activitydirection
            "WEBDAV_ELEMENTS":
                                [
                                    { fileData:
document.getElementById('file1') }
                                ],
            "FILES":
                                [
                                    { fileData:
document.getElementById('file1') }
                                ] //после
установки модуля disk и конвертации данных
из webdav можно будет указавать FILES вместо
```

```
WEBDAV_ELEMENTS
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.info("Создан новый
звонок с ID " + result.data());
     }
);
```

crm.activity.communication.fie

```
crm.activity.communication.fields()
```

Возвращает описание коммуникации для активности.
Коммуникации хранят номера телефонов в звонках, email-адреса в письмах, имена во встречах.

Параметры

Без параметров

Пример

```
        BX24.callMethod(

"crm.activity.communication.fields",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else

console.dir(result.data());
```

```
                                        }
                                );
```

crm.activity.delete

```
crm.activity.delete(id)
```

Удаляет активность.

Параметры

Параметр	Описание
id	Идентификатор активности.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(

"crm.activity.delete",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
```

```
                                    else

console.info(result.data());
                            }
                    );
```

crm.activity.fields

```
crm.activity.fields()
```

Возвращает описание полей активности.

Параметры

Без параметров

Пример

```
BX24.callMethod(
"crm.activity.fields",
{},
function(result)
{
if(result.error())
console.error(result.error());
else
console.dir(result.data());
}
```

```
                    );
```

Поля

Поле	Описание	Тип
ASSOCIATED_ENTITY_ID	Идентификатор связанной с делом сущности	integer
AUTHOR_ID	Создатель дела	user
AUTOCOMPLETE_RULE	Автозаполнение	integer
BINDINGS	Привязки	crm_activity_binding
COMMUNICATIONS		crm_activity_communica
COMPLETED	Завершено	char
CREATED	Создано	datetime
DEADLINE	Срок исполнения	datetime
DESCRIPTION	Описание	string
DESCRIPTION_TYPE	Тип описания	crm.enum.contenttype

DIRECTION	Направление дела: входящее/ исходящее.	crm.enum.activitydirecti
EDITOR_ID	Кто изменил	user
END_TIME	Время завершения	datetime
FILES	Добавленные файлы	diskfile
ID	Идентификатор дела	integer
LAST_UPDATE	Дата последнего обновления	datetime
LOCATION	Местоположение.	string
NOTIFY_TYPE	Тип уведомлений	crm.enum.activitynotifyt
NOTIFY_VALUE		integer
ORIGINATOR_ID	Идентификатор источника данных	string
ORIGIN_ID	Идентификатор элемента в источнике данных	string
ORIGIN_VERSION	Оригинальная версия	string

OWNER_ID	Собственник	integer
OWNER_TYPE_ID	Тип собственника	crm.enum.ownertype
PRIORITY	Приоритет	crm.enum.activitypriority
PROVIDER_DATA		string
PROVIDER_GROUP_ID		string
PROVIDER_ID	Идентификатор провайдера	string
PROVIDER_TYPE_ID	Идентификатор типа провайдера	string
PROVIDER_PARAMS		object
RESPONSIBLE_ID	Ответственный	user
RESULT_CURRENCY_ID		string
RESULT_MARK		integer
RESULT_SOURCE_ID		string
RESULT_STATUS		integer

RESULT_STREAM	Статистика отчётов	integer
RESULT_SUM		double
RESULT_VALUE		double
SETTINGS	Настройки	object
START_TIME	Время начала выполнения	datetime
STATUS	Статус	crm_enum_activitystatus
SUBJECT	Субьект	string
TYPE_ID	Тип	crm_enum_activitytype
WEBDAV_ELEMENTS	Добавленные файлы	diskfile

crm.activity.get

```
crm.activity.get(id)
```

Возвращает активность по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор активности.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(
                        "crm.activity.get",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
```

```
console.dir(result.data());
                            }
                  );
```

crm.activity.list

Возвращает список активностей по фильтру. (Для получения COMMUNICATIONS его надо явно указать в select.) Является реализацией списочного метода для активностей.

Параметры

См. описание списочных методов.

Пример

В примере мы получаем список дел контакта с ИД 102.

```
        BX24.callMethod(
                        "crm.activity.list",
                        {
                                order:{
"ID": "DESC" },
                                filter:
                                {

"OWNER_TYPE_ID": 3,

"OWNER_ID": 102
                                },
                                select:[
"*", "COMMUNICATIONS" ]
                        },
                        function(result)
```

```
                        {

if(result.error())

console.error(result.error());
                                else
                                {

console.dir(result.data());

if(result.more())

result.next();
                                }
                        }
                );
```

crm.activity.update

```
crm.activity.update(id, fields)
```

Обновляет существующую активность.

Параметры

Параметр	Описание
id	Идентификатор активности.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.activity.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.activity.fields и посмотрите формат пришедших значений этих полей.

Пример

```
var d = new Date();
    d.setSeconds(0);
```

```
                var dateStr =
d.getFullYear() + '-' + paddatepart(1 +
d.getMonth()) + '-' +
paddatepart(d.getDate()) + 'T' +
paddatepart(d.getHours()) + ':'
                     +
paddatepart(d.getMinutes()) + ':' +
paddatepart(d.getSeconds()) + '+00:00';
                var paddatepart =
function(part)
                {
                     return part >= 10 ?
part.toString() : '0' + part.toString();
                }
                var id = prompt("Введите
ID");

                BX24.callMethod(

"crm.activity.update",
                        {
                                id: id,
                                fields:
                                {

"START_TIME": dateStr,

"END_TIME": dateStr,

COMPLETED: 'Y'
                                }
                        },
                        function(result)
                        {

if(result.error())
```

```
console.error(result.error());
                                else
                                {

console.info(result.data());
                                }
                        }
                );
```

crm.activity.type.add

Метод для регистрации своего подтипа дел с указанием ему названия и иконки.

Параметры

Параметр	Описание	С версии
TYPE_ID	Тип дела провайдера (при создании дела это PROVIDER_TYPE_ID)	
NAME	Название вашего типа дел	
ICON_FILE	Файл иконки вашего типа дел	

Пример

```
BX24.callMethod(
   'crm.activity.type.add',
   {
      fields:
         {
            "TYPE_ID": '1C',
            "NAME": "Дело 1с",
            'ICON_FILE':
```

```
document.getElementById('type-icon') // file
input node
         }
   },
   function(result)
   {
      if(result.error())
         alert("Error: " + result.error());
      else
      {
         alert("Success: " + result.data());
      }
   }
);
```

После этого достаточно при создании дела указывать свой тип, иконка и название будут подгружаться автоматически.

```
BX24.callMethod(
   'crm.activity.add',
   {
      fields:
         {
            "OWNER_TYPE_ID": 1,
            "OWNER_ID": selectedEntityId,
            "PROVIDER_ID": 'REST_APP',
            "PROVIDER_TYPE_ID": '1C',
            "SUBJECT": "Новое дело",
            "COMPLETED": "N",
            "RESPONSIBLE_ID": 1,
            "DESCRIPTION": "Описание нового
дела"
         }
   },
   function(result)
   {
```

```
        if(result.error())
           alert("Error: " + result.error());
        else
        {
           alert("Success: " + result.data());
        }
    }
);
```

crm.activity.type.delete

Метод для удаления подтипа дел.

Параметры

Параметр	Описание	С версии
TYPE_ID	Идентификатор типа дела провайдера	

Пример

```
BX24.callMethod(
   'crm.activity.type.delete',
   {
      TYPE_ID: id
   },
   function(result)
   {
      if(result.error())
         alert("Error: " + result.error());
      else
      {
         alert("Success: " + result.data());
      }
```

```
    }
);
```

crm.activity.type.list

Метод для получения списка подтипов дел.

Параметры

Без параметров

Пример

```
BX24.callMethod(
            'crm.activity.type.list',
            {

            },
            function(result)
            {
                if(result.error())
                    alert("Error: " +
result.error());
                else
                {

console.log(result.data());
                }
            }
        );
```

Создание дел из приложений

Приложения могут создавать дела с провайдером специального типа. У такого дела будет [dw]соответствующая иконка[/dw][di] [/di], оно будет отображаться в таймлайн и по клику на дело будет открываться приложение в слайдере с опциями в PLACEMENT_OPTIONS.

Изменять/удалять дела такого подтипа можно только в контексте приложение, которым оно создано. То есть при обновлении такого дела методом crm.activity.update через вебхук будет ошибка: `Access denied! Application context required`.

Параметры

Метод	Описание
PROVIDER_ID	Идентификатор провайдера. Для специального типа значение должно быть равно 'REST_APP'.
PROVIDER_TYPE_ID	Идентификатор типа дела. В случае использования провайдера 'REST_APP' разработчик может указывать произвольные идентификаторы типа в зависимости от своих задач.

Пример

```
<?php
header('Content-Type: text/html;
charset=UTF-8');
?>
<!DOCTYPE html>
<html>
<
        <meta charset="UTF-8">
        <title>
        <style type="text/css">

        </style>
</head>
<body style="display: none">
<script src="//api.bitrix24.com/api/v1/">
</script>

<?if (isset($_POST['PLACEMENT']) &&
!empty($_POST['PLACEMENT_OPTIONS'])):
        $opt =
json_decode($_POST['PLACEMENT_OPTIONS'],
true);
?>
<p>Activity ID: <?=(int)$opt['activity_id']?
></p>
<button oncl ick="updateActivity(<?=
(int)$opt['activity_id']?>);">Upd ate
activity (se t new description + completed)
<p><button oncl ick="deleteActivity(<?=
(int)$opt['activity_id']?>);">delete
activity</button>
<?else:?>
<button oncl ick="selectCRMEntity();">Select
LEAD</button>
<span id="selected-entity"></span>
<p>
<button oncl ick="addActivity();">Add
```

```
activity</button>
<?endif;?>
<script type="text/javascript">
        BX24.init(function()
        {
                document.body.style.display
= '';
        });

        var selectedEntityId = null;

        function addActivity()
        {

                if (!selectedEntityId)
                {
                        alert('Lead not
selected');
                        return;
                }
                BX24.callMethod(
                        'crm.activity.add',
                        {
                                fields:
                                        {

"OWNER_TYPE_ID": 1,

"OWNER_ID": selectedEntityId,

"PROVIDER_ID": 'REST_APP',

"PROVIDER_TYPE_ID": 'LINK',

"SUBJECT": "Новое дело",

"COMPLETED": "N",
```

```
"RESPONSIBLE_ID": 1,

"DESCRIPTION": "Описание нового дела"
                                        }
                        },
                        function(result)
                        {

if(result.error())

alert("Error: " + result.error());
                                else
                                {

alert("Success: " + result.data());
                                }
                        }
                );
        }
        function updateActivity(id)
        {
                BX24.callMethod(

'crm.activity.update',
                        {
                                id: id,
                                fields:
                                        {

COMPLETED: 'Y',

SUBJECT: "Дело выполнено!",

DESCRIPTION: "Описание нового дела
(выполнено)"
                                        }
```

```
                        },
                        function(result)
                        {

if(result.error())

alert("Error: " + result.error());
                                else
                                {

alert("Success: " + result.data());
                                }
                        }
                );
        }

        function deleteActivity(id)
        {
                BX24.callMethod(

'crm.activity.delete',
                        {
                                id: id
                        },
                        function(result)
                        {

if(result.error())

alert("Error: " + result.error());
                                else
                                {

alert("Success: " + result.data());
                                }
                        }
                );
```

```
        }

        function selectCRMEntity()
        {

document.getElementById('selected-
entity').textContent = '';
                BX24.selectCRM({
                        entityType: ['lead']
                }, function(selected)
                {
                        if (selected['lead']
&& selected['lead'][0])
                        {

document.getElementById('selected-
entity').textContent = selected['lead'][0]
['title'];
                                var     id =
selected['lead'][0]['id'];


selectedEntityId = id.substring(2);


console.log(selectedEntityId);
                        }
                })
        }
</script>
</body>
</html>
```

onCrmActivityAdd

Событие, вызываемое при создании дела.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора созданного дела.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmActivityDelete

Событие, вызываемое при удалении дела.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора удалённого дела.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmActivityUpdate

Событие, вызываемое при обновлении дела.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора обновлённого дела.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.activity.binding.add

```
crm.activity.binding.add({activityId:
number, entityTypeId: number, entityId:
number})
```

Добавление привязки.

Параметры

Параметр	Описание
activityId	идентификатор дела
entityTypeId	идентификатор типа элемента (Справочник доступных типов)
entityId	идентификатор элемента

В случае успеха, метод вернет `true`.

Привязка дела возможна только к элементу, к которому у текущего пользователя есть доступ на редактирование.

Пример

```
crm.activity.binding.add?
activityId=1&entityTypeId=4&entityId=1000
```

Успешный результат:

```
{
        "result": true
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.activity.binding.delete

```
crm.activity.binding.delete({activityId:
number, entityTypeId: number, entityId:
number})
```

Удаление привязки.

Параметры

Параметр	Описание
activityId	идентификатор дела
entityTypeId	идентификатор типа элемента (Справочник доступных типов)
entityId	идентификатор элемента

В случае успеха, метод вернет `true`.

Удаление привязки дела возможно только для элементов, к которому у текущего пользователя есть доступ на редактирование.

Если дело привязано только к одному элементу, удалить эту привязку нельзя.

Пример

```
crm.activity.binding.delete?
activityId=1&entityTypeId=4&entityId=1000
```

Успешный результат:

```
{
        "result": true
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.activity.binding.list

```
crm.activity.binding.list({activityId:
number})
```

Получить список привязок.

Параметры

Параметр	Описание
activityId	идентификатор дела

Метод вернет массив, элементами которого будут массивы, содержащие:

- `entityTypeId` - идентификатор типа элемента (Справочник доступных типов);
- `entityId` - идентификатор элемента.

Пример

```
crm.activity.binding.list?activityId=1
```

Успешный результат:

```
{
        "result": [
                {
                        "entityTypeId": 1,
                        "entityId": 123
                },
                {
                        "entityTypeId": 2,
                        "entityId": 456
                },
                {
                        "entityTypeId": 3,
                        "entityId": 789
                }
        ]
}
```

Поля реквизитов

Название	Описание	Чтен
	Общие реквизиты	
ID	Идентификатор реквизита. Создается автоматически и уникален в рамках БД.	Да
[dw]ENTITY_TYPE_ID[/dw] [di]Идентификаторы типов сущностей возвращает метод crm.enum.ownertype[/di]	Идентификатор типа родительской сущности. Возможные типы: "Компания", "Контакт". Обязательное поле.	Да
ENTITY_ID	Идентификатор родительской сущности. Обязательное поле.	Да
PRESET_ID	Идентификатор шаблона реквизитов. Обязательное поле.	Да
DATE_CREATE	Дата создания.	Да
DATE_MODIFY	Дата изменения.	Да
CREATED_BY_ID	Идентификатор создавшего реквизит.	Да
MODIFY_BY_ID	Идентификатор изменившего реквизит.	Да
NAME	Название реквизита.	Да

	Обязательное для общих реквизитов.	
CODE	Символьный код реквизита.	Да
XML_ID	Внешний ключ, используется для операций обмена. Идентификатор объекта внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да
ACTIVE	Признак активности.	Да
SORT	Сортировка.	Да
RQ_NAME	Ф.И.О.	Да
RQ_FIRST_NAME	Имя.	Да
RQ_LAST_NAME	Фамилия.	Да
RQ_COMPANY_NAME	Сокращенное наименование организации.	Да
RQ_COMPANY_FULL_NAME	Полное наименование организации.	Да
RQ_COMPANY_REG_DATE	Дата государственной регистрации.	Да
RQ_DIRECTOR	Ген. директор.	Да
RQ_ACCOUNTANT	Гл. бухгалтер.	Да
RQ_CEO_NAME	ФИО первого руководителя.	Да

RQ_CEO_WORK_POS	Должность первого руководителя.	Да
RQ_CONTACT	Контактное лицо.	Да
RQ_EMAIL	E-Mail.	Да
RQ_PHONE	Телефон.	Да
RQ_FAX	Факс.	Да
RQ_IDENT_DOC	Вид документа.	Да
RQ_IDENT_DOC_SER	Серия.	Да
RQ_IDENT_DOC_NUM	Номер.	Да
RQ_IDENT_DOC_DATE	Дата выдачи.	Да
RQ_IDENT_DOC_ISSUED_BY	Кем выдан.	Да
RQ_IDENT_DOC_DEP_CODE	Код подразделения.	Да
RQ_INN	ИНН.	Да
RQ_KPP	КПП.	Да
RQ_USRLE	Handelsregisternummer (для страны DE).	Да
RQ_IFNS	ИФНС.	Да
RQ_OGRN	ОГРН.	Да
RQ_OGRNIP	ОГРНИП.	Да
RQ_OKPO	ОКПО.	Да
RQ_OKTMO	ОКТМО.	Да
RQ_OKVED	ОКВЭД.	Да

RQ_EDRPOU	ЄДРПОУ.	Да
RQ_DRFO	ДРФО.	Да
RQ_KBE	КБЕ.	Да
RQ_IIN	ИИН.	Да
RQ_BIN	БИН.	Да
RQ_VAT_PAYER	Платник ПДВ (для страны UA).	Да
RQ_VAT_ID	VAT ID (идентификационный номер (плательщика) НДС).	Да
RQ_VAT_CERT_SER	Серия свидетельства по НДС.	Да
RQ_VAT_CERT_NUM	Номер свидетельства по НДС.	Да
RQ_VAT_CERT_DATE	Дата свидетельства по НДС.	Да
RQ_RESIDENCE_COUNTRY	Страна резидента.	Да
ORIGINATOR_ID	Идентификатор внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да
Банковские реквизиты		
ID	Идентификатор реквизита. Создается автоматически и уникален в рамках БД.	Да
[dw]ENTITY_TYPE_ID[/dw]	Идентификатор типа	Да

[di]Идентификаторы типов сущностей возвращает метод crm.enum.ownertype[/di]	родительской сущности. По умолчанию тип: "Реквизит". Обязательное поле.	
ENTITY_ID	Идентификатор родительской сущности. Обязательное поле.	Да
COUNTRY_ID	Идентификатор страны.	Да
DATE_CREATE	Дата создания.	Да
DATE_MODIFY	Дата изменения.	Да
CREATED_BY_ID	Идентификатор создавшего реквизит.	Да
MODIFY_BY_ID	Идентификатор изменившего реквизит.	Да
NAME	Название реквизита. Обязательное поле.	Да
CODE	Символьный код реквизита.	Да
XML_ID	Внешний ключ, используется для операций обмена. Идентификатор объекта внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да
ACTIVE	Признак активности.	Да
SORT	Сортировка.	Да
RQ_BANK_NAME	Наименование банка.	Да
RQ_BANK_ADDR	Адрес банка.	Да

RQ_BANK_ROUTE_NUM	Bank Routing Number.	Да
RQ_BIK	БИК.	Да
RQ_MFO	МФО.	Да
RQ_ACC_NAME	Bank Account Holder Name.	Да
RQ_ACC_NUM	Bank Account Number.	Да
RQ_IIK	ИИК.	Да
RQ_ACC_CURRENCY	Валюта счёта.	Да
RQ_COR_ACC_NUM	Кор. счёт.	Да
RQ_IBAN	IBAN.	Да
RQ_SWIFT	SWIFT.	Да
RQ_BIC	BIC.	Да
COMMENTS	Комментарий.	Да
ORIGINATOR_ID	Идентификатор внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да
Шаблоны реквизитов		
ID	Идентификатор реквизита. Создается автоматически и уникален в рамках БД.	Да
[dw]ENTITY_TYPE_ID[/dw] [di]Идентификаторы типов сущностей возвращает метод crm.enum.ownertype[/di]	Идентификатор типа родительской сущности. Обязательное поле.	Да

	Примечание. Идентификаторы типов сущностей CRM отдаёт метод crm.enum.ownertype.	
COUNTRY_ID	Страна, которой соответствует набор полей шаблона реквизита.	Да
DATE_CREATE	Дата создания.	Да
DATE_MODIFY	Дата изменения.	Да
CREATED_BY_ID	Идентификатор создавшего реквизит.	Да
MODIFY_BY_ID	Идентификатор изменившего реквизит.	Да
NAME	Название реквизита.	Да
XML_ID	Внешний ключ, используется для операций обмена. Идентификатор объекта внешней информационной базы. Назначение поля может меняться конечным разработчиком.	Да
ACTIVE	Признак активности.	Да
SORT	Сортировка.	Да
Поля шаблонов реквизитов		
ID	Идентификатор реквизита. Создается автоматически и уникален в рамках реквизита.	Да

FIELD_NAME	Название поля.	Да
FIELD_TITLE	Альтернативное название поля для реквизита.	Да
SORT	Сортировка.	Да
IN_SHORT_LIST	Показывать в кратком списке.	Да
Адреса реквизитов		
TYPE_ID	Идентификатор типа адреса. Обязательное поле. Элемент перечисления "Тип адреса". **Примечание**. Элементы перечисления "Тип адреса" возвращает метод crm.enum.addresstype.	Да
[dw]ENTITY_TYPE_ID[/dw] [di]Идентификаторы типов сущностей возвращает метод crm.enum.ownertype[/di]	Идентификатор типа родительской сущности. Возможные типы: "Реквизит", "Компания", "Контакт", "Лид". Обязательное поле. **Примечание**. Идентификаторы типов сущностей CRM отдаёт метод crm.enum.ownertype.	Да
ENTITY_ID	Идентификатор родительской сущности. Обязательное поле.	Да
ADDRESS_1	Улица, дом, корпус, строение.	Да

ADDRESS_2	Квартира / офис.	Да
CITY	Город.	Да
POSTAL_CODE	Почтовый индекс.	Да
REGION	Район.	Да
PROVINCE	Область.	Да
COUNTRY	Страна.	Да
COUNTRY_CODE	Код страны.	Да
ANCHOR_TYPE_ID	Родительская сущность, с которой связана текущая запись. **Примечание**. Для **Реквизита** родительской сущностью может быть только Контакт или Компания. Для **Банковского реквизита** родительской сущностью может быть только Реквизит. Адреса могут быть привязаны [dw]только[/dw][di]Для обратной совместимости оставлена возможность связывать Адреса с Контактами или Компаниями. Но эта связь возможна только на некоторых старых порталах, где специально техподдержкой был включен старый режим работы с адресами.[/di] к Реквизитам. **Примечание**. Эти поля для служебного использования. Они нужны для решения	Да
ANCHOR_ID		Да

	задачи производительности в выборках по старшим сущностям (Компаниям/Контактам) в элементах CRM, которые с ними напрямую не связаны, а связаны опосредовано через другой элемент CRM. При добавлении записи Адреса обязательно указывается непосредственная привязка адреса к старшей сущности - ENTITY_ID, [dw]ENTITY_TYPE_ID[/dw] [di]Идентификаторы типов сущностей возвращает метод crm.enum.ownertype[/di]. Например, это Реквизит. Но в поля записи Адреса ANCHOR_TYPE_ID и ANCHOR_ID автоматически проставится привязка к более старшей сущности самого Реквизита - Компании/Клиенту.	

crm.address.add

```
crm.address.add(fields)
```

Создаёт новый адрес для реквизита.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей адреса. Где "поле" может принимать значения из возвращаемых методом crm.address.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.address.fields и посмотрите формат пришедших значений этих полей.

Пример

```
BX24.callMethod(
    "crm.address.add",
    {
        fields:
```

```
        {
            "TYPE_ID": 1,
            "ENTITY_TYPE_ID": 3,
            "ENTITY_ID": 1,
            "ADDRESS_1": "Московский пр-т,
261",
            "CITY": "Калининград"
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.address.delete

```
crm.address.delete(fields)
```

Удаляет адрес для реквизита.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]). Где "поле" может принимать значения из возвращаемых методом crm.address.fields.

Пример

```
BX24.callMethod(
    "crm.address.delete",
    {
                        fields:
                        {

"TYPE_ID": 1,
```

```
"ENTITY_TYPE_ID": 3,

"ENTITY_ID": 1
                                }
                        },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else

console.info(result.data());
            }
        );
```

crm.address.fields

```
crm.address.fields()
```

Возвращает описание полей адреса.

Параметры

Без параметров.

Пример

```
    BX24.callMethod(
            "crm.address.fields",
            {},
            function(result)
            {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
            }
        );
```

crm.address.list

```
crm.address.list
```

Возвращает список адресов по фильтру. Является реализацией списочного метода для адресов.

Адреса перемещены в реквизиты (но в карточке CRM они имеют отображение в виде отдельного поля) - см. описание полей реквизитов. К сущности CRM могут быть привязаны несколько реквизитов. Внутри реквизита может быть несколько **Общих реквизитов** по шаблонам, **Банковских реквизитов** и **Адресов**.

Параметры

См. описание списочных методов.

Пример

```
    //Поиск адресов по привязке к типу -
Реквизит
        BX24.callMethod(
            "crm.address.list",
            {
                order: { "TYPE_ID": "ASC" },
                filter: { "ENTITY_ID": 8},
                select: [ "TYPE_ID",
```

```
"ADDRESS_1", "ADDRESS_2" ]
            },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else
                {

console.dir(result.data());
                    if(result.more())
                        result.next();
                }
            }
        );
```

Примеры ответов

- Если адрес привязан к сущности без реквизита (обычно создаётся при конвертации из Лида, у которого заполнен Адрес в карточке):

```
[result] => Array
       (
           [1] => Array
               (
                   [TYPE_ID] => 1
                   [ENTITY_TYPE_ID] => 3
                   [ENTITY_ID] => 17192
                   [ADDRESS_1] =>
                   [ADDRESS_2] =>
                   [CITY] =>
                   [POSTAL_CODE] =>
```

```
                        [REGION] =>
                        [PROVINCE] =>
                        [COUNTRY] =>
                        [COUNTRY_CODE] =>
                        [LOC_ADDR_ID] => 0
                        [ANCHOR_TYPE_ID] => 3
                        [ANCHOR_ID] => 17192
                    )
        )
```

- Если у контакта 2 разных реквизита, к которым привязаны адреса:

```
[result] => Array
        (
            [2] => Array
                (
                        [TYPE_ID] => 1
                        [ENTITY_TYPE_ID] => 8
                        [ENTITY_ID] => 7335
                        [ADDRESS_1] => Ленина
2
                        [ADDRESS_2] => 701
                        [CITY] => Тюмень
                        [POSTAL_CODE] =>
625003
                        [REGION] => Тюменская
обл
                        [PROVINCE] =>
Тюменская обл
                        [COUNTRY] => Россия
                        [COUNTRY_CODE] =>
                        [LOC_ADDR_ID] => 479
                        [ANCHOR_TYPE_ID] => 3
                        [ANCHOR_ID] => 17192
```

```
                )
        )
```

```
[result] => Array
        (
            [3] => Array
                (
                    [TYPE_ID] => 1
                    [ENTITY_TYPE_ID] => 8
                    [ENTITY_ID] => 8191
                    [ADDRESS_1] => Ленина
                    [ADDRESS_2] => 2
                    [CITY] => Тюмень
                    [POSTAL_CODE] =>
666000
                    [REGION] => Тюменская
область рег
                    [PROVINCE] =>
Тюменская область
                    [COUNTRY] => Россия
                    [COUNTRY_CODE] =>
                    [LOC_ADDR_ID] => 129
                    [ANCHOR_TYPE_ID] => 3
                    [ANCHOR_ID] => 17192
                )
        )
```

Поле **ANCHOR_TYPE_ID** заполнено значением из crm.enum.ownertype (в примере это Контакты), а поле **ANCHOR_ID** содержит ID сущности (в данном случае Контакта). Поля **ANCHOR_TYPE_ID** и **ANCHOR_ID** в двух вышеуказанных примерах одинаковы, следовательно, оба адреса относятся к одному Контакту.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.address.update

```
crm.address.update(fields)
```

Обновляет адрес для реквизита.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), содержащий значения полей адреса. Где "обновляемое поле" может принимать значения из возвращаемых методом crm.address.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.address.fields и посмотрите формат пришедших значений этих полей.

Пример

```
BX24.callMethod(
    "crm.address.update",
    {
        fields:
```

```
        {
            "TYPE_ID": 1,
            "ENTITY_TYPE_ID": 3,
            "ENTITY_ID": 1,
            "ADDRESS_1": "Московский
проспект, 261",
            "CITY": "Калининград"
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
    }
);
```

crm.requisite.add

```
crm.requisite.add(fields)
```

Создаёт новый реквизит.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей адреса. Где "поле" может принимать значения из возвращаемых методом crm.requisite.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.requisite.fields и посмотрите формат пришедших значений этих полей.

Пример

```
BX24.callMethod(
    "crm.requisite.add",
    {
        fields:
```

```
        {
            "ENTITY_TYPE_ID":4,
                        "ENTITY_ID":8,
                        "PRESET_ID":1,
                        "NAME":"Реквизит",
                        "XML_ID":"5e4641fd-
1dd9-11e6-b2f2-005056c00008",
                        "ACTIVE":"Y",
                        "SORT":100
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.info("Создан реквизит с
ID " + result.data());
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.requisite.delete

```
crm.requisite.delete(id)
```

Удаляет реквизит и все связанные с ним объекты.

Параметры

Параметр	Описание
id	Идентификатор реквизита.

Пример

```
   var id = prompt("Введите ID");
        BX24.callMethod(
            "crm.requisite.delete",
            { id: id },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else
```

```
console.info(result.data());
            }
        );
```

crm.requisite.fields

```
crm.requisite.fields()
```

Возвращает описание полей реквизита.

Параметры

Без параметров.

Пример

```
    BX24.callMethod(
            "crm.requisite.fields",
            {},
            function(result)
            {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
            }
        );
```

crm.requisite.get

```
crm.requisite.get(id)
```

Возвращает реквизит по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор реквизита.

Пример

```
    var id = prompt("Введите ID");
        BX24.callMethod(
            "crm.requisite.get",
            { id: id },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else
```

```
console.dir(result.data());
            }
        );
```

crm.requisite.list

```
crm.requisite.list()
```

Возвращает список реквизитов по фильтру. Является реализацией списочного метода для реквизитов. Полное описание полей реквизита можно получить вызовом метода crm.requisite.fields .

Параметры

См. описание списочных методов.

Пример

```
    //Поиск реквизитов по идентификатору
шаблона
        BX24.callMethod(
            "crm.requisite.list",
            {
                order: { "DATE_CREATE":
"ASC" },
                filter: { "PRESET_ID": "1"},
                select: [ "ID", "NAME"]
            },
            function(result)
            {
                if(result.error())
```

```
console.error(result.error());
                else
                {

console.dir(result.data());
                    if(result.more())
                        result.next();
                }
            }
        );
```

```
// Получение значения пользовательского поля
в реквизитах.
BX24.callMethod(
            "crm.requisite.list",
        {
            order: {},
            filter: { "ID": "2622"}, //ID
РЕВИЗИТА
            select: [ "UF_CRM_1564433257"]
//ID ПОЛЬЗОВАТЕЛЬСКОГО ПОЛЯ
        },
        function(result)
        {
            if(result.error())

console.error(result.error());
            else
            {
                console.dir(result.data());
                if(result.more())
                    result.next();
            }
```

```
            }
        );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.requisite.update

```
crm.requisite.update(id, fields, params)
```

Метод обновляет существующий реквизит.

Параметры функции

Параметр	Описание
id	Идентификатор реквизита.
fields	Набор полей - массив вида `array("обновляемое поле"=>"значение"[, ...])`, где "обновляемое поле" может принимать значения из возвращаемых методом crm.requisite.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.requisite.fields и посмотрите формат пришедших значений этих полей.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
    "crm.requisite.update",
    {
        id: id,
        fields:
        {
            "NAME": "Реквизит (архив)",
            "SORT" : 200,
            "ACTIVE": "N"
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
        {
            console.info(result.data());
        }
    }
);
```

crm.requisite.bankdetail.add

```
crm.requisite.bankdetail.add(fields)
```

Создаёт новый банковский реквизит.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей адреса. Где "поле" может принимать значения из возвращаемых методом crm.requisite.bankdetail.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.requisite.bankdetail.fields и посмотрите формат пришедших значений этих полей. Внимание! В поле ENTITY_ID указывается не ID контакта/компании, а ID реквизита.

Пример

```
BX24.callMethod(
    "crm.requisite.bankdetail.add",
    {
        fields:
        {
                        "ENTITY_ID":2,
                        "COUNTRY_ID":1,
                        "NAME":"Реквизит
банка",
                        "XML_ID":"1e4641fd-
2dd9-31e6-b2f2-105056c00008",
                        "ACTIVE":"Y",
                        "SORT":100
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.info("Создан банковский
реквизит с ID " + result.data());
    }
);
```

crm.requisite.bankdetail.delete

```
crm.requisite.bankdetail.delete(id)
```

Удаляет банковский реквизит.

Параметры

Параметр	Описание
id	Идентификатор банковского реквизита.

Пример

```
    var id = prompt("Введите ID");
        BX24.callMethod(

"crm.requisite.bankdetail.delete",
            { id: id },
            function(result)
            {
                if(result.error())

console.error(result.error());
```

```
                    else

console.info(result.data());
                }
            );
```

crm.requisite.bankdetail.fields

```
crm.requisite.bankdetail.fields()
```

Возвращает описание полей банковских реквизитов.

Параметры

Без параметров.

Пример

```
    BX24.callMethod(

"crm.requisite.bankdetail.fields",
            {},
            function(result)
            {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
            }
```

```
            );
```

crm.requisite.bankdetail.get

```
crm.requisite.bankdetail.get(id)
```

Возвращает банковский реквизит по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор реквизита.

Пример

```
    var id = prompt("Введите ID");
        BX24.callMethod(
            "crm.requisite.bankdetail.get",
            { id: id },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else
```

```
console.dir(result.data());
            }
        );
```

crm.requisite.bankdetail.list

```
crm.requisite.bankdetail.list()
```

Возвращает список банковских реквизитов по фильтру. Является реализацией списочного метода для реквизитов.

Параметры

См. описание списочных методов.

Пример

```
    //Поиск банковских реквизитов по
идентификатору страны
        BX24.callMethod(
            "crm.requisite.bankdetail.list",
            {
                order: { "DATE_CREATE":
"ASC" },
                filter: { "COUNTRY_ID":
"1"},
                select: [ "ID", "NAME"]
            },
            function(result)
```

```
            {
                if(result.error())

console.error(result.error());
                else
                {

console.dir(result.data());
                    if(result.more())
                        result.next();
                }
            }
        );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.requisite.bankdetail.updat

```
crm.requisite.bankdetail.update(id, fields,
params)
```

Обновляет существующий банковский реквизит.

Параметры

Параметр	Описание
id	Идентификатор реквизита.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.requisite.bankdetail.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.requisite.bankdetail.fields и посмотрите формат пришедших значений этих полей.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
    "crm.requisite.bankdetail.update",
    {
        id: id,
        fields:
        {
            "NAME": "Банковский реквизит
(архив)",
            "SORT" : 200,
            "ACTIVE": "N"
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
        {
            console.info(result.data());
        }
    }
);
```

crm.requisite.link.fields

```
crm.requisite.link.fields()
```

Возвращает описание полей для связей реквизитов.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(
"crm.requisite.link.fields",
                        {},
                        function(result)
                        {
if(result.error())
console.error(result.error());
                                else
console.dir(result.data());
```

```
                                        }
            );
```

crm.requisite.link.get

```
crm.requisite.link.get(entityTypeId,
entityId)
```

Возвращает связь реквизитов с сущностью.

Параметры

Параметр	Описание
entityTypeId	Идентификатор типа сущности (см. метод crm.enum.ownertype). Для связей реквизитов может быть только сделка, предложение или счёт.
entityId	Идентификатор сущности (сделки, предложения или счёта).

Пример

```
var id = prompt("Введите ID счёта");
        BX24.callMethod(
                "crm.requisite.link.get", {
                        entityTypeId: 5,
```

```
                        entityId: id
                },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
        );
```

crm.requisite.link.list

```
crm.requisite.link.list(order, filter)
```

Возвращает список связей реквизитов. Связи определяют, какие реквизиты выбраны для сделки, предложения или счёта. При этом реквизиты должны принадлежать выбранной компании или контакту. Так, если в счёте в качестве покупателя выбрана компания, то реквизиты покупателя должны принадлежать этой компании. В качестве продавца может быть выбрана только компания из справочника "Реквизиты ваших компаний". Каждая запись связей содержит идентификаторы:

- "[dw]ENTITY_TYPE_ID[/dw][di]Идентификаторы типов сущностей возвращает метод crm.enum.ownertype[/di]" - ключевое поле. Идентификатор типа сущности, для которой выбраны реквизиты. Возможные типы: "Сделка", "Предложение", "Счёт". Обязательное поле.
- "ENTITY_ID" - ключевое поле. Идентификатор сущности, для которой выбраны реквизиты. Обязательное поле.
- "REQUISITE_ID" - идентификатор реквизитов клиента (компании либо контакта).
- "BANK_DETAIL_ID" - идентификатор банковских реквизитов клиента.
- "MC_REQUISITE_ID" - идентификатор реквизитов компании продавца.
- "MC_BANK_DETAIL_ID" - идентификатор банковских реквизитов компании продавца.

  Если идентификатор равен 0, значит соответствующий реквизит не выбран для сделки, предложения или счёта.

Параметры

Параметр	Описание
order	Поля сортировки.
filter	Поля фильтра.

Пример

```
BX24.callMethod(
                "crm.requisite.link.list", {
                        order: {"ENTITY_ID":
"ASC"},
                        filter:
{"ENTITY_TYPE_ID": 5}    // Счёт
                },
                function (result)
                {
                        if (result.error())
                        {

console.error(result.error());
                        }
                        else
                        {

console.dir(result.data());
                                if
(result.more())

result.next();
                        }
                }
```

```
            );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.requisite.link.register

```
crm.requisite.link.register(fields)
```

Регистрирует связь реквизитов с сущностью. Для успешной регистрации идентификаторы реквизитов должны принадлежать клиенту и продавцу, которые выбраны в той сущности, для которой регистрируется связь. Если какого-то реквизита нет, то его идентификатор передаётся как 0. Можно даже указать все идентификаторы реквизитов нулевыми, тогда считается, что к сущности реквизиты не привязаны.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей связи реквизитов.

Пример

```
var entityId = prompt("Введите ID счёта");
        var requisiteId = prompt("Введите ID
реквизита, принадлежащего покупателю");
```

```
        var bankDetailId = prompt("Введите
ID банковского реквизита, принадлежащего
покупателю");
        var mcRequisiteId = prompt("Введите
ID реквизита, принадлежащего компании
продавцу");
        var mcBankDetailId = prompt("Введите
ID банковского реквизита, принадлежащего
компании продавцу");
        BX24.callMethod(

"crm.requisite.link.register", {
                        fields: {

ENTITY_TYPE_ID: 5,
                                ENTITY_ID:
entityId,

REQUISITE_ID: requisiteId,

BANK_DETAIL_ID: bankDetailId,

MC_REQUISITE_ID: mcRequisiteId,

MC_BANK_DETAIL_ID: mcBankDetailId
                        }
                },
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
```

```
            );
```

crm.requisite.link.unregister

```
crm.requisite.link.unregister(entityTypeId,
entityId)
```

Удаляет связь реквизитов с сущностью.

Параметры

Параметр	Описание
entityTypeId	Идентификатор типа сущности (см. метод crm.enum.ownertype). Для связей реквизитов может быть только сделка, предложение или счёт.
entityId	Идентификатор сущности (сделки, предложения или счёта).

Пример

```
var id = prompt("Введите ID счёта");
        BX24.callMethod(

"crm.requisite.link.unregister", {
```

```
                        entityTypeId: 5,
                        entityId: id
                },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
        );
```

crm.requisite.preset.add

```
crm.requisite.preset.add(fields)
```

Создаёт новый шаблон.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей шаблона. Где "поле" может принимать значения из возвращаемых методом crm.requisite.preset.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.requisite.preset.fields и посмотрите формат пришедших значений этих полей.

Пример

```
BX24.callMethod(
    "crm.requisite.preset.add",
    {
```

```
        fields:
        {
            "ENTITY_TYPE_ID": 8,
            "COUNTRY_ID": 1,
            "NAME": "ИП",
            "XML_ID":
"#CRM_REQUISITE_PRESET_DEF_RU_INDIVIDUAL#",
            "ACTIVE": "Y",
            "SORT": 520
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.info("Создан шаблон с ID
" + result.data());
    }
);
```

crm.requisite.preset.countries

```
crm.requisite.preset.countries()
```

Возвращает возможный список стран для шаблонов реквизита.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(

"crm.requisite.preset.countries",
            {},
            function(result)
            {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
            }
```

```
            );
```

crm.requisite.preset.delete

```
crm.requisite.preset.delete(id)
```

Удаляет шаблон реквизита по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор шаблона реквизита.

Пример

```
    var id = prompt("Введите ID");
        BX24.callMethod(

"crm.requisite.preset.countries",
            { id: id },
            function(result)
            {
                if(result.error())

console.error(result.error());
```

```
                    else

console.dir(result.data());
                }
            );
```

crm.requisite.preset.fields

```
crm.requisite.preset.fields()
```

Возвращает описание полей шаблона реквизитов.

Параметры

Без параметров.

Пример

```
    BX24.callMethod(
            "crm.requisite.preset.fields",
            {},
            function(result)
            {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
            }
```

```
            );
```

crm.requisite.preset.get

```
crm.requisite.preset.get(id)
```

Возвращает шаблон реквизита по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор шаблона реквизита.

Пример

```
    var id = prompt("Введите ID");
        BX24.callMethod(
            "crm.requisite.preset.get",
            { id: id },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else
```

```
console.dir(result.data());
            }
        );
```

crm.requisite.preset.list

```
crm.requisite.preset.list
```

Возвращает список шаблонов по фильтру. Является реализацией списочного метода для шаблонов реквизитов..

Параметры

См. описание списочных методов.

Пример

```
    //Поиск шаблонов по привязке к стране
        BX24.callMethod(
            "crm.requisite.preset.list",
            {
                order: { "ID": "ASC" },
                filter: { "COUNTRY_ID":
"1"},
                select: [ "ID", "NAME"]
            },
            function(result)
            {
                if(result.error())
```

```
console.error(result.error());
                else
                {

console.dir(result.data());
                    if(result.more())
                        result.next();
                }
            }
        );
```

crm.requisite.preset.update

```
crm.requisite.preset.update(fields)
```

Обновление шаблона реквизита.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей шаблона. Где "поле" может принимать значения из возвращаемых методом crm.requisite.preset.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.requisite.preset.fields и посмотрите формат пришедших значений этих полей.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
    "crm.requisite.preset.update",
```

```
    {
        id: id,
                fields:
        {
            "ENTITY_TYPE_ID": 8,
            "COUNTRY_ID": 1,
            "NAME": "ИП (архив)",
            "ACTIVE": "N"
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
        {
            console.info(result.data());
        }
    }
);
```

crm.requisite.preset.field.add

```
crm.requisite.preset.field.add(preset, fields)
```

Добавляет поле в набор полей шаблона, связанного с реквизитом. В наборе могут быть значения из возвращаемых методом crm.requisite.preset.field.fields.

Параметры

Параметр	Описание
preset	Поля шаблона, к которому привязан набор.
fields	Описание полей набора.

Пример

```
BX24.callMethod(
        "crm.requisite.preset.field.add",
        {
                preset:
                {
```

```
                        "ID":1
                },
                fields:
                {

"FIELD_NAME":"RQ_NAME",

"FIELD_TITLE":"TEST",
                        "IN_SHORT_LIST":"N",
                        "SORT":580
                }
        },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.info("В набор полей шаблона
добавлено поле с ID " + result.data());
                }
);
```

crm.requisite.preset.field.avail

```
crm.requisite.preset.field.availabletoadd(preset)
```

Возвращает доступные поля из набора шаблона для определенного реквизита. В наборе могут быть значения из возвращаемых методом crm.requisite.preset.field.fields.

Параметры

Параметр	Описание
preset	Поля шаблона, к которому привязан набор.

Пример

```
BX24.callMethod(
"crm.requisite.preset.field.availabletoadd",
{
preset:
{
```

```
"ID":1
                                    }
                            },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
            }
        );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.requisite.preset.field.delet

```
crm.requisite.preset.field.delete(id,
preset)
```

Удаляет поле из набора полей шаблона для определенного реквизита. В наборе могут быть значения из возвращаемых методом crm.requisite.preset.field.fields.

Параметры

Параметр	Описание
id	Идентификатор поля из набора шаблона.
preset	Поля шаблона, к которому привязан набор.

Пример

```
var id = prompt("Введите ID");
        BX24.callMethod(

"crm.requisite.preset.field.delete",
            {
```

```
                                    ID:id,
                                    preset:
                                    {

"ID":1
                                    }
                            },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
            }
        );
```

crm.requisite.preset.field.fields

```
crm.requisite.preset.field.fields()
```

Возвращает описание набора полей шаблона.

Параметры

Без параметров.

Пример

```
    BX24.callMethod(
            "crm.requisite.preset.fields",
            {},
            function(result)
            {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
            }
        );
```

crm.requisite.preset.field.get

```
crm.requisite.preset.field.get(id, preset)
```

Возвращает описание поля из набора полей шаблона для определенного реквизита. В наборе могут быть значения из возвращаемых методом crm.requisite.preset.field.fields.

Параметры

Параметр	Описание
id	Идентификатор поля.
preset	Поля шаблона, к которому привязан набор полей.

Пример

```
var id = prompt("Введите ID");
        BX24.callMethod(
            "crm.company.get",
            {
                                ID:id,
```

```
                                    preset:
                                    {

"ID":1
                                    }
                            },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
            }
        );
```

crm.requisite.preset.field.list

```
crm.requisite.preset.field.list(preset)
```

Возвращает список полей из набора полей шаблона для определенного реквизита. В наборе могут быть значения из возвращаемых методом crm.requisite.preset.field.fields.

Параметры

Параметр	Описание
preset	Поля шаблона, к которому привязан набор полей.

Пример

```
var id = prompt("Введите ID");

        BX24.callMethod(

"crm.requisite.preset.field.list",
            {
                                preset:
```

```
                                    {

"ID":1
                                    }
                            },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
            }
        );
```

crm.requisite.preset.field.upda

```
crm.requisite.preset.field.update(preset,
fields)
```

Обновляет поле из набора полей шаблона, связанного с реквизитом. В наборе могут быть значения из возвращаемых методом crm.requisite.preset.field.fields.

Параметры

Параметр	Описание
preset	Поля шаблона, к которому привязан набор полей.
fields	Описание полей из набора.

Пример

```
BX24.callMethod(
        "crm.requisite.preset.field.update",
        {
                ID:id,
```

```
                preset:
                {
                        "ID":1
                },
                fields:
                {
                        "FIELD_TITLE":"Имя",
                        "IN_SHORT_LIST":"Y",
                }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                {

console.info(result.data());
                }
        }
);
```

crm.requisite.userfield.add

```
crm.requisite.userfield.add(fields)
```

Создаёт новое пользовательское поле для реквизита.

Системное ограничение на название поля - 20 знаков. К названию пользовательского поля всегда добавляется префикс UF_CRM_, то есть реальная длина названия - 13 знаков.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий описание пользовательского поля. Полное описание полей можно получить вызовом метода crm.userfield.fields.
LIST	Содержит набор значений списка для пользовательских полей типа Список. Указывается при создании/обновлении поля. Каждое значение представляет собой массив с полями: ▪ **VALUE** - значение элемента списка. Поле является обязательным в случае, когда создается новый элемент. ▪ **SORT** - сортировка.

- **DEF** - если равно Y, то элемент списка является значением по-умолчанию. Для множественного поля допустимо несколько DEF=Y. Для не множественного, дефолтным будет считаться первое.
- **XML_ID** - внешний код значения. Параметр учитывается только при обновлении уже существующих значений элемента списка.
- **ID** - идентификатор значения. Если он указан, то считается что это обновление существующего значения элемента списка, а не создание нового. Имеет смысл только при вызове методов `*.userfield.update`.
- **DEL** - если равно Y, то существующий элемент списка будет удален. Применяется, если заполнен параметр ID.

Пример

```
BX24.callMethod(
    "crm.requisite.userfield.add",
    {
        fields:
        {
            "ENTITY_ID": "CRM_REQUISITE",
            "FIELD_NAME": "MY_STRING",
            "EDIT_FORM_LABEL": "Моя строка",
            "USER_TYPE_ID": "string",
            "XML_ID": "MY_STRING",
            "SETTINGS": { "DEFAULT_VALUE":
"Привет, мир!" }
        }
    },
    function(result)
```

```
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
```

crm.requisite.userfield.delete

```
crm.requisite.userfield.delete(id)
```

Удаляет пользовательское поле реквизита.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
    "crm.requisite.userfield.delete",
    {
        id: id
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
```

```
        else
            console.info(result.data());
    }
);
```

crm.requisite.userfield.get

```
crm.requisite.userfield.get(id)
```

Возвращает пользовательское поле реквизита по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
    "crm.requisite.userfield.get",
    {
        id: id
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
```

```
        else
            console.dir(result.data());
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.requisite.userfield.list

```
crm.requisite.userfield.list(order, filter)
```

Возвращает список пользовательских полей реквизита по фильтру. Полное описание полей можно получить вызовом метода crm.userfield.fields

Параметры

Параметр	Описание
order	Поля сортировки.
filter	Поля фильтра.

Пример

```
BX24.callMethod(
    "crm.requisite.userfield.list",
    {
        order: { "SORT": "ASC" },
        filter: { "MANDATORY": "N" }
    },
```

```
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
        {
            console.dir(result.data());
            if(result.more())
                result.next();
        }
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.requisite.userfield.update

```
crm.requisite.userfield.update(id, fields)
```

Обновляет существующее пользовательское поле реквизита..

Параметры

Параметр	Описание
id	Идентификатор пользовательского поля.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.userfield.fields.
LIST	Содержит набор значений списка для пользовательских полей типа Список. Указывается при создании/обновлении поля. Каждое значение представляет собой массив с полями: ▪ **VALUE** - значение элемента списка. Поле является обязательным в случае, когда создается новый элемент. ▪ **SORT** - сортировка. ▪ **DEF** - если равно Y, то элемент списка является значением по-умолчанию. Для

	множественного поля допустимо несколько DEF=Y. Для не множественного, дефолтным будет считаться первое. ▪ **XML_ID** - внешний код значения. Параметр учитывается только при обновлении уже существующих значений элемента списка. ▪ **ID** - идентификатор значения. Если он указан, то считается что это обновление существующего значения элемента списка, а не создание нового. Имеет смысл только при вызове методов `*.userfield.update`. ▪ **DEL** - если равно Y, то существующий элемент списка будет удален. Применяется, если заполнен параметр ID.

Пример

```
var id = prompt("Введите ID");
var label = prompt("Введите новое
название");
BX24.callMethod(
    "crm.requisite.userfield.update",
    {
        id: id,
        fields:
        {
            "EDIT_FORM_LABEL": label,
            "LIST_COLUMN_LABEL": label
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
```

```
        else
        {
            console.dir(result.data());
            if(result.more())
                result.next();
        }
    }
);
```

onCrmAddressRegister

Событие вызывается при регистрации адреса.

Параметры

Параметр	Описание	С версии
FIELDS	Массив содержит следующие поля: ▪ TYPE_ID - значение типа адреса ▪ ENTITY_TYPE_ID - значение типа сущности ▪ ENTITY_TYPE_ID - значение типа сущности ▪ ENTITY_ID - значение идентификатор сущности ▪ ANCHOR_ID - значение идентификатор старшей сущности (например самого Реквизита) ▪ ANCHOR_TYPE_ID - значение типа старшей сущности (например самого Реквизита)	

© «Битрикс», 2001-2008, «1С-

onCrmAddressUnregister

Событие вызывается при удалении адреса.

Параметры

Параметр	Описание	С версии
FIELDS	Массив содержит следующие поля: ▪ TYPE_ID - значение типа адреса ▪ ENTITY_TYPE_ID - значение типа сущности ▪ ENTITY_TYPE_ID - значение типа сущности ▪ ENTITY_ID - значение идентификатор сущности ▪ ANCHOR_ID - значение идентификатор старшей сущности (например самого Реквизита) ▪ ANCHOR_TYPE_ID - значение типа старшей сущности (например самого Реквизита)	

© «Битрикс», 2001-2008, «1С-

onCrmBankDetailAdd

Событие вызывается при добавления банковского реквизита.

Параметры

Параметр	Описание	С версии
FIELDS	Массив содержит следующие поля: ID - значение идентификатор банковского реквизита	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmBankDetailDelete

Событие вызывается при удалении банковского реквизита.

Параметры

Параметр	Описание	С версии
FIELDS	Массив содержит следующие поля: ID - значение идентификатор реквизита	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmBankDetailUpdate

Событие вызывается при обновлении банковского реквизита.

Параметры

Параметр	Описание	С версии
FIELDS	Массив содержит следующие поля: ID - значение идентификатора обновляемого банковского реквизита	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmRequisiteAdd

Событие вызывается при добавлении реквизита.

Параметры

Параметр	Описание	С версии
FIELDS	Массив содержит следующие поля: ID - значение идентификатор реквизита	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmRequisiteDelete

Событие вызывается при удалении реквизита.

Параметры

Параметр	Описание	С версии
FIELDS	Массив содержит следующие поля: ID - значение идентификатора удаляемого реквизита	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmRequisiteUpdate

Событие вызывается при обновлении реквизита.

Параметры

Параметр	Описание	С версии
FIELDS	Массив содержит следующие поля: ID - значение идентификатора изменяемого реквизита.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmRequisiteUserFieldAdd

Событие, вызываемое при добавлении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmRequisiteUserFieldDelete

Событие, вызываемое при удалении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmRequisiteUserFieldSetEn

Событие, вызываемое при изменении набора значений для пользовательского поля списочного типа.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmRequisiteUserFieldUpdat

Событие, вызываемое при изменении пользовательского поля.

Параметры

Параметр	Описание
id	идентификатор пользовательского поля.
entityId	символьный идентификатор сущности, для которой создано поле
fieldName	имя созданного пользовательского поля

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.documentgenerator.docum

```
crm.documentgenerator.document.add(templateId, entityTypeId, entityId, values = [])
```

Метод создает новый документ на основании шаблона и данных из соответствующей сущности. В `values` можно передать массив дополнительных значений полей. В случае успешного выполнения в результате придёт структура, аналогичная методу crm.documentgenerator.document.get() на новом документе.

Почему в результате crm.documentgenerator.document.add нет ссылки на pdf?

Параметры

Параметр	Описание
templateId	ID шаблона.
entityTypeId	ID типа сущности CRM.
entityId	ID сущности.
values	дополнительные значения полей.

stampsEnabled	1 (поставить), 0 (убрать) печати и подписи.

Смотри также

- Примеры генерации документа
- Презентация по генератору документов
- Шаблоны документов

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.documentgenerator.docum

```
crm.documentgenerator.document.delete(id)
```

Удаляет документ. Ответ пустой.

Параметры

Параметр	Описание
id	идентификатор документа.

Примеры

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.documentgenerator.docum

```
crm.documentgenerator.document.enablepublicurl(id, status = 1)
```

Включает / выключает публичную ссылку на документ.

Параметры

Параметр	Описание
id	идентификатор документа.
status	1 (включить), 0 (выключить) публичную ссылку на документ.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.documentgenerator.docum

```
crm.documentgenerator.document.get(id)
```

Возвращает информацию о документе по его идентификатору.

Почему в результате crm.documentgenerator.document.add нет ссылки на pdf?

Параметры

Параметр	Описание
id	ID документа

Пример

```
"document": {
    "id": 1, // id документа
    "title": "Счет (Россия) 1", // название
```

```
    "number": "1", // номер
    "createTime": "2018-06-
05T16:04:40+02:00", // дата создания
    "updateTime": "2018-06-
05T16:04:40+02:00", // дата обновления
    "createdBy": "1", // ид пользователя,
кто создал документ
    "updatedBy": "1", // ид пользователя,
кто обновил документ
    "stampsEnabled": true // вставлены ли
печати и подписи
    "downloadUrl": "", // ссылка на
скачивание docx файла пользователем
    "downloadUrlMachine": "", // ссылка на
скачивание docx файла приложением
    "imageUrl": "", // ссылка на скачивание
картинки пользователем
    "imageUrlMachine": "", // ссылка на
скачивание картинки приложением
    "pdfUrl": "", // ссылка на скачивание
pdf пользователем
    "pdfUrlMachine": "", // ссылка на
скачивание pdf приложением
    "publicUrl": "" // публичная ссылка
(если есть)
    "isTransformationError": false, // была
ли ошибка конвертации
    "templateId": "1", // ид шаблона
    "entityTypeId": 1, // ид типа сущности
CRM
    "entityId": "1", // ИД сущности CRM
    "values": {},// массив дополнительных
значений
}
```

crm.documentgenerator.docum

```
crm.documentgenerator.document.getfields(id, values = [])
```

Возвращает список полей документа с их описанием. Возвращаемые значения абсолютно идентичны методу crm.documentgenerator.template.getfields().

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.documentgenerator.docum

```
crm.documentgenerator.document.list(select = ['*'], order = [], filter = [], start = 0)
```

Возвращает список документов по фильтру.

Параметры

Параметр	Описание
select	массив полей для вывода.
order	массив для указания порядка вывода `{"id": "desc"}`.
filter	массив для фильтрации.
start	offset для постраничной навигации.

Пример фильтра

```
"filter": {
  "entityTypeId": 2,
  "entityId": 5
}
```

В результате будет список документов, сформированный для сделки с ID=5.

Ответ


```
"documents": [
    "0": {
        "id": "1523",
        "title": "Акт (Россия)  1",
        "number": "1",
        "templateId": "115",
        "entityTypeId": "2",
        "fileId": "4315",
        "imageId": "4316",
        "pdfId": "4317",
        "createTime": "2018-06-
05T16:04:40+02:00",
        "updateTime": "2018-06-
05T16:04:40+02:00",
        "values": {},
        "entityId": "5",
        "downloadUrl": "",
        "downloadUrlMachine": "",
        "imageUrl": "",
        "imageUrlMachine": "",
        "pdfUrl": "",
        "pdfUrlMachine": "",
```

```
        }
]
```

crm.documentgenerator.docum

```
crm.documentgenerator.document.update(id,
values, stampsEnabled = 1)
```

Обновляет существующий документ с новыми значениями. Надо учесть, что обновление документа на удалённом шаблоне невозможно. Работает аналогично crm.documentgenerator.document.add().

Параметры

Параметр	Описание
id	идентификатор документа.
values	массив новых значений полей документа.
stampsEnabled	1 (поставить), 0 (убрать) печати и подписи.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.documentgenerator.docum

```
crm.documentgenerator.document.upload(fileCo
ntent, region, entityTypeId, entityId,
title, number)
```

Загружает сформированный документ и прикрепляет его к указанной сущности. В процессе загрузки создаётся скрытый шаблон, привязанный к текущему rest-приложению (один шаблон на каждое приложение). К этому шаблону прикрепляется пустой файл (т.к. шаблон не может быть без файла).

Метод возвращает значения, аналогичные crm.documentgenerator.document.add.

Параметры

Параметр	Описание
fileContent	контент docx файла в *base64*.
region	страна.
entityTypeId	ID типа сущности CRM.
entityId	ID сущности CRM.
title	заголовок документа.

number	номер документа.
pdfContent	контент pdf файла в *base64*, не обязательно.
imageContent	контент картинки в *base64*, не обязательно.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.documentgenerator.templa

```
crm.documentgenerator.template.add(fields)
```

Метод добавляет новый шаблон. Возвращает те же данные, что и при вызове crm.documentgenerator.template.get() на новом шаблоне.

Параметры

Параметр	Описание
fields	массив полей шаблона, среди которых: ▪ `fields[name]` - название шаблона (обязательное). ▪ `fields[file]` - контент файла, закодированный в *base64* (обязательное). Как альтернативу, контент файла можно передать в *multipart / form-data*. В этом случае его не надо кодировать в *base64*. ▪ `fields[numeratorId]` - идентификатор нумератора (обязательное). ▪ `fields[region]` - страна (обязательное). ▪ `fields[entityTypeId]` - массив идентификаторов привязанных сущностей (обязательное). Здесь необходимо передавать код сделки с учетом фильтрации по направлениям.

- `fields[users]` - массив прав доступа.
- `fields[active]` - Y/N флаг активности.
- `fields[withStamps]` - Y/N ставить печати и подписи.
- `fields[sort]` - индекс сортировки.

crm.documentgenerator.templa

```
crm.documentgenerator.template.delete(id)
```

Удаляет шаблон. Ответ пустой.

Параметры

Параметр	Описание
id	идентификатор шаблона

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.documentgenerator.templa

```
crm.documentgenerator.template.get(id)
```

Возвращает информацию о шаблоне по его идентификатору.

Параметры

Параметр	Описание
id	ID шаблона

Пример

```
"template": {
    "id": 1, // id шаблона
    "name": "Счет (Россия)", // название
    "region": "ru", // страна
    "code": "INVOICE_RU", // код
    "download": '', // ссылка на скачивание
для пользователя
    "downloadMachine": '', // ссылка на
скачивание для приложения
    "active": "Y", // активность
```

```
    "moduleId": "crm", // ид модуля
    "numeratorId": 1, // ид нумератора
    "withStamps": "Y", // ставить печати по
умолчанию
    "isDeleted": "N" // удален или нет
    "entityTypeId": [ // привязанные
сущности
        "0": "4",
        "1": "3",
        "2": "2_category_0",
        "3": "2_category_1",
        "4": "5",
        "5": "1",
        "6": "14",
        "7": "7"
    ],
    "users" [ // привязанные пользователи
      "0": "UA"
    ],
    "sort": 500, // индекс сортировки
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.documentgenerator.templa

```
crm.documentgenerator.template.getfields(id, entityTypeId, entityId, values = [])
```

Возвращает список полей шаблона с их описанием.

Параметры

Параметр	Описание
id	ID шаблона
entityTypeId	ID типа CRM-сущности
entityId	ID используемой сущности
values	массив дополнительных значений

Пример

```
"templateFields": {
    "DocumentNumber": {
```

```
        "title": "Номер" // заголовок
        "value": "22" // значение
        "group": [ // иерархия
            0: "Документ"
        ],
        "default": "22" // значение по
умолчанию
    },
    "MyCompanyUfLogo": { // поле типа
картинка
        "title": "Логотип",
        "value": "",
        "type": "IMAGE",
        "group": [
            0: "Документ",
            1: "Моя компания"
        ],
        "default": ""
    },
    "MY_COMPANY": { // поле, у которого есть
несколько вариантов выбора
        "title": "Моя компания"
        "value": [
            0: {
                "value": "6"
                "title": "1С-Битрикс"
                "selected": "1"
            },
            1: {
                "value": "11"
                "title": "ИП Копытов"
                "selected":" "
            }
        "group": [
            0: "Документ",
            1: "Моя компания"
        ],
```

```
        }
    }
```

crm.documentgenerator.templa

Описание

```
crm.documentgenerator.template.list(select =
['*'], filter = [], order = [], start = 0)
```

Возвращает список шаблонов по фильтру.

Параметры

Параметр	Описание
select	массив полей для вывода. По умолчанию выводит все поля шаблона, кроме *users* и *entityTypeId*. Чтобы они появились, надо добавить дополнительно. Например, `['*', 'entityTypeId', 'users']`.
order	массив для указания порядка вывода `{"id": "desc"}`.
filter	массив для фильтрации.
start	offset для постраничной навигации.

Примеры фильтра

```
filter: {
  "code": "%_RU",
  "numeratorId": "2",
  "region": "by",
  "active": "Y"
}
```

В фильтр можно передать ID сущности по ключу *entityTypeId*. Сюда надо передавать код сделки с учетом фильтрации по направлениям. Однако, если нужно вернуть шаблоны, привязанные к любому направлению сделки, можно передать в фильтр.

```
filter: {
  "entityTypeId": "2%"
}
```

Метод вернет список шаблонов с их полями.

Пример

```
templates: {
   115: {
        "id": "115"
        "active": "Y"
        "name": "Акт (Россия)"
        "code": "ACT_RU"
        "region": "ru"
        "sort": "100"
        "createTime": "2018-06-
05T13:07:12+02:00"
```

```
        "updateTime": "2018-09-
06T14:26:24+02:00"
        "moduleId": "crm"
        "numeratorId": "29"
        "withStamps": "N"
        "isDeleted": "N"
        "entityTypeId": [
            "0": "4",
            "1": "3",
            "2": "2_category_0",
            "3": "2_category_1",
            "4": "5",
            "5": "1",
            "6": "14",
            "7": "7"
        ],
        "users" [
          "0": "UA"
        ]
    }
}
```

crm.documentgenerator.templa

```
crm.documentgenerator.template.update(id,
fields)
```

Метод обновляет существующий шаблон. Возвращает те же данные, что и при вызове crm.documentgenerator.template.get().

Параметры

Параметр	Описание
id	идентификатор шаблона.
fields	массив полей. Аналогично методу crm.documentgenerator.template.add, только здесь все поля необязательные.

© «Битрикс», 2001-2008, «1С-

crm.documentgenerator.numer

```
crm.documentgenerator.numerator.add(fields)
```

Метод добавляет новый нумератор. Возвращает результат, идентичный crm.documentgenerator.numerator.get().

Параметры

Параметр	Описание
name	Имя
template	Шаблон
settings	Настройки генераторов

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.documentgenerator.numer

```
crm.documentgenerator.numerator.delete(id)
```

Метод удаляет нумератор.

Удалить можно только те нумераторы, которые были созданы через crm.documentgenerator.numerator.add().

Ответ пустой.

Параметры

Параметр	Описание
id	ID нумератора

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.documentgenerator.numer

```
crm.documentgenerator.numerator.get(id)
```

Метод возвращает информацию о нумераторе по его идентификатору.

Параметры

Параметр	Описание
id	ID нумератора.

Ответ

```
"numerator": {
    "id": "202", // id шаблона
    "name": "Rest Template", // название
    "template": "{NUMBER}", // шаблон
    "settings": { // настройки генераторов

"Bitrix_Main_Numerator_Generator_SequentNumb
```

```
erGenerator": {
            "start":  20,
            "step":  5,
            "periodicBy": '',
            "timezone": '',
            "isDirectNumeration": ''
        }
    },
}
```

crm.documentgenerator.numer

```
crm.documentgenerator.numerator.list(start =
0)
```

Метод возвращает список нумераторов.

Параметры

Параметр	Описание
start	offset при запросе, для постраничной навигации.

Ответ

```
"numerators": [
    0: {
        "id": "202", // id шаблона
        "name": "Rest Template", // название
        "template": "{NUMBER}", // шаблон
        "settings": { // настройки
```

```
генераторов

"Bitrix_Main_Numerator_Generator_SequentNumb
erGenerator": {
                "start":  20,
                "step":  5,
                "periodicBy": '',
                "timezone": '',
                "isDirectNumeration": ''
            }
        }
    }
]
```

crm.documentgenerator.numer

```
crm.documentgenerator.numerator.update(id, fields)
```

Метод обновляет существующий нумератор с новыми значениями.

Обновить можно только те нумераторы, которые были созданы через crm.documentgenerator.numerator.add().

Возвращает результат, идентичный crm.documentgenerator.numerator.get().

Параметры

Параметр	Описание
id	ID нумератора
fields	Массив, аналогичный crm.documentgenerator.numerator.add(), только все поля необязательны.

© «Битрикс», 2001-2008, «1С-

Настройка карточек сущностей::crm.lead.details.cc

Метод для получения параметров настройки карточки лидов. Метод читает личные настройки карточки указанного пользователя или общие настройки, заданные для всех пользователей.

Обратите внимание, что настройки карточки повторных лидов могут отличаться от настроек карточки простых лидов. Для переключения между настройками карточек лидов применяется параметр **leadCustomerType**.

Параметры

Параметр	Описание	С версии
scope	Область применения настроек. Допустимые значения: ▪ **P** - личные настройки, ▪ **C** - общие настройки	
userId	Идентификатор пользователя. Если не задан, то берётся текущий. Нужен только при запросе личных настроек.	
extras	Дополнительные параметры. Здесь для лидов может быть	

	задан параметр `leadCustomerType`, с допустимыми значениями: ▪ **1** - простые лиды, ▪ **2** - повторные лиды	

Пример

```
//--
//Запрос личных настроек карточки лидов для
пользователя с идентификатором 1.
BX24.callMethod(
    "crm.lead.details.configuration.get",
    {
        scope: "P",
        userId: 1
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//Запрос общих настроек карточки лидов.
BX24.callMethod(
    "crm.lead.details.configuration.get",
    {
        scope: "C"
    },
    function(result)
    {
        if(result.error())
```

```
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//Запрос общих настроек карточки повторных
лидов.
BX24.callMethod(
    "crm.lead.details.configuration.get",
    {
        scope: "C",
        extras: { leadCustomerType: 2 }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//--
```

Настройка карточек сущностей::crm.lead.details.cc

Метод установливает настройки карточки лидов. Метод записывает личные настройки карточки указанного пользователя или общие настройки для всех пользователей.

Обратите внимание, что настройки карточки повторных лидов могут отличаться от настроек карточки простых лидов. Для переключения между настройками карточек лидов применяется параметр **leadCustomerType**.

Параметры

Параметр	Описание	С версии
scope	Область применения настроек. Допустимые значения: ▪ **P** - личные настройки, ▪ **C** - общие настройки	
userId	Идентификатор пользователя. Если не задан, то берётся текущий. Нужен только при установке личных настроек.	
extras	Дополнительные параметры. Здесь для лидов может быть	

	задан параметр `leadCustomerType`, с допустимыми значениями: ▪ **1** - простые лиды, ▪ **2** - повторные лиды	

Пример

```
//---
//Установка личных настроек карточки лидов
для пользователя с идентификатором 1.
BX24.callMethod(
    "crm.lead.details.configuration.set",
    {
        scope: "P",
        userId: 1,
        data:
            [
                {
                    name: "main",
                    title: "Общие сведения",
                    type: "section",
                    elements:
                        [
                            { name: "TITLE"
},
                            { name:
"STATUS_ID" },
                            { name: "NAME"
},
                            { name:
"BIRTHDATE" },
                            { name: "POST"
},
```

```
                                { name: "PHONE"
},
                                { name: "EMAIL"
}
                            ]
                    },
                    {
                        name: "additional",
                        title: "Дополнительно",
                        type: "section",
                        elements:
                            [
                                { name:
"SOURCE_ID" },
                                { name:
"SOURCE_DESCRIPTION" },
                                { name: "OPENED"
},
                                { name:
"ASSIGNED_BY_ID" },
                                { name:
"OBSERVER" },
                                { name:
"COMMENTS" }
                            ]
                    }
                ]
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//---
```

Настройка карточек сущностей::crm.lead.details.cc

Метод сбрасывает настройки карточки лидов. Метод удаляет личные настройки карточки указанного пользователя или общие настройки, заданные для всех пользователей.

Обратите внимание, что настройки карточки повторных лидов могут отличаться от настроек карточки простых лидов. Для переключения между настройками карточек лидов применяется параметр **leadCustomerType**.

Параметры

Параметр	Описание	С версии
scope	Область применения настроек. Допустимые значения: ▪ **P** - личные настройки, ▪ **C** - общие настройки	
userId	Идентификатор пользователя. Если не задан, то берётся текущий. Нужен только при сбросе личных настроек.	
extras	Дополнительные параметры. Здесь для лидов может быть	

	задан параметр `leadCustomerType`, с допустимыми значениями: ▪ **1** - простые лиды, ▪ **2** - повторные лиды	

Пример

```
//---
//Сброс личных настроек карточки лидов для
пользователя с идентификатором 1.
BX24.callMethod(
    "crm.lead.details.configuration.reset",
    {
        scope: "P",
        userId: 1
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//---
```

Настройка карточек сущностей::crm.lead.details.cc

Метод принудительно устанавливает общую карточку лидов для всех пользователей.

Обратите внимание, что настройки карточки повторных лидов могут отличаться от настроек карточки простых лидов. Для переключения между настройками карточек лидов применяется параметр **leadCustomerType**.

Параметры

Параметр	Описание	С версии
extras	Дополнительные параметры. Здесь для лидов может быть задан параметр `leadCustomerType`, с допустимыми значениями: ▪ **1** - простые лиды, ▪ **2** - повторные лиды	

Пример

```
//---
//Установить общую карточку лидов для всех
пользователей.
BX24.callMethod(

"crm.lead.details.configuration.forceCommonS
copeForAll",
    {},
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//---
```

Настройка карточек сущностей::crm.deal.details.cc

Метод для получения настроек карточки сделок. Метод читает личные настройки карточки указанного пользователя или общие настройки, заданные для всех пользователей.

Обратите внимание, что настройки карточки сделок разных направлений (или воронок) могут отличаться друг от друга. Для переключения между настройками карточек сделок разных направлений применяется параметр **dealCategoryId**.

Параметры

Параметр	Описание	С версии
scope	Область применения настроек. Допустимые значения: ▪ **P** - личные настройки, ▪ **C** - общие настройки	
userId	Идентификатор пользователя. Если не задан, то берётся текущий. Нужен только при запросе личных настроек.	
extras	Дополнительные параметры. Здесь для сделок может быть	

	задан параметр `dealCategoryId`.	

Пример

```
//--
//Запрос личных настроек карточки сделок для
пользователя с идентификатором 1.
BX24.callMethod(
    "crm.deal.details.configuration.get",
    {
        scope: "P",
        userId: 1
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//Запрос общих настроек карточки сделок для
общего направления.
BX24.callMethod(
    "crm.deal.details.configuration.get",
    {
        scope: "C"
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
```

```
    }
);
//Запрос общих настроек карточки сделок для
направления с идентификатором 1.
BX24.callMethod(
    "crm.deal.details.configuration.get",
    {
        scope: "C",
        extras: { dealCategoryId: 1 }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//--
```

Настройка карточек сущностей::crm.deal.details.cc

Метод позволяет установить настройки карточки сделок. Метод записывает личные настройки карточки указанного пользователя или общие настройки для всех пользователей.

Обратите внимание, что настройки карточки сделок разных направлений (или воронок) могут отличаться друг от друга. Для переключения между настройками карточек сделок разных направлений применяется параметр **dealCategoryId**.

Параметры

Параметр	Описание	С версии
scope	Область применения настроек. Допустимые значения: ▪ **P** - личные настройки, ▪ **C** - общие настройки	
userId	Идентификатор пользователя. Если не задан, то берётся текущий. Нужен только при установке личных настроек.	
extras	Дополнительные параметры. Здесь для сделок может быть	

	задан параметр `dealCategoryId`.	

Пример

```
//---
//Установка личных настроек карточки сделок
общего направления для пользователя с
идентификатором 1.
BX24.callMethod(
    "crm.deal.details.configuration.set",
    {
        scope: "P",
        userId: 1,
        data:
            [
                {
                    name: "main",
                    title: "О сделке",
                    type: "section",
                    elements:
                        [
                            { name: "TITLE"
},
                            { name:
"OPPORTUNITY_WITH_CURRENCY" },
                            { name:
"STAGE_ID" },
                            { name:
"BEGINDATE" },
                            { name:
"CLOSEDATE" },
                            { name: "CLIENT"
}
                        ]
```

```
                },
                {
                    name: "additional",
                    title: "Дополнительно",
                    type: "section",
                    elements:
                        [
                            { name:
"TYPE_ID" },
                            { name:
"SOURCE_ID" },
                            { name:
"SOURCE_DESCRIPTION" },
                            { name: "OPENED"
},
                            { name:
"ASSIGNED_BY_ID" },
                            { name:
"OBSERVER" },
                            { name:
"COMMENTS" }
                        ]
                },
                {
                    name: "products",
                    title: "Товары",
                    type: "section",
                    elements:
                        [
                            { name:
"PRODUCT_ROW_SUMMARY" }
                        ]
                }
            ]
    },
    function(result)
    {
```

```
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//---
```

Настройка карточек сущностей::crm.deal.details.cc

Метод для сброса настроек карточки сделок. Метод удаляет личные настройки карточки указанного пользователя или общие настройки, заданные для всех пользователей.

Обратите внимание, что настройки карточки сделок разных направлений (или воронок) могут отличаться друг от друга. Для переключения между настройками карточек сделок разных направлений применяется параметр **dealCategoryId**.

Параметры

Параметр	Описание	С версии
scope	Область применения настроек. Допустимые значения: ▪ **P** - личные настройки, ▪ **C** - общие настройки	
userId	Идентификатор пользователя. Если не задан, то берётся текущий. Нужен только при сбросе личных настроек.	
extras	Дополнительные параметры. Здесь для сделок может быть	

	задан параметр `dealCategoryId`.	

Пример

```
//---
//Сброс личных настроек карточки сделок
общего направления для пользователя с
идентификатором 1.
BX24.callMethod(
    "crm.deal.details.configuration.reset",
    {
        scope: "P",
        userId: 1
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//---
```

Настройка карточек сущностей::crm.deal.details.cc

Метод принудительно устанавливает общую карточку сделок для всех пользователей.

Обратите внимание, что настройки карточки сделок разных направлений (или воронок) могут отличаться друг от друга. Для переключения между настройками карточек сделок разных направлений применяется параметр **dealCategoryId**.

Параметры

Параметр	Описание	С версии
extras	Дополнительные параметры. Здесь для сделок может быть задан параметр `dealCategoryId`.	

Пример

```
//---
//Установить общую карточку сделок общего направления для всех пользователей.
```

```
BX24.callMethod(

"crm.deal.details.configuration.forceCommonS
copeForAll",
    {},
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//---
```

Настройка карточек сущностей::crm.contact.details

Метод для получения настроек карточки контактов. Метод читает личные настройки карточки указанного пользователя или общие настройки, заданные для всех пользователей.

Параметры

Параметр	Описание	С версии
scope	Область применения настроек. Допустимые значения: ▪ **P** - личные настройки, ▪ **C** - общие настройки	
userId	Идентификатор пользователя. Если не задан, то берётся текущий. Нужен только при запросе личных настроек.	

Пример

```
//--
//Запрос личных настроек карточки контактов
```

```
для пользователя с идентификатором 1.
BX24.callMethod(
    "crm.contact.details.configuration.get",
    {
        scope: "P",
        userId: 1
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//Запрос общих настроек карточки контактов.
BX24.callMethod(
    "crm.contact.details.configuration.get",
    {
        scope: "C"
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//---
```

Настройка карточек сущностей::crm.contact.details

Метод устанавливает настройки карточки контактов. Метод записывает личные настройки карточки указанного пользователя или общие настройки для всех пользователей.

Параметры

Параметр	Описание	С версии
scope	Область применения настроек. Допустимые значения: ▪ **P** - личные настройки, ▪ **C** - общие настройки	
userId	Идентификатор пользователя. Если не задан, то берётся текущий. Нужен только при установке личных настроек.	
data	Массив настроек.	

Пример

```
//Установка личных настроек карточки
контактов для пользователя с идентификатором
1.
BX24.callMethod(
    "crm.contact.details.configuration.set",
    {
        scope: "P",
        userId: 1,
        data:
            [
                {
                    name: "main",
                    title: "О контакте",
                    type: "section",
                    elements:
                        [
                            { name: "NAME"
},
                            { name:
"SECOND_NAME" },
                            { name:
"LAST_NAME" },
                            { name: "PHOTO"
},
                            { name:
"BIRTHDATE" },
                            { name: "POST"
},
                            { name: "PHONE"
},
                            { name: "EMAIL"
},
                            { name:
"COMPANY" }
                        ]
                },
                {
```

```
                        name: "additional",
                        title: "Дополнительно",
                        type: "section",
                        elements:
                            [
                                { name:
"TYPE_ID" },
                                { name:
"SOURCE_ID" },
                                { name:
"SOURCE_DESCRIPTION" },
                                { name: "OPENED"
},
                                { name: "EXPORT"
},
                                { name:
"ASSIGNED_BY_ID" },
                                { name:
"COMMENTS" }
                            ]
                    }
                ]
        },
        function(result)
        {
            if(result.error())
                console.error(result.error());
            else
                console.dir(result.data());
        }
);
//---
```

Настройка карточек сущностей::crm.contact.details

Метод для сброса настроек карточки контактов. Метод удаляет личные настройки карточки указанного пользователя или общие настройки, заданные для всех пользователей.

Параметры

Параметр	Описание	С версии
scope	Область применения настроек. Допустимые значения: ▪ **P** - личные настройки, ▪ **C** - общие настройки	
userId	Идентификатор пользователя. Если не задан, то берётся текущий. Нужен только при сбросе личных настроек.	

Пример

```
//---
//Сброс личных настроек карточки контактов
```

```
для пользователя с идентификатором 1.
BX24.callMethod(

"crm.contact.details.configuration.reset",
    {
        scope: "P",
        userId: 1
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//---
```

Настройка карточек сущностей::crm.contact.details

Метод позволяет принудительно установить общую карточку контактов для всех пользователей.

Параметры

Без параметров

Пример

```
//---
//Установить общую карточку контактов для
всех пользователей.
BX24.callMethod(

"crm.contact.details.configuration.forceComm
onScopeForAll",
    {},
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
```

```
        }
);
//---
```

Настройка карточек сущностей::crm.company.deta

Метод для получения настроек карточки компаний. Метод читает личные настройки карточки указанного пользователя или общие настройки, заданные для всех пользователей.

Параметры

Параметр	Описание	С версии
scope	Область применения настроек. Допустимые значения: ▪ **P** - личные настройки, ▪ **C** - общие настройки	
userId	Идентификатор пользователя. Если не задан, то берётся текущий. Нужен только при запросе личных настроек.	

Пример

```
//--
//Запрос личных настроек карточки компаний
```

```
для пользователя с идентификатором 1.
BX24.callMethod(
    "crm.company.details.configuration.get",
    {
        scope: "P",
        userId: 1
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//Запрос общих настроек карточки компаний.
BX24.callMethod(
    "crm.company.details.configuration.get",
    {
        scope: "C"
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//---
```

Настройка карточек сущностей::crm.company.deta

Метод устанавливает настройки карточки компаний. Метод записывает личные настройки карточки указанного пользователя или общие настройки для всех пользователей.

Параметры

Параметр	Описание	С версии
scope	Область применения настроек. Допустимые значения: ▪ **P** - личные настройки, ▪ **C** - общие настройки	
userId	Идентификатор пользователя. Если не задан, то берётся текущий. Нужен только при установке личных настроек.	
data	Массив параметров.	

Пример

```
//Установка личных настроек карточки
компаний для пользователя с идентификатором
1.
BX24.callMethod(
    "crm.company.details.configuration.set",
    {
        scope: "P",
        userId: 1,
        data:
            [
                {
                    name: "main",
                    title: "О компании",
                    type: "section",
                    elements:
                        [
                            { name: "TITLE"
},
                            { name: "LOGO"
},
                            { name:
"COMPANY_TYPE" },
                            { name: "POST"
},
                            { name: "PHONE"
},
                            { name: "EMAIL"
},
                            { name:
"CONTACT" }
                        ]
                },
                {
                    name: "additional",
                    title: "Дополнительно",
                    type: "section",
                    elements:
```

```
                            [
                                { name:
"INDUSTRY" },
                                { name: "OPENED"
},
                                { name:
"ASSIGNED_BY_ID" },
                                { name:
"COMMENTS" }
                            ]
                        }
                    ]
        },
        function(result)
        {
            if(result.error())
                console.error(result.error());
            else
                console.dir(result.data());
        }
);
//---
```

Настройка карточек сущностей::crm.company.deta

Метод для сброса настроек карточки компаний. Метод удаляет личные настройки карточки указанного пользователя или общие настройки, заданные для всех пользователей.

Параметры

Параметр	Описание	С версии
scope	Область применения настроек. Допустимые значения: ▪ **P** - личные настройки, ▪ **C** - общие настройки	
userId	Идентификатор пользователя. Если не задан, то берётся текущий. Нужен только при сбросе личных настроек.	

Пример

```
//---
//Сброс личных настроек карточки компаний
для пользователя с идентификатором 1.
BX24.callMethod(

"crm.company.details.configuration.reset",
    {
        scope: "P",
        userId: 1
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
//---
```

Настройка карточек сущностей::crm.company.deta

Метод позволяет принудительно установить общую карточку компаний для всех пользователей.

Параметры

Без параметров

Примеры

```
//---
//Установить общую карточку компаний для
всех пользователей.
BX24.callMethod(

"crm.company.details.configuration.forceComm
onScopeForAll",
    {},
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
```

```
        }
);
//---
```

Общие принципы работы с пользовательскими полями

Есть несколько правил работы с **пользовательскими полями** для всех сущностей CRM:

1. Для корректного запуска методов, работающих с пользовательскими полями, то есть тех, в названии которых встречается фрагмент **userfield**, следует установить для соответствующей роли CRM право доступа "Разрешить изменять настройки".
2. Чтобы в методах, работающих с пользовательскими полями, получить названия полей на текущем языке портала, надо в фильтр добавить:

```
'LANG' => 'ru'
```

3. Для корректного обновления множественных полей, таких как телефон и email, в CRM надо передавать id текущего значения, пример:

```
"PHONE": [ { "ID":245570, "VALUE":
"555100501888", "VALUE_TYPE": "WORK" } ],
```

4. Для загрузки файлов используйте такой код:

```
{
            "UF_CRM_1499876148": [

{"fileData": ["test.txt", "dfgdfgdfgh"]},
```

```
{"fileData": ["test2.txt", "dfgdfgdfgh"]}
]
        }
```

crm.userfield.enumeration.field

```
crm.userfield.enumeration.fields
```

Возвращает описание полей для пользовательского поля типа "enumeration" (список).

Пример

```
BX24.callMethod(
        "crm.userfield.enumeration.fields",
        {},
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

crm.userfield.fields

```
crm.userfield.fields
```

Возвращает описание полей для пользовательских полей.

Пример

```
BX24.callMethod(
        "crm.userfield.fields",
        {},
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

© «Битрикс», 2001-2008, «1С-

crm.userfield.settings.fields

```
crm.userfield.settings.fields(type)
```

Возвращает описание полей настроек для типа пользовательского поля.

Параметры

Параметр	Описание
type	Тип пользовательского поля. Значение из списка возвращаемого методом crm.userfield.types.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.userfield.settings.fields",
        {
                type: "string"
        },
        function(result)
```

```
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

crm.userfield.types

```
crm.userfield.types
```

Возвращает список типов пользовательских полей.

Содержит описания типов:

- string
- integer
- double
- boolean
- datetime
- enumeration
- iblock_section
- iblock_element
- employee
- crm_status
- crm
- address
- money
- url

Также будут возвращены типы пользовательских полей, зарегистрированные текущим приложением.

Пример

```
BX24.callMethod(
        "crm.userfield.types",
        {},
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

crm.automation.trigger

Активирует триггер Webhook, настроенный в автоматизации CRM.

Параметры

Параметр	Описание
target	Целевой объект для автоматизации, указывается в виде TYPENAME_ID (например, LEAD_25).
code	Уникальный символьный код триггера, настроенного в Автоматизации на конкретный статус/стадию документа.

Результат возвращается в виде true или ошибки. При обнаружении нескольких триггеров по указанной в параметре **target** сущности срабатывает первый, который устанавливает более ранний статус сущности.

Пример

```
        BX24.callMethod(
"crm.automation.trigger",
{
target: 'LEAD_25'
},
function(result)
{
```

```
if(result.error())
console.error(result.error());
else
console.dir(result.data());
}
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.automation.trigger.add

Метод добавляет триггер. Возвращает *true* или ошибку с описанием.

Параметры

Метод	Описание	С версии
CODE	Внутренний уникальный (в рамках приложения) идентификатор триггера. Должен соответствовать маске [a-z0-9\.\-_]	
NAME	Название триггера	

Примеры

```
function addTrigger(code, name)
{
   if (!name)
      name = window.prompt('Enter trigger
name:');

   BX24.callMethod(
```

```
      'crm.automation.trigger.add',
      {
         "CODE": code,
         "NAME": name
      },
      function(result)
      {
         if (result.error())
            alert("Error: " +
result.error());
         else
         {
            alert("Success: " +
result.data());
         }
      }
   );
}
```

crm.automation.trigger.delete

Метод удаляет триггер. Возвращает *true* или ошибку с описанием.

Параметры

Метод	Описание	С версии
CODE	Внутренний уникальный (в рамках приложения) идентификатор триггера. Должен соответствовать маске [a-z0-9\.\-_]	

Пример

```
function deleteTrigger(code)
{
   BX24.callMethod(
      'crm.automation.trigger.delete',
      {
         "CODE": code
      },
      function(result)
      {
         if(result.error())
```

```
            alert("Error: " +
result.error());
         else
         {
            alert("Success: " +
result.data());
         }
      }
   );
}
```

crm.automation.trigger.execut

Метод, запускающий выполнение триггера. Возвращает *true* или ошибку с описанием

Параметры

Метод	Описание	С версии
CODE	Внутренний уникальный (в рамках приложения) идентификатор триггера. Должен соответствовать маске [a-z0-9\.\-_]	
OWNER_TYPE_ID	Тип сущности CRM (Лид или Сделка по справочнику crm.enum.ownertype)	
OWNER_ID	Идентификатор сущности	

Пример

```
function executeTrigger(code)
{
   BX24.selectCRM({
      entityType: ['lead', 'deal']
```

```
   }, function(selected)
   {
      var typeId, id;
      if (selected['lead'] &&
selected['lead'][0])
      {
         typeId = 1;
         id = selected['lead'][0]
['id'].substring(2);
      }
      else if (selected['deal'] &&
selected['deal'][0])
      {
         typeId = 2;
         id = selected['deal'][0]
['id'].substring(2);
      }

      BX24.callMethod(
         'crm.automation.trigger.execute',
         {
            CODE: code,
            OWNER_TYPE_ID: typeId,
            OWNER_ID: id
         },
         function(result)
         {
            if(result.error())
               alert("Error: " +
result.error());
            else
            {
               alert("Success: " +
result.data());
            }
         }
      );
```

```
    });
}
```

crm.automation.trigger.list

Метод для получения списка приложений и триггеров. Возвращает массив добавленных приложением триггеров с полями `NAME` и `CODE`.

Праметры

Без параметров

Пример

```
function getList()
{
   BX24.callMethod(
      'crm.automation.trigger.list',
      {},
      function(result)
      {
         if(result.error())
            alert("Error: " +
result.error());
         else
         {
            alert("Success: data is in
console");
            console.log(result.data());
         }
      }
```

```
    );
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.externalchannel.activity.co

Описание

```
crm.externalchannel.activity.company
```

Создает дело "Документ от компании".

Параметры

См. параметры функций.

Пример вызова для Дела компании

```
BX24.callMethod(
   "crm.externalchannel.activity.company",
   {
        batch:[{
            agent{
                "fields":{
                        "обновляемое
поле":"значение",

"ORIGIN_VERSION":"ascFSrbtJfIpEPIEPEnCg==",
```

"ORIGIN_ID":"fff6e1b4-55bc-11d9-848a-00112f43529a-10001",
},
"external_fields":{
"EXTERNAL_URL":"company_id=2001",
},
"REQUISITE":[{
"XML_ID":"faa1c2a9-55bc-11d9-848a-00112f43529a-40001",
"RQ_ADDR":{
"PRIMARY":{
"ADDRESS_1":"#Улица, дом, корпус, строение#",
"ADDRESS_2":"#Квартира / офис#",
"CITY":"#Город#",
"POSTAL_CODE":"#Почтовый индекс#",
"REGION":"#Район#",
"PROVINCE":"#Область#",
"COUNTRY":"#Страна#",
"COUNTRY_CODE":""
},
"HOME":{
"ADDRESS_1":"#Улица, дом, корпус, строение#",
"ADDRESS_2":"#Квартира / офис#",

"CITY":"#Город#",

"POSTAL_CODE":"#Почтовый индекс#",

"REGION":"#Район#",

"PROVINCE":"#Область#",

"COUNTRY":"#Страна#",

"COUNTRY_CODE":""
},

"REGISTERED":{

"ADDRESS_1":"#Улица, дом, корпус,
строение#",

"ADDRESS_2":"#Квартира / офис#",

"CITY":"#Город#",

"POSTAL_CODE":"#Почтовый индекс#",

"REGION":"#Район#",

"PROVINCE":"#Область#",

"COUNTRY":"#Страна#",

"COUNTRY_CODE":""
},

"BENEFICIARY":{

"ADDRESS_1":"#Улица, дом, корпус,

строение#",

"ADDRESS_2":"#Квартира / офис#",

"CITY":"#Город#",

"POSTAL_CODE":"#Почтовый индекс#",

"REGION":"#Район#",

"PROVINCE":"#Область#",

"COUNTRY":"#Страна#",

"COUNTRY_CODE":""
}
},

"RQ_NAME":"#Организация#",

"RQ_FIRST_NAME":"",

"RQ_LAST_NAME":"",

"RQ_SECOND_NAME":""

"RQ_COMPANY_NAME":"#Сокращенное наименование организации#",

"RQ_COMPANY_FULL_NAME":"#Полное наименование организации#",

"RQ_COMPANY_REG_DATE":"#Дата государственной регистрации#",

"RQ_DIRECTOR":"#Ген. директор#",

"RQ_ACCOUNTANT":"#Гл. бухгалтер#",

"RQ_CEO_NAME":"",

"RQ_CEO_WORK_POS":"",

"RQ_CONTACT":"",

"RQ_EMAIL":"",

"RQ_PHONE":"",
"RQ_FAX":"",

"RQ_IDENT_DOC":""

"RQ_IDENT_DOC_SER":"",

"RQ_IDENT_DOC_NUM":"",

"RQ_IDENT_DOC_DATE":"",

"RQ_IDENT_DOC_ISSUED_BY":"",

"RQ_IDENT_DOC_DEP_CODE":"",

"RQ_INN":"#ИНН#",
"RQ_KPP":"",

"RQ_USRLE":"",

"RQ_IFNS":"",

"RQ_OGRN":"#ОГРН#",

"RQ_OGRNIP":"#ОКПО#",

"RQ_OKPO":"#ОКТМО#",

"RQ_OKTMO":"",

"RQ_OKVED":"",

"RQ_EDRPOU":"",

"RQ_DRFO":"",
"RQ_KBE":"",
"RQ_IIN":"",
"RQ_BIN":"",

"RQ_VAT_PAYER":"",

"RQ_VAT_ID":"",

"RQ_VAT_CERT_SER":"",

"RQ_VAT_CERT_NUM":"",

"RQ_VAT_CERT_DATE":"",

"RQ_RESIDENCE_COUNTRY":"",

"BANK_DETAILS":[{"

"RQ_BANK_NAME":"#Наименование банка#",

"RQ_BANK_ADDR":"#Адрес банка#",

"RQ_BANK_ROUTE_NUM":"",

"RQ_BIK":"#БИК#",

"RQ_MFO":"",

"RQ_ACC_NAME":"",

```
"RQ_ACC_NUM":"#Расчетный счёт#",

"RQ_IIK":"",

"RQ_ACC_CURRENCY":"#Валюта счёта#",

"RQ_COR_ACC_NUM":"#Кор. счёт#",

"RQ_IBAN":"",

"RQ_SWIFT":"#SWIFT#",

"RQ_BIC":"",

"COMMENTS":"#Комментарий#",

"XML_ID":"caa2g4q8-55bc-11d9-848a-
00112f43529a-50001"
                                    }]
                                }]
            },
            activity{
                "fields":{
                    "обновляемое
поле":"значение"
                },
                "external_fields":{
                    "NUMBER":"N-
1\/20160518",
                    "MANAGER":"",
                    "TYPE_ID":"Реализация"
                     }
            }
        }],
        params:{
```

```
"CHANNEL_ID":"ext_channel.574b405083de90.990
16531"
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
```

Для документов типа Реализация достаточно передавать следующий список полей:

```
batch:[{
  "agent":{
      "fields":{
            "обновляемое поле":"значение",
            ...
       },
       "external_fields":{
            ...
       }
 "activity":{
        "fields":{

"ORIGIN_ID":"A_xmlid_1_1_20160516",
            "SUBJECT":"Название документа/
дела на портале ",
            "START_TIME":"2016-05-
20T12:00:50+04:00",
            "DESCRIPTION":"Произвольное
```

```
описание документа/дела",
            "RESULT_VALUE":"1",
            "RESULT_SUM":"300",
            "RESULT_CURRENCY_ID":"RUB"
        },
        "external_fields":{
            "NUMBER":"N-1\/20160518",
            "MANAGER":"Иван Иванов",
        }
}],
params:{
     ...
}
```

Удаление реквизитов

Если на обновление поступил контрагент без реквизитов, то есть отсутствует массив REQUISITE, то будут удалены все его реквизиты, включая банковские, и адреса. Если реквизиты переданы, но не переданы банковские BANK_DETAILS или адреса RQ_ADDR, то будет удалена та сущность, поле которой не передано.

Языковые ограничения

Поля реквизитов, на которые накладываются по умолчанию языковые ограничения.

Поля, которые будут доступны в форме редактирования реквизитов для языка RU и дефолтном наборе пресетов:

```
RQ_ADDR
RQ_NAME
RQ_FIRST_NAME
RQ_LAST_NAME
RQ_SECOND_NAME
RQ_COMPANY_NAME
RQ_COMPANY_FULL_NAME
RQ_COMPANY_REG_DATE
RQ_DIRECTOR
RQ_ACCOUNTANT
RQ_CEO_NAME
RQ_CEO_WORK_POS
RQ_CONTACT
RQ_EMAIL
RQ_PHONE
RQ_FAX
RQ_IDENT_DOC
RQ_IDENT_DOC_SER
RQ_IDENT_DOC_NUM
RQ_IDENT_DOC_DATE
RQ_IDENT_DOC_ISSUED_BY
RQ_IDENT_DOC_DEP_CODE
RQ_INN
RQ_KPP
RQ_USRLE
RQ_IFNS
RQ_OGRN
RQ_OGRNIP
RQ_OKPO
RQ_OKTMO
RQ_OKVED
```

Ограничения для банковских реквизитов для языка RU

```
RQ_BANK_NAME
RQ_BANK_ADDR
RQ_BIK
RQ_ACC_NUM
RQ_ACC_CURRENCY
RQ_COR_ACC_NUM
RQ_SWIFT
```

crm.externalchannel.activity.co

```
crm.externalchannel.activity.contact
```

Создает дело "Документ от контакта".

Параметры

См. параметры функций.

Пример вызова для Дела контакта

Аналогично вызову для Компании.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.externalchannel.company

```
crm.externalchannel.company
```

Импортирует компанию и создает дело "Импорт компании".

Параметры

См. параметры функций.

Пример вызова для Компании

Аналогично вызову для Контакта.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.externalchannel.contact

```
crm.externalchannel.contact
```

Импортирует Контакт и создает дело "Импорт контакта".

Параметры

См. параметры функций.

Пример вызова для Контакта (для вызова Компании структура аналогична)

```
BX24.callMethod(
   "crm.externalchannel.contact",
   {
    batch:[{
            "agent":{
                    "fields":{
                        "обновляемое
поле":"значение",

"ORIGIN_VERSION":"qqohNDrbtJfIpEPIEPEnCg==",
```

"ORIGIN_ID":"fff6e1b4-55bc-11d9-848a-00112f43529a-00001",
},
"EXTERNAL_FIELDS":{
"COMPANY_ORIGIN_ID":"dee6e1b4-55bc-11d9-848a-00112f43529a-00001",
"EXTERNAL_URL":"contact_id=1001"
},
"REQUISITE":[{
"XML_ID":"faa1c2a9-55bc-11d9-848a-00112f43529a-00001",
"RQ_ADDR":{
"PRIMARY":{
"ADDRESS_1":"#Улица, дом, корпус, строение#",
"ADDRESS_2":"#Квартира / офис#",
"CITY":"#Город#",
"POSTAL_CODE":"#Почтовый индекс#",
"REGION":"#Район#",
"PROVINCE":"#Область#",
"COUNTRY":"#Страна#",
"COUNTRY_CODE":""
},
"HOME":{

"ADDRESS_1":"#Улица, дом, корпус, строение#",

"ADDRESS_2":"#Квартира / офис#",

"CITY":"#Город#",

"POSTAL_CODE":"#Почтовый индекс#",

"REGION":"#Район#",

"PROVINCE":"#Область#",

"COUNTRY":"#Страна#",

"COUNTRY_CODE":""
},

"REGISTERED":{

"ADDRESS_1":"#Улица, дом, корпус, строение#",

"ADDRESS_2":"#Квартира / офис#",

"CITY":"#Город#",

"POSTAL_CODE":"#Почтовый индекс#",

"REGION":"#Район#",

"PROVINCE":"#Область#",

"COUNTRY":"#Страна#",

"COUNTRY_CODE":""
},

"BENEFICIARY":{

"ADDRESS_1":"#Улица, дом, корпус, строение#",

"ADDRESS_2":"#Квартира / офис#",

"CITY":"#Город#",

"POSTAL_CODE":"#Почтовый индекс#",

"REGION":"#Район#",

"PROVINCE":"#Область#",

"COUNTRY":"#Страна#",

"COUNTRY_CODE":""
}
},

"RQ_NAME":"#Организация#",

"RQ_FIRST_NAME":"",

"RQ_LAST_NAME":"",

"RQ_SECOND_NAME":""

"RQ_COMPANY_NAME":"#Сокращенное наименование организации#",

"RQ_COMPANY_FULL_NAME":"#Полное наименование организации#",

"RQ_COMPANY_REG_DATE":"#Дата государственной

регистрации#",

"RQ_DIRECTOR":"#Ген. директор#",

"RQ_ACCOUNTANT":"#Гл. бухгалтер#",

"RQ_CEO_NAME":"",

"RQ_CEO_WORK_POS":"",

"RQ_CONTACT":"",

"RQ_EMAIL":"",

"RQ_PHONE":"",
"RQ_FAX":"",

"RQ_IDENT_DOC":""

"RQ_IDENT_DOC_SER":"",

"RQ_IDENT_DOC_NUM":"",

"RQ_IDENT_DOC_DATE":"",

"RQ_IDENT_DOC_ISSUED_BY":"",

"RQ_IDENT_DOC_DEP_CODE":"",

"RQ_INN":"#ИНН#",
"RQ_KPP":"",

"RQ_USRLE":"",

"RQ_IFNS":"",

"RQ_OGRN":"#ОГРН#",

"RQ_OGRNIP":"#ОКПО#",

"RQ_OKPO":"#ОКТМО#",

"RQ_OKTMO":"",

"RQ_OKVED":"",

"RQ_EDRPOU":"",

"RQ_DRFO":"",
"RQ_KBE":"",
"RQ_IIN":"",
"RQ_BIN":"",

"RQ_VAT_PAYER":"",

"RQ_VAT_ID":"",

"RQ_VAT_CERT_SER":"",

"RQ_VAT_CERT_NUM":"",

"RQ_VAT_CERT_DATE":"",

"RQ_RESIDENCE_COUNTRY":"",

"BANK_DETAILS":[{"

"RQ_BANK_NAME":"#Наименование банка#",

"RQ_BANK_ADDR":"#Адрес банка#",

"RQ_BANK_ROUTE_NUM":"",

"RQ_BIK":"#БИК#",

```
"RQ_MFO":"",

"RQ_ACC_NAME":"",

"RQ_ACC_NUM":"#Расчетный счёт#",

"RQ_IIK":"",

"RQ_ACC_CURRENCY":"#Валюта счёта#",

"RQ_COR_ACC_NUM":"#Кор. счёт#",

"RQ_IBAN":"",

"RQ_SWIFT":"#SWIFT#",

"RQ_BIC":"",

"COMMENTS":"#Комментарий#",

"XML_ID":"caa2g4q8-55bc-11d9-848a-
00112f43529a-00001"
                                    }]
                                }]
                        }
            }],
            params:{

"CHANNEL_ID":"ext_channel.574b405083de90.990
16531"
            }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
```

```
        else
            console.dir(result.data());
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.externalchannel.connector

Описание

```
crm.externalchannel.connector.register
```

Регистрирует коннектор внешнего канала.

Параметры

См. параметры функций.

Пример вызова для регистрации коннектора

```
BX24.callMethod(
   "crm.externalchannel.connector.register",
   {
    fields:{
        "NAME":"Коннектор 1С",
        "ORIGINATOR_ID":"1С",

"EXTERNAL_SERVER_HOST":"localhost:9090\/"
        }
   },
   function(result)
```

```
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
```

В зависимости от того, является ли контрагент Контактом или Компанией, внешним сервисом вызываются команды:

```
crm.externalchannel.company
crm.externalchannel.contact
crm.externalchannel.activity.company
crm.externalchannel.activity.contact
```

Коды ошибок

Код	Статус	Текст по умолчанию
ERROR_IMPORT_BATCH	400 Bad Request	Batch is not defined.
ERROR_CONNECTOR_NOT_FOUND	404 Not Found	Connector not found!
ERROR_CONNECTOR_CREATE	500 Internal Server Error	Connector not created.
ERROR_CONNECTOR_REGISTRATION	400 Bad Request	

ERROR_CONNECTOR_INVALID	403 Forbidden	Connector is invalid!
ERROR_PRESET_NOT_FOUND	403 Forbidden	Preset is not defined.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.externalchannel.connector

```
crm.externalchannel.connector.unregister(fie
lds)
```

Удаляет коннектор внешнего канала.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей адреса. Где "поле" может принимать два значения "TYPE_ID", "ORIGINATOR_ID"

Предопределенный справочник типов параметра TYPE_ID:

CUSTOM	Кастомный тип
BITRIX	БУС
ONE_C	1С
WORDPRESS	Wordpress
JOOMLA	Joomla

DRUPAL	Drupal
MAGENTO	Magento

Пример

```
BX24.callMethod(
   "crm.externalchannel.connector.register",
   {
    fields:{
        "ORIGINATOR_ID":"sale",
        "TYPE_ID":"2"
        }
   },
   function(result)
   {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
```

Каталог::crm.catalog.fields

```
crm.catalog.fields()
```

Возвращает описание полей каталога товаров.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(
"crm.catalog.fields",
                        {},
                        function(result)
                        {
if(result.error())
console.error(result.error());
                                else
console.dir(result.data());
```

```
                                        }
                        );
```

Поля

Поле	Описание	Тип	Примечание
ID	Идентификатор	integer	Только для чтения
NAME	Название	string	
ORIGINATOR_ID	Идентификатор источника данных	string	Используется только для привязки к внешнему источнику.
ORIGIN_ID	Идентификатор элемента в источнике данных	string	Используется только для привязки к внешнему источнику.
XML_ID	Символьный код	string	Только для чтения

Каталог::crm.catalog.get

```
crm.catalog.get(id)
```

Возвращает товарный каталог по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор товарного каталога.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(
                        "crm.catalog.get",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
```

```
console.dir(result.data());
                            }
                  );
```

Каталог::crm.catalog.list

Возвращает список товарных каталогов. Является реализацией списочного метода для товарных каталогов.

Параметры

См. описание списочных методов.

Пример

```
        BX24.callMethod(
                        "crm.catalog.list",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
                                {

console.dir(result.data());

if(result.more())

result.next();
                                }
```

```
                                        }
                                );
```

© «Битрикс», 2001-2008, «1С-

Валюты::crm.currency.add

```
crm.currency.add(fields)
```

Создаёт новую валюту.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей валюты, где "поле" может принимать значения из возвращаемых методом crm.currency.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.currency.fields и посмотрите формат пришедших значений этих полей.

Внимание! Настоятельно рекомендуется определить набор локализаций в поле LANG (см. crm.currency.localizations.set). Если поле LANG отсутствует, то необходимо вызвать для каждого активного языка метод crm.currency.localizations.set. В противном случае при выводе цены в этой валюте будут использоваться настройки форматирования по умолчанию.

Пример

```
        BX24.callMethod(
        "crm.currency.add",
        {
                fields:
                {
                        "CURRENCY": "KWD",
                        "AMOUNT_CNT": 1,
                        "AMOUNT": 112.25,
                        "SORT": 1000,
                        "LANG":
                        {
                                ru:
                                {

DEC_POINT:'.',

FORMAT_STRING:'# динар',

FULL_NAME:'Кувейтский динар',

THOUSANDS_VARIANT:'C',

DECIMALS: 2,

HIDE_ZERO: "Y" //Если десятичная часть
нулевая, то отбрасываем её при выводе
                                },
                                en:
                                {

DEC_POINT:',',

FORMAT_STRING:'# KD',

FULL_NAME:'Kuwaiti Dinar',

THOUSANDS_VARIANT:'C',
```

```
DECIMALS: 2,

HIDE_ZERO: "Y"
                                        }
                                }
                        }
                },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.info("Создана валюта с ID " +
result.data());
                }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Валюты::crm.currency.base.ge

```
crm.currency.base.get()
```

Метод позволяет получить символьный идентификатор базовой валюты.

Параметры

Без параметров

Пример

```
BX24.callMethod(
            "crm.currency.base.get",
           { },
           function(result)
           {
               if(result.error())

console.error(result.error());
               else

console.dir(result.data());
           }
       );
```

Валюты::crm.currency.base.se

```
crm.currency.base.set(id)
```

Метод устанавливает валюту в качестве базовой. Если попробовать сделать базовой валюту, которая и так ей является - вернется *null*. Иначе - *true/false* (успех или ошибка).

Параметры

Параметр	Описание	С версии
id	Символьный код валюты, которую необходимо сделать базовой	

Пример

```
var cur = prompt("Введите символьный код
валюты");
       BX24.callMethod(
           "crm.currency.base.set",
           { id: cur },
           function(result)
```

```
            {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
            }
        );
```

Валюты::crm.currency.delete

```
crm.currency.delete(id))
```

Удаляет валюту.

Параметры

Параметр	Описание
id	Символьный идентификатор валюты.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(

"crm.currency.delete",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
```

```
                                    else

console.info(result.data());
                            }
                    );
```

Валюты::crm.currency.fields

```
crm.currency.fields()
```

Возвращает описание полей валюты.

Параметры

Без параметров.

Пример

```
BX24.callMethod(

"crm.currency.fields",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else

console.dir(result.data());
```

```
                                        }
                                );
```

Поля

Поле	Описание	Тип	Примеч
AMOUNT	Номинал	double	
AMOUNT_CNT	Курс обмена	int	
CURRENCY	Код валюты	string	
DATA_UPDATE	Дата изменения	datetime	Только для чтения
DECIMALS	Количество знаков после запятой	int	Только для чтения
DEC_POINT	Десятичная точка при выводе	string	Только для чтения
FORMAT_STRING	Строка формата для вывода валюты	string	Только для чтения
FULL_NAME	Название	string	Только для чтения
LANG	Языковое обозначение	currency_localization	Множестве
LID	Привязка к языку	string	Только для чтения
SORT	Сортировка	int	

THOUSANDS_SEP	Разделитель триад	string	Только для чтения. Че будут отде сотни от ты тысячи от миллионов далее.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Валюты::crm.currency.get

```
crm.currency.get(id)
```

Возвращает валюту по символьному идентификатору.

Параметры

Параметр	Описание
id	Символьный идентификатор валюты.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(
                        "crm.currency.get",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
```

```
console.dir(result.data());
                            }
                  );
```

Валюты::crm.currency.list

Возвращает список валют. Является реализацией списочного метода для валют. Обратите внимание, что в данной реализации параметры "filter", "select" и "navigation" не поддерживаются.

Параметры

См. описание списочных методов.

Пример

```
        BX24.callMethod(
                        "crm.currency.list",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                              else
                              {

console.dir(result.data());

if(result.more())

result.next();
```

```
                                                    }
                                                }
                                );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Валюты::crm.currency.localiza

Удаляет выбранные локализации для валюты, указанной по символьному идентификатору.

Параметры

Параметр	Описание
id	Символьный идентификатор валюты.
lids	Массив идентификаторов языков, локализации которых требуется удалить.

Пример

```
        var id = prompt("Введите ID
валюты");
                var langId = prompt("Введите
ID языка (ru, en, de)");
                BX24.callMethod(

"crm.currency.localizations.delete",
                        {
                                id:id,
                                lids:[
langId ]
                        },
```

```
                            function(result)
                            {

if(result.error())

console.error(result.error());
                                  else

console.info(result.data());
                            }
                  );
```

Валюты::crm.currency.localiza

```
crm.currency.localizations.fields
```

Возвращает описание локализаций для валюты.

Пример

```
        BX24.callMethod(
        "crm.currency.localizations.fields",
        {},
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

© «Битрикс», 2001-2008, «1С-

Валюты::crm.currency.localiza

```
crm.currency.localizations.get(id)
```

Возвращает локализации для валюты, указанной по символьному идентификатору.

Параметры

Параметр	Описание
id	Символьный идентификатор валюты.

Пример

```
        BX24.callMethod(

"crm.currency.localizations.get",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
```

```
                                        else
                                        {

console.dir(result.data());
                                        }
                                }
                        );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Валюты::crm.currency.localiza

```
crm.currency.localizations.set(id,
localizations)
```

Устанавливает локализации для валюты, указанной по символьному идентификатору.

Параметры

Параметр	Описание
id	Символьный идентификатор валюты.
localizations	Набор локализаций - массив вида array("язык" => array("поле"=>"значение"[, ...])), содержащий значения полей локализаций, где "язык" - идентификатор языка, "поле" - одно из возвращаемых методом crm.currency.localizations.fields значений.

Пример

```
BX24.callMethod(
'crm.currency.localizations.set',
{
```

```
                id: "KWD",
                localizations:
                {
                        ru:
                        {

DEC_POINT:'.',

FORMAT_STRING:'# динар',

FULL_NAME:'Кувейтский динар',

THOUSANDS_VARIANT:'C',
                                DECIMALS: 2,
                                HIDE_ZERO:
"Y" //Если десятичная часть нулевая, то
отбрасываем её при выводе
                        },
                        en:
                        {

DEC_POINT:',',

FORMAT_STRING:'# KD',

FULL_NAME:'Kuwaiti Dinar',

THOUSANDS_VARIANT:'C',
                                DECIMALS: 2,
                                HIDE_ZERO:
"Y"
                        }
                }
        },
        function(result)
        {
                if(result.error())
```

```
console.error(result.error());
                else

console.info(result.data());
        }
);
```

Валюты::crm.currency.update

```
crm.currency.update(id, fields)
```

Обновляет существующую валюту.

Параметры

Параметр	Описание
id	Символьный идентификатор валюты.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.currency.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.currency.fields и посмотрите формат пришедших значений этих полей.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
```

```
"crm.currency.update",
                        {
                                id: id,
                                fields:
                                {

"AMOUNT_CNT": 1,

"AMOUNT": 112.10,

"SORT": 9000
                                }
                        },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
                                {

console.info(result.data());
                                }
                        }
                );
```

onCrmCurrencyAdd

Событие, вызываемое после создании валюты.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора созданной валюты.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmCurrencyDelete

Событие, вызываемое после удалении валюты.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора удаленной валюты.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmCurrencyUpdate

Событие, вызываемое после обновлении валюты.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора обновленной валюты.

© «Битрикс», 2001-2008, «1С-Битрикс» 2008-2022

crm.measure.add

```
crm.measure.add(fields)
```

Добавляет новую единицу измерения.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей единицы измерения. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.measure.fields и посмотрите формат пришедших значений этих полей. Поля для добавления единицы измерения: ▪ CODE - Код; ▪ MEASURE_TITLE - Наименование единицы измерения; ▪ SYMBOL_RUS - Условное обозначение; ▪ SYMBOL_INTL - Условное обозначение (международное); ▪ SYMBOL_LETTER_INTL - Кодовое буквенное обозначение (международное); ▪ IS_DEFAULT - По умолчанию.

Пример

```
        BX24.callMethod(
                "crm.measure.add",
                {
                        fields: {
                                "CODE":
"212",

"MEASURE_TITLE": "Ватт",

"SYMBOL_RUS": "Вт",

"SYMBOL_INTL": "W",

"SYMBOL_LETTER_INTL": "WTT",

"IS_DEFAULT": "N"
                        }
                },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.info("Создана единица измерения с ID
" + result.data());
                }
        );
```

© «Битрикс», 2001-2008, «1С-

crm.measure.delete

```
crm.measure.delete(id)
```

Удаляет единицу измерения.

Параметры

Параметр	Описание
id	Идентификатор единицы измерения.

Пример

```
        var id = prompt("Введите ID");
BX24.callMethod(
        "crm.measure.delete",
        {id: id},
        function (result)
        {
                if (result.error())

console.error(result.error());
                else
```

```
console.info(result.data());
        }
);
```

crm.measure.fields

```
crm.measure.fields()
```

Возвращает описание полей для единиц измерений.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(

"crm.measure.fields",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else

console.dir(result.data());
                        }
```

```
            );
```

Поля

Поле	Описание	Тип	Примечани
CODE	Код единицы	integer	Обязательное
ID	Идентификатор	integer	Только для чтения
IS_DEFAULT	По умолчанию	char	
MEASURE_TITLE	Наименование единицы измерения	string	Обязательное.
SYMBOL_INTL	Условное обозначение (международное)	string	
SYMBOL_LETTER_INTL	Кодовое буквенное обозначение (международное)	string	
SYMBOL_RUS	Условное обозначение	string	

crm.measure.get

```
crm.measure.get(id)
```

Возвращает единицу измерения по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор единицы измерения.

Пример

```
        var id = prompt("Введите ID");
BX24.callMethod(
        "crm.measure.get",
        {id: id},
        function (result)
        {
                if (result.error())

console.error(result.error());
                else
```

```
console.dir(result.data());
        }
);
```

crm.measure.list

```
crm.measure.list()
```

Возвращает список единиц измерений.

Параметры

См. описание списочных методов.

Пример

```
BX24.callMethod(
        "crm.measure.list",
        {
                order: {"ID": "ASC"},
                filter: {"IS_DEFAULT": "Y"},
                select: ["ID", "CODE",
"STAGE_ID", "SYMBOL_RUS", "SYMBOL_INTL"]
        },
        function (result)
        {
                if (result.error())

console.error(result.error());
                else
```

```
                    {
console.dir(result.data());
                            if (result.more())
result.next();
                    }
          }
);
```

crm.measure.update

```
crm.measure.update(id, fields)
```

Обновляет существующую единицу измерения.

Параметры

Параметр	Описание
id	Идентификатор единицы измерения.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.measure.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.measure.fields и посмотрите формат пришедших значений этих полей.

Пример

```
        var id = prompt("Введите ID");
var title = prompt("Введите новое
```

```
наименование для единицы измерения");
BX24.callMethod(
        "crm.measure.update",
        {
                id: id,
                fields: {
                        "MEASURE_TITLE":
title
                }
        },
        function (result)
        {
                if (result.error())

console.error(result.error());
                else

console.info(result.data());
        }
);
```

onCrmMeasureAdd

Событие вызывается после добавления новой единицы измерения на портале. Возвращает массив:

```
array('FIELDS' => array('ID' => $id))
```

где $id - идентификатор созданной единицы измерения.

В случае ошибок вызывается исключение `\Bitrix\Rest\RestException`.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmMeasureDelete

Событие вызывается после удаления единицы измерения на портале. Возвращает массив:

```
array('FIELDS' => array('ID' => $id))
```

где $id - идентификатор созданной единицы измерения.

В случае ошибок вызывается исключение `\Bitrix\Rest\RestException`.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmMeasureUpdate

Событие вызывается после изменения единицы измерения на портале. Возвращает массив:

```
array('FIELDS' => array('ID' => $id))
```

где $id - идентификатор созданной единицы измерения.

В случае ошибок вызывается исключение `\Bitrix\Rest\RestException`.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.productsection.add

```
crm.productsection.add(fields)
```

Создаёт новый раздел товаров.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей раздела товаров. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.productsection.fields и посмотрите формат пришедших значений этих полей.

Пример

```
        var catalogId = prompt("Введите ID
каталога");
                var sectionId =
prompt("Введите ID родительской секции (если
в корне, то 0))");
                var sectionName =
```

```
prompt("Введите название секции");
                BX24.callMethod(

"crm.productsection.add",
                        {
                                fields:
                                {

CATALOG_ID: catalogId,

NAME: sectionName,

SECTION_ID: sectionId
                                }
                        },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else

console.info("Создан новый раздел с ID " +
result.data());
                        }
                );
```

crm.productsection.delete

```
crm.productsection.delete(id)
```

Удаляет раздел каталога товаров.

Параметры

Параметр	Описание
id	Идентификатор раздела товаров.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(

"crm.productsection.delete",
                        { id: id },
                        function(result)
                        {

if(result.error())
```

```
console.error(result.error());
                                else

console.info(result.data());
                        }
                );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.productsection.fields

```
crm.productsection.fields()
```

Возвращает описание полей раздела товара.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(

"crm.productsection.fields",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else

console.dir(result.data());
```

```
                                        }
                        );
```

Поля

Поле	Описание	Тип	Примечание
CATALOG_ID	Идентификатор каталога	integer	Неизменяемое
ID	Идентификатор раздела	integer	Только для чтения
NAME	Название раздела	string	Обязательное
SECTION_ID	Идентификатор привязанного раздела	integer	
XML_ID	Символьный код	string	

crm.productsection.get

```
crm.productsection.get(id)
```

Возвращает раздел товаров по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор раздела товаров.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(

"crm.productsection.get",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
```

```
                                    else

console.dir(result.data());
                            }
                  );
```

crm.productsection.list

Возвращает список разделов товаров по фильтру. Является реализацией списочного метода для разделов товаров. Ожидается, что в фильтре будет определён параметр CATALOG_ID. В противном случае разделы будут выбираться из каталога по умолчанию.

Параметры

См. описание списочных методов.

Пример

```
        var catalogId = prompt("Введите ID
каталога");
                BX24.callMethod(

"crm.productsection.list",
                        {
                                order: {
"NAME": "ASC" },
                                filter: {
"CATALOG_ID": catalogId },
                                select: [
"ID", "NAME" ]
                        },
                        function(result)
                        {
```

```
if(result.error())

console.error(result.error());
                                else
                                {

console.dir(result.data());

if(result.more())

result.next();
                                }
                        }
                );
```

crm.productsection.update

```
crm.productsection.update(id, fields)
```

Обновляет существующий раздел товаров.

Параметры

Параметр	Описание
id	Идентификатор раздела товаров.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.productsection.fields. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.productsection.fields и посмотрите формат пришедших значений этих полей.

Пример

```
        var id = prompt("Введите ID");
                var sectionName =
prompt("Введите название секции");
                BX24.callMethod(

"crm.productsection.update",
                        {
                                id: id,
                                fields:
                                {

"NAME": sectionName
                                }
                        },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
                                {

console.info(result.data());
                                }
                        }
                );
```

onCrmProductSectionAdd

Вызывается после добавления раздела. Возвращает массив

```
array('FIELDS' => array('ID' => $id))
```

В случае ошибки либо при создании раздела не в инфоблоке CRM, выводит исключение `\Bitrix\Rest\RestException`.

© «Битрикс», 2001-2008, «1С-Битрикс»: 2008-2022

onCrmProductSectionDelete

Вызывается после удаления раздела. Возвращает массив:

```
array('FIELDS' => array('ID' => $id))
```

В случае ошибки либо при удалении раздела не в инфоблоке CRM, выводит исключение `\Bitrix\Rest\RestException`.

© «Битрикс», 2001-2008, «1С-Битрикс»: 2008-2022

onCrmProductSectionUpdate

Вызывается после изменения раздела. Возвращает массив:

```
array('FIELDS' => array('ID' => $id))
```

В случае ошибки либо при изменении раздела не в инфоблоке CRM, выводит исключение `\Bitrix\Rest\RestException`.

© «Битрикс», 2001-2008, «1С-

crm.productrow.fields

```
crm.productrow.fields()
```

Возвращает описание полей товарных позиций.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(

"crm.productrow.fields",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else

console.dir(result.data());
```

```
                                        }
                                );
```

crm.productrow.list

Возвращает список товарных позиций по фильтру. Является реализацией списочного метода для товарных позиций. Владелец товарных позиций определяется обязательными полями OWNER_TYPE и OWNER_ID, где OWNER_TYPE - односимвольный код типа ("D" - сделка, "L" - лид), OWNER_ID - идентификатор.

Параметры

См. описание списочных методов.

Пример

```
        var ownerType = prompt("Введите тип
владельца (D, L)");
                var ownerId =
prompt("Введите ID владельца");
                BX24.callMethod(

"crm.productrow.list",
                        {
                                filter:
                                {

"OWNER_TYPE": ownerType,

"OWNER_ID": ownerId
                                }
```

```
                            },
                            function(result)
                            {

if(result.error())

console.error(result.error());
                                  else
                                  {

console.dir(result.data());

if(result.more())

result.next();
                                  }
                            }
                  );
```

crm.item.productrow.fields

```
crm.item.productrow.fields()
```

Отдает информацию о полях товарных позиций. Названия полей для входных данных следует брать из этого метода.

Без параметров.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.item.productrow.get

```
crm.item.productrow.get({id: number})
```

Отдает информацию о товарной позиции с идентификатором **id**.

Параметры

Параметр	Описание	С версии
id	Идентификатор товарной позиции	

Ответ:

```
{
        "productRow": {
                "id": 1263,
                "ownerId": 707,
                "ownerType": "Q",
                "productId": 2,
                "productName": "Название
товарной позиции",
                "price": 611,
```

```
                    "priceAccount": 611,
                    "priceExclusive": 611,
                    "priceNetto": 678.89,
                    "priceBrutto": 678.89,
                    "quantity": 1,
                    "discountTypeId": 2,
                    "discountRate": 10,
                    "discountSum": 67.89,
                    "taxRate": 0,
                    "taxIncluded": "N",
                    "customized": "Y",
                    "measureCode": 796,
                    "measureName": "шт",
                    "sort": 0
            }
}
```


Значения в ответе:

- **ownerId** - идентификатор элемента CRM, к которому привязана товарная позиция (например, идентификатор предложения)
- **ownerType** - символьный код типа сущности CRM, к которому привязана товарная позиция. Узнать, какому типу сущности соответствует конкретный код можно через методы класса *\CCrmOwnerTypeAbbr*.
- **productId** - идентификатор товара из каталога, которому соответствует товарная позиция.
- **productName** - название товарной позиции. По умолчанию совпадает с названием товара из каталога, но может быть изменено.
- **price** - стоимость за единицу товарной позиции с учетом скидок и налогов.
- **priceAccount** - стоимость за единицу товарной позиции с учетом скидок и налогов, сконвертированная в валюту отчетов.
- **priceExclusive** - стоимость за единицу товарной позиции с учетом скидок, но без учета налогов
- **priceNetto** - стоимость за единицу товарной позиции без учета скидок и без учета налогов.

- **priceBrutto** - стоимость за единицу товарной позиции с учетом налогов, но без учета скидок.
- **quantity** - количество единиц товарной позиции.
- **discountTypeId** - тип скидки. Может быть 1 для скидки в абсолютном значении и 2 для скидки в процентах. По умолчанию равно 2.
- **discountRate** - процент скидки на товарную позицию.
- **discountSum** - абсолютное значение скидки на товарную позицию.
- **taxRate** - процент налога на товарную позицию.
- **measureCode** - код единицы измерения количества товарной позиции. Настроить можно в разделе Единицы измерения настроек CRM.
- **measureName** - условное обозначение единицы измерения количества товарной позиции. Настроить можно в разделе Единицы измерения настроек CRM.

> **Примечание**: С версии **CRM 21.1800.0** через REST-методы семейства `crm.item.productrow.*` можно менять единицы измерения товарных позиций, передавая только `measureCode`, а `measureName` заполнится автоматически.

- **sort** - коэффициент сортировки.

crm.item.productrow.add

```
crm.item.productrow.add({fields: {}})
```

Метод создает новую товарную позицию с полями **fields**. При этом новая товарная позиция привязывается к элементу CRM, указанному в полях **ownerType** и **ownerId**.

Параметры

Параметр	Описание	С версии
fields	значение полей товарной позиции	

Значения некоторых полей (priceExclusive и др.) будут вычислены автоматически на основании предоставленных данных. Задать их вручную нельзя. Полный список можно read-only полей можно узнать с помощью метода crm.item.productrow.fields.

Метод вернет результат аналогичный вызову метода crm.item.productrow.get на только что созданной товарной позиции.

crm.item.productrow.update

```
crm.item.productrow.update({id: number, fields: {}})
```

Метод обновит товарную позицию с идентификатором id, задав ей новые значения полей из fields. Если какое-то поле будет отсутствовать в fields, то его значение останется неизменным.

Параметры

Параметр	Описание	С версии
id	Идентификатор товарной позиции	
fields	Значение полей товарной позиции, которые необходимо изменить	

Значения некоторый полей (priceExclusive и др.) будут автоматически пересчитаны, если в ходе обновления товарной позиции изменятся значимые для вычислений поля (например, price).

Метод вернет результат аналогичный вызову метода crm.item.productrow.get для изменённой товарной позиции.

crm.item.productrow.set

```
crm.item.productrow.set({ownerType: string,
ownerId: number, productRows: []})
```

Метод привяжет к элементу CRM с типом ownerType и идентификатором ownerId товарные позиции productRows. Метод перезапишет все уже существующие товарные позиции, привязанные к элементу. Таким образом, метод "заменяет" уже существующие товарные позиции на те, что были присланы.

Параметры

Параметр	Описание	С версии
ownerType	Символьный код типа сущности CRM. Узнать, какому типу сущности соответствует конкретный код можно через методы класса \CCrmOwnerTypeAbbr.	
ownerId	Идентификатор элемента CRM	
productRows	Массив товарных позиций. Каждый элемент массива - объект с данными,	

	аналогичными fields из crm.item.productrow.add.	

Значения некоторых полей (priceExclusive и др.) будут вычислены автоматически на основании предоставленных данных. Задать их вручную нельзя. Полный список можно read-only полей можно узнать с помощью метода crm.item.productrow.fields.

Метод вернет результат аналогичный вызову метода crm.item.productrow.list с фильтром по заданному элементу CRM.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.item.productrow.delete

```
crm.item.productrow.delete({id: number})
```

Метод удалит товарную позицию с идентификатором id.

Параметры

Параметр	Описание	С версии
id	Идентификатор товарной позиции	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.item.productrow.list

```
crm.item.productrow.list({order: ?{} = null,
filter: {}, start: ?number = 0})
```

Метод вернет массив товарных позиций

```
{
        "productRows": []
}
```

где каждый элемент массива — это структура, аналогичная результату метода crm.item.productrow.get.

Параметры

Параметр	Описание	С версии
order	Список для сортировки, где ключ — поле, а значение - ASC или DESC.	
filter	Список для фильтрации. Примеры фильтров ниже. Ключи	

	=ownerType и =ownerId являются обязательными.	
start	Сдвиг для постраничной навигации.	

Примеры

Найти все товарные позиции, привязанные к предложению с идентификатором 1

```
{
        "filter": {
                "=ownerType": "Q",
                "=ownerId": 1
        }
}
```

Найти все товарные позиции, привязанные к смарт-процессу с entityTypeId = 128 и с идентификатором 9, у которых скидка больше 10% или цена ниже 1000

```
{
        "filter": {
                "=ownerType": "T80",
                "=ownerId": 9,
                "0": {
                        "logic": "OR",
                        "0": {

">discountRate": 10
                        },
                        "1": {
                                "<price":
```

```
1000
                                }
                        }
                }
}
```

Найти все товарные позиции, привязанные к предложению с идентификатором 2, которые привязаны к товарам из каталога

```
{
        "filter": {
                "=ownerType": "Q",
                "=ownerId": 2,
                "!=productId": 0
        }
}
```

crm.product.add

```
crm.product.add(fields)
```

Создаёт новый товар.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения полей товара. Необходимо обязательно указать CURRENCY_ID для установки цены. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.product.fields и посмотрите формат пришедших значений этих полей.

С версии **CRM 21.700.0** включена поддержка автогенерации символьного кода товара, при условии, что в настройках инфоблока для символьного кода включена генерация и не используется внешний сервис. Задействован метод generateMnemonicCode.

Пример

```
        BX24.callMethod(
                        "crm.product.add",
                        {
                                fields:
                                {

"NAME": "1С-Битрикс: Управление сайтом -
Старт",

"CURRENCY_ID": "RUB",

"PRICE": 4900,

"SORT": 500
                                }
                        },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else

console.info("Создан новый товар с ID " +
result.data());
                        }
                );
```

Добавление файлов в CRM указанным методом имеет свои особенности.

© «Битрикс», 2001-2008, «1С-

crm.product.delete

```
crm.product.delete(id)
```

Удаляет товар.

Параметры

Параметр	Описание
id	Идентификатор товара.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
"crm.product.delete",
{ id: id },
function(result)
{
if(result.error())
console.error(result.error());
```

```
                                    else

console.info(result.data());
                            }
                    );
```

crm.product.fields

```
crm.product.fields()
```

Возвращает описание полей товара.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(
"crm.product.fields",
                        {},
                        function(result)
                        {
if(result.error())
console.error(result.error());
                                else
console.dir(result.data());
```

```
                                        }
                                );
```

Поля

Поле	Описание	Тип	Примечани
ACTIVE	Активен	char	
CATALOG_ID	Идентификатор каталога	integer	Только для чтения
CREATED_BY	Кем создан товар	integer	
CURRENCY_ID	Идентификатор валюты	string	
DATE_CREATE	Дата создания товара	datetime	
DESCRIPTION	Описание	string	
DESCRIPTION_TYPE	Тип описания	string	
DETAIL_PICTURE	Детальная картинка	product_file	
ID	Идентификатор товара	integer	Только для чтения
MEASURE	Единица измерения	integer	
MODIFIED_BY	Кем изменён товар	integer	

NAME	Название	string	Обязательное
PREVIEW_PICTURE	Картинка для анонса	product_file	
PRICE	Цена	double	
SECTION_ID	Идентификатор раздела	integer	
SORT	Сортировка	integer	
TIMESTAMP_X	Дата изменения товара	datetime	Неизменяемое поле
VAT_ID	Идентификатор ставки НДС	integer	
VAT_INCLUDED	НДС включён в цену	char	
XML_ID	Внешний код	string	

© «Битрикс», 2001-2008, «1С-

crm.product.get

```
crm.product.get(id)
```

Возвращает товар по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор товара.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(
                        "crm.product.get",
                        { id: id },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
```

```
console.dir(result.data());
                            }
                  );
```

crm.product.list

Возвращает список товаров по фильтру. Является реализацией списочного метода для товаров. Ожидается, что в фильтре будет определён параметр CATALOG_ID. В противном случае товары будут выбираться из каталога по умолчанию.

Параметры

См. описание списочных методов.

Пример

```
        var catalogId = prompt("Введите ID
каталога");
                BX24.callMethod(
                        "crm.product.list",
                        {
                                order: {
"NAME": "ASC" },
                                filter: {
"CATALOG_ID": catalogId },
                                select: [
"ID", "NAME", "CURRENCY_ID", "PRICE" ]
                        },
                        function(result)
                        {

if(result.error())
```

```
console.error(result.error());
                                else
                                {

console.dir(result.data());

if(result.more())

result.next();
                                }
                        }
                );
```

Что-бы получить свойства товара нужно в **select** указать `PROPERTY_*`

```
$arFields['select'] = array('*',
'PROPERTY_*');
```

crm.product.property.add

```
crm.product.property.add(fields)
```

Создаёт новое свойство товаров.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий описание свойства товаров.

Пример 1

```
                // Создание свойства
пользовательского типа S:HTML
        function
addPropertyExample(catalogId)
        {
                BX24.callMethod(

"crm.product.property.add",
                                {
```

```
fields: {

"ACTIVE": "Y",

"IBLOCK_ID": catalogId,

"NAME": "Свойство - HTML/текст",

"SORT": 500,

"DEFAULT_VALUE": {

"TYPE": "html",

"TEXT": "<u><b>Вкусные</b> \"<span
style=\"color: #00a650;\">африканские
бананы</span>\"</u>"

},

"USER_TYPE_SETTINGS": {

"HEIGHT": 300

},

"USER_TYPE": "HTML",

"PROPERTY_TYPE": "S"
                                        }
                                    },

function(result)
                                    {

if(result.error())
```

```
console.error(result.error());
                                        else

console.dir(result.data());
                                }
                );
        }

        function getCatalogId()
        {
                BX24.callMethod(

"crm.catalog.list",
                                {filter:
{"ORIGINATOR_ID": "", "ORIGIN_ID": ""}},

function(result)
                                {

if(result.error())
                                        {

console.error(result.error());
                                        }
                                        else
                                        {

var catalogId = 0;

if (result.total() !== 1)

{

catalogId = 0

}
```

```
else

{

data = result.data();

if (data && data instanceof Array &&
typeof(data[0]) === "object" && data[0]
["ID"])

{

catalogId = parseInt(data[0]["ID"]);

}

}

if (catalogId <= 0)

{

console.error("Не удалось получить
идентификатор товарного каталога CRM");

}

else

{

addPropertyExample(catalogId);

}
                                                }
                                        }
```

```
                );
        }

        function start()
        {
                getCatalogId();
        }

        start();
```

Пример 2

```
function addPropertyExample(catalogId)
        {
                BX24.callMethod(

"crm.product.property.add",
                                {

fields: {

"ACTIVE": "Y",

"IBLOCK_ID": catalogId,

"NAME": "Свойство - Список",

"SORT": 510,

"MULTIPLE": "Y",

"ROW_COUNT": 7,
```

```
"LIST_TYPE": "L",
"PROPERTY_TYPE": "L",
"VALUES": {
"n0": {
"ID": "n0",
"VALUE": "Значение списка 1",
"SORT": 100,
"DEF": "Y"
},
"n1": {
"ID": "n1",
"VALUE": "Значение списка 2",
"SORT": 200,
"DEF": "N"
},
"n2": {
"ID": "n2",
"VALUE": "Значение списка 3",
```

"SORT": 300,

"DEF": "Y"

},

"n3": {

"ID": "n3",

"VALUE": "Значение списка 4",

"SORT": 400,

"DEF": "N"

},

"n4": {

"ID": "n4",

"VALUE": "Значение списка 5",

"SORT": 500,

"DEF": "N"

},

"n5": {

"ID": "n5",

"VALUE": "Значение списка 6",

"SORT": 600,

"DEF": "N"

},

"n6": {

"ID": "n6",

"VALUE": "Значение списка 7",

"SORT": 700,

"DEF": "N"

},

"n7": {

"ID": "n7",

"VALUE": "Значение списка 8",

"SORT": 800,

"DEF": "N"

}

}

}

},

function(result)

{

if(result.error())

```
console.error(result.error());
                                        else

console.dir(result.data());
                                }
                );
        }

        function getCatalogId()
        {
                BX24.callMethod(

"crm.catalog.list",
                                {filter:
{"ORIGINATOR_ID": "", "ORIGIN_ID": ""}},

function(result)
                                {

if(result.error())
                                        {

console.error(result.error());
                                        }
                                        else
                                        {

var catalogId = 0;

if (result.total() !== 1)

{

catalogId = 0

}
```

```
else

{

data = result.data();

if (data && data instanceof Array &&
typeof(data[0]) === "object" && data[0]
["ID"])

{

catalogId = parseInt(data[0]["ID"]);

}

}

if (catalogId <= 0)

{

console.error("Не удалось получить
идентификатор товарного каталога CRM");

}

else

{

addPropertyExample(catalogId);

}
                                                }
                                        }
```

```
                );
        }

        function start()
        {
                getCatalogId();
        }

        start();
```

crm.product.property.delete

```
crm.product.property.delete(id)
```

Удаляет свойство товаров.

Параметры

Параметр	Описание
id	Идентификатор свойства товаров.

Пример

```
        var id = prompt("Введите ID");
        BX24.callMethod(

"crm.product.property.delete",
                        {
                                id: id
                        },
                        function(result)
                        {

if(result.error())
```

```
console.error(result.error());
                                else

console.info(result.data());
                            }
        );
```

crm.product.property.enumera

```
crm.product.property.enumeration.fields()
```

Возвращает описание полей элемента свойства товаров списочного типа.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(

"crm.product.property.enumeration.fields",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
```

```
console.dir(result.data());
                            }
        );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.product.property.fields

```
crm.product.property.fields()
```

Возвращает описание полей для свойств товаров.

Для полного понимания назначения полей свойств товаров рекомендуем почитать про свойства элементов инфоблока в документации для разработчиков.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(

"crm.product.property.fields",
                        {},
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
```

```
console.dir(result.data());
                            }
        );
```

crm.product.property.get

```
crm.product.property.get(id)
```

Возвращает свойство товаров по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор свойства товаров.

Пример

```
var id = prompt("Введите ID");
        BX24.callMethod(

"crm.product.property.get",
                        {
                                id: id
                        },
                        function(result)
                        {

if(result.error())
```

```
console.error(result.error());
                                else

console.dir(result.data());
                          }
        );
```

crm.product.property.list

```
crm.product.property.list(order, filter)
```

Возвращает список свойств товаров.

Параметры

Параметр	Описание
order	Поля сортировки.
filter	Поля фильтра.

Пример

```
BX24.callMethod(
                "crm.product.property.list",
{
                        order: {"SORT":
"ASC"},
                        filter: {
"PROPERTY_TYPE": "S",
                                "USER_TYPE":
```

```
"HTML"
                        }
                },
                function (result)
                {
                        if (result.error())
                        {

console.error(result.error());
                        }
                        else
                        {

console.dir(result.data());
                                if
(result.more())

result.next();
                        }
                }
        );
```

crm.product.property.settings.f

```
crm.product.property.settings.fields(propertyType, userType)
```

Возвращает описание полей дополнительных настроек свойства товаров пользовательского типа.

Параметры

Параметр	Описание
propertyType	Тип свойства.
userType	Пользовательский тип свойства.

Пример

```
        BX24.callMethod(
"crm.product.property.settings.fields",
                        {propertyType: "S",
userType: "HTML"},
                        function(result)
```

```
                                {

if(result.error())

console.error(result.error());
                                        else

console.dir(result.data());
                                }
        );
```

crm.product.property.types

```
crm.product.property.types()
```

Возвращает список типов свойств товаров.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(
"crm.product.property.types",
                        {},
                        function(result)
                        {
if(result.error())
console.error(result.error());
                                else
console.dir(result.data());
                        }
```

```
            );
```

crm.product.property.update

```
crm.product.property.update(id, fields)
```

Обновляет существующее свойство товаров.

Параметры

Параметр	Описание
id	Идентификатор свойства товаров.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.product.property.fields.

Пример

```
        var id = prompt("Введите ID");
        var propertyName = prompt("Введите
новое название");
        BX24.callMethod(

"crm.product.property.update",
```

```
                            {
                                    id: id,
                                    fields:
                                    {

"NAME": propertyName
                                    }
                            },
                            function(result)
                            {

if(result.error())

console.error(result.error());
                                    else
                                    {

console.dir(result.data());

if(result.more())

result.next();
                                    }
                            }
        );
```

crm.product.update

```
crm.product.update(id, fields)
```

Обновляет существующий товар.

Параметры

Параметр	Описание
id	Идентификатор товара.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.product.fields. Необходимо обязательно указать CURRENCY_ID для установки цены. **Примечание**: чтобы узнать требуемый формат полей, выполните метод crm.product.fields и посмотрите формат пришедших значений этих полей.

Для удаления файла в поле **valueId** указывается идентификатор значения свойства, а не идентификатор файла.

Пример

```
        var id = prompt("Введите ID");
                BX24.callMethod(

"crm.product.update",
                        {
                                id: id,
                                fields:
                                {

"CURRENCY_ID": "RUB",

"PRICE": 5000
                                }
                        },
                        function(result)
                        {

if(result.error())

console.error(result.error());
                                else
                                {

console.info(result.data());
                                }
                        }
                );
```

Добавление файлов в CRM указанным методом имеет свои особенности.

В этом вызове удаляется значение свойства с идентификатором 124 и добавляется новое значение с файлом 1.jpg.

```
BX24.callMethod(
    "crm.product.update",
    {
        id: 4611,
        fields:
            {
                "PROPERTY_186": [
                    {
                        "valueId": 0,
                        "fileData":
["1.jpg",
"/9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAIBAQIBAQI
CAgICAgICAwUDAwMDAwYEBAMFBwYH"

+"BwcGBwcICQsJCAgKCAcHCg0KCgsMDAwMBwkODw0MDg
sMDAz/2wBDAQICAgMDAwYDAwYMCAcIDAwMDAwMDAwMDA
wMDAwMD"

+"AwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAw
MDAz/wAARCAARABEDASIAAhEBAxEB/8QAHwAAAQUBAQE
BAQEAAA"

+"AAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUF
BAQAAAF9AQIDAAQRBRIhMUEGE1FhByJxFDKBkaEII0Kx
wRVS0fA"

+"kM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdIS
UpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJi
pKTlJWW"

+"l5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJyt
LT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAHw
EAAwEBA"
```

```
+"QEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8QAtREAAgE
CBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoE
IFEKRob"

+"HBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpKjU2Nzg5
OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6
goOEhYa"

+"HiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6w
sPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+
Pn6/9oA"

+"DAMBAAIRAxEAPwDqvg78Hf8Ahawjt7eO+n1Ce5NvDD
bso3YVWycg4xkkkkAAZOACa0vjF+z3J8ILO6j1CHULXU
LdI5Fjl"

+"kR0dWYDIKjDDkjIPUEdQRR+z38YofhBcQ6hHdR2+oW
t20sayQtIrqyBCDgdCNw4IPPBBwa2P2hv2ho/jdbXV1d
XVu140U"

+"UEMMFu8caIrhsDcM9SzZYk5PpgD+OcViuKVxTGlSi/
qN1d2le/MtFpbl5d366q2vDk2TeGUvDJ4jEPBfXvqVaX
vVoLEfW"

+"FCfIlDnvzX5bLlu38k/F6KKK/Xj+EQoooooAKKKKAP/
9k="]
                    },
                    {
                        "valueId": 124,
                        "value": {"remove":
"Y"}
                    }
                ]
            }
    },
    function (result) {
        if (result.error())
```

```
            console.error(result.error());
        else
        {
            console.info(result.data());
        }
    }
);
```

onCrmProductAdd

Событие, вызываемое при создании товара.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора созданного товара.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmProductDelete

Событие, вызываемое при удалении товара.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора удалённого товара.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmProductPropertyAdd

Событие, вызываемое при добавлении свойства товара.

Параметры события

Параметр	Описание	С версии
FIELDS	Массив содержит поле ID со значением идентификатора созданного свойства.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmProductPropertyDelete

Событие, вызываемое при удалении свойства товара.

Параметры события

Параметр	Описание	С версии
FIELDS	Массив содержит поле ID со значением идентификатора удалённого свойства.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmProductPropertyUpdate

Событие, вызываемое при изменении свойства товара.

Параметры события

Параметр	Описание	С версии
FIELDS	Массив содержит поле ID со значением идентификатора изменённого свойства.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

onCrmProductUpdate

Событие, вызываемое при обновлении товара.

Параметры события:

Параметр	Описание
FIELDS	Массив содержит поле ID со значением идентификатора обновлённого товара.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.duplicate.findbycomm

```
crm.duplicate.findbycomm()
```

Возвращает идентификаторы лидов, контактов и компаний содержащих телефоны или email-адреса из заданного списка.

Параметры

Параметр	Описание
type	Тип коммуникации: ▪ **EMAIL** - email-адрес; ▪ **PHONE** - телефон. Обязательный параметр.
values	Массив email или телефонов (до [dw]20 значений[/dw][di]Ограничено с целью снижения нагрузки.[/di]). Обязательный параметр.
entity_type	Необязательный параметр. Может быть опущен, в этом случае вернутся все три типа сущности. Если параметр используется, то можно оперировать только с одним из них. Если же задать массив или несуществующий

	параметр, то вернутся все типы. Типы сущности: ▪ **LEAD** - лид; ▪ **CONTACT** - контакт; ▪ **COMPANY** - компания.

Результат возвращается в виде объекта, содержащего массивы идентификаторов лидов, контактов и компаний.

Доступ к массиву идентификаторов производится по имени типа. Пример:

```
 { 'LEAD': [1, 2, 3], 'CONTACT': [4, 5, 6],
'COMPANY': [7, 8, 9] }
```

Пример поиска контакта по телефону:

```
//Поиск контакта по телефону
        BX24.callMethod(
            "crm.duplicate.findbycomm",
            {
                entity_type: "CONTACT",
                type: "PHONE",
                values: [ "8976543",
"11223355" ],
            },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else
                {
```

```
console.dir(result.data());
                    }
                }
            );
```

crm.entity.mergeBatch

Объединение дубликатов

```
crm.entity.mergeBatch({params:
{entityTypeId: number, entityIds:
number[]}})
```

`params` - ассоциативный массив, содержащий ключи:

Ключ	Описание
entityTypeId	идентификатор типа сущности. Может принимать значения `1` (лид), `2` (сделка), `3` (контакт) или `4` (компания)
entityIds	массив идентификаторов элементов, которые необходимо объединить

Метод вернет ассоциативный массив вида:

```
{
  "STATUS": "SUCCESS",
  "ENTITY_IDS": [
    "1"
```

```
    ]
  }
```

Здесь:

Параметр	Описание
STATUS	может принимать значения: ▪ `SUCCESS` - объединение прошло успешно. ▪ `CONFLICT` - при объединении возник конфликт. ▪ `ERROR` - при объединении произошла ошибка. Например, если у текущего пользователя нет прав на изменение или удаление записей.
ENTITY_IDS	содержит идентификаторы элементов, которые были объединены, кроме идентификатора элемента, оставшегося после объединения.

Пример вызова:

```
BX24.callMethod(
        'crm.entity.mergeBatch',
        {
                params: {
                        entityTypeId: 3,
                        entityIds: [1, 2,
3],
                }
        },
        (result) => {
```

```
                    console.log(result);
            }
);
```

Ручное объединение для случаев, когда возник конфликт

Для продолжения объединения вручную, в привычном пользователю интерфейсе **Битиркс24**, достаточно перенаправить его в раздел ручного объединения по соответствующей ссылке:

- Контакты: `/crm/contact/merge/?id=1,2,3`
- Компании: `/crm/company/merge/?id=1,2,3`
- Лиды: `/crm/lead/merge/?id=1,2,3`
- Сделки: `/crm/deal/merge/?id=1,2,3`

где параметр `id` содержит указанные через запятую идентификаторы объединяемых записей.

crm.duplicate.volatileType.field

Список полей, доступных для использования

```
crm.duplicate.volatileType.fields({?
entityTypeId: number})
```

Ключ	Описание
entityTypeId	Идентификатор типа сущности. Может принимать значения `1` (лид), `3` (контакт) или `4` (компания). Не обязательный параметр. Если не указан, будут возвращены доступные поля для всех сущностей.

Метод вернет массив, содержащий элементы-массивы с ключами:

Параметр	Описание
entityTypeId	Идентификатор типа сущности

fieldCode	Код поля
fieldTitle	Название поля

Например:

```
[
        {
                "entityTypeId": 1,
                "fieldCode": "TITLE",
                "fieldTitle": "Название
лида"
        },
        {
                "entityTypeId": 1,
                "fieldCode": "ADDRESS",
                "fieldTitle": "Адрес"
        },
        {
                "entityTypeId": 1,
                "fieldCode":
"SOURCE_DESCRIPTION",
                "fieldTitle": "Дополнительно
об источнике"
        },
        ...
]
```

crm.duplicate.volatileType.list

Список нестандартных полей, участвующих в поиске дубликатов

```
crm.duplicate.volatileType.list()
```

Метод вернет массив, содержащий элементы-массивы с ключами:

Параметр	Описание
id	Идентификатор типа дубликата
entityTypeId	Идентификатор типа сущности
fieldCode	Код поля

Например:

```
[
        {
```

```
                "id": 33554432,
                "entityTypeId": 1,
                "fieldCode": "TITLE"
        },
        {
                "id": 67108864,
                "entityTypeId": 3,
                "fieldCode":
"UF_CRM_1620140992555"
        }
]
```

crm.duplicate.volatileType.regi

Добавить поле в поиск дубликатов

```
crm.duplicate.volatileType.register({entityT
ypeId: number, fieldCode: string})
```

Ключ	Описание
entityTypeId	Идентификатор типа сущности. Может принимать значения `1` (лид), `3` (контакт) или `4` (компания).
fieldCode	Код поля, по которому необходимо включить поиск дубликатов.

Метод вернет массив, содержащий:

Параметр	Описание
id	Идентификатор добавленного типа дубликата.

	Обратите внимание, что этот идентификатор по сути является б**и**товой маской, поэтому принимает не стандартные автоинкрементные значения, а специфические.

crm.duplicate.volatileType.unre

Удалить поле из поиска дубликатов

```
crm.duplicate.volatileType.unregister({id:
number})
```

Для удаления поля из поиска дубликатов, нужно знать идентификатор типа, соответствующего этому полю. Узнать его можно на основе данных, возвращаемых методом crm.duplicate.volatileType.list.

Параметр	Описание
id	Идентификатор типа дубликата.

Метод вернет `true` в случае успешного удаления.

crm.multifield.fields

```
crm.multifield.fields()
```

Возвращает описание множественных полей. Множественные поля применяются для хранения телефонов, email-адресов и другой контактной информации. В лидах, контактах и компаниях полями этого типа являются PHONE, EMAIL, WEB и IM.

Параметры

Без параметров

Пример

```
BX24.callMethod(
"crm.multifield.fields",
{},
function(result)
{
if(result.error())
console.error(result.error());
```

```
                                    else

console.dir(result.data());
                            }
                  );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.enum.fields

```
crm.enum.fields()
```

Возвращает описание полей перечисления.

Параметры

Без параметров

Пример

```
        BX24.callMethod(
                "crm.enum.fields",
                {},
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
        );
```

crm.enum.activitydirection

```
crm.enum.activitydirection()
```

Возвращает элементы перечисления "Направление активности" (для писем и звонков).

Параметры

Без параметров

Пример

```
        BX24.callMethod(

"crm.enum.activitydirection",
                {},
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else
```

```
console.dir(result.data());
                }
        );
```

crm.enum.activitynotifytype

```
crm.enum.activitynotifytype()
```

Возвращает элементы перечисления "Тип уведомления о начале активности" (для встреч и звонков).

Параметры

Без параметров

Пример

```
        BX24.callMethod(

"crm.enum.activitynotifytype",
                {},
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else
```

```
console.dir(result.data());
                }
        );
```

crm.enum.activitypriority

```
crm.enum.activitypriority()
```

Возвращает элементы перечисления "Приоритет активности".

Параметры

Без параметров

Пример

```
        BX24.callMethod(
                "crm.enum.activitypriority",
                {},
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.dir(result.data());
```

```
                        }
            );
```

© «Битрикс», 2001-2008, «1С-

crm.enum.activitystatus

```
crm.enum.activitystatus()
```

Возвращает элементы перечисления "Статус" (STATUS).

Параметры

Без параметров

Пример

```
BX24.callMethod(
        "crm.enum.activitystatus",
       {},
       function(result)
       {
           if(result.error())

console.error(result.error());
           else
               console.dir(result.data());
       }
   );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.enum.activitytype

```
crm.enum.activitytype()
```

Возвращает элементы перечисления "Тип активности".

Параметры

Без параметров

Пример

```
        BX24.callMethod(
                "crm.enum.activitytype",
                {},
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.dir(result.data());
```

```
                        }
            );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.enum.addresstype

```
crm.enum.addresstype()
```

Возвращает элементы перечисления "Тип адреса".

Возможные значения


```
{
    "result": [
        {
            "ID": 1,
            "NAME": "Фактический адрес"
        },
        {
            "ID": 4,
            "NAME": "Адрес регистрации"
        },
        {
            "ID": 6,
            "NAME": "Юридический адрес"
        },
        {
            "ID": 9,
            "NAME": "Адрес бенефициара"
```

```
        }
    ],
    "time": {
        "start": 1561544164.224608,
        "finish": 1561544164.245065,
        "duration": 0.020457029342651367,
        "processing": 0.008939027786254883,
        "date_start": "2019-06-
26T13:16:04+03:00",
        "date_finish": "2019-06-
26T13:16:04+03:00"
    }
}
```


Параметры

Без параметров.

Пример

```
    BX24.callMethod(
        "crm.enum.addresstype",
        {},
        function(result)
        {
            if(result.error())

console.error(result.error());
            else
                console.dir(result.data());
        }
    );
```

crm.enum.contenttype

```
crm.enum.contenttype()
```

Возвращает элементы перечисления "Тип содержания".

Параметры

Без параметров

Пример

```
        BX24.callMethod(
                "crm.enum.contenttype",
                {},
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.dir(result.data());
```

```
                        }
            );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.enum.ownertype

Описание и пример

```
crm.enum.ownertype()
```

Метод возвращает идентификаторы типов сущностей CRM.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(
                "crm.enum.ownertype",
                {},
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else
```

```
console.dir(result.data());
                }
        );
```

Возможные значения


```
{
    "result": [
        {
            "ID": 1,
            "NAME": "Лид"
        },
        {
            "ID": 2,
            "NAME": "Сделка"
        },
        {
            "ID": 3,
            "NAME": "Контакт"
        },
        {
            "ID": 4,
            "NAME": "Компания"
        },
        {
            "ID": 7,
            "NAME": "Предложение"
        },
        {
            "ID": 5,
            "NAME": "Счёт"
        },
        {
            "ID": 8,
            "NAME": "Реквизиты"
```

```
        }
    ],
    "time": {
        "start": 1561543765.254939,
        "finish": 1561543765.270899,
        "duration": 0.015959978103637695,
        "processing": 0.003108978271484375,
        "date_start": "2019-06-
26T13:09:25+03:00",
        "date_finish": "2019-06-
26T13:09:25+03:00"
    }
}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.enum.settings.mode

```
crm.enum.settings.mode()
```

Возвращает описание режимов работы CRM.

Всегда возвращает массив: `[ { ID: 1, NAME: "Классическая CRM" }, { ID: 2, NAME: "Простая CRM" } ]`.

То есть метод crm.settings.mode.get возвращает значение, определённое в `crm.enum.settings.mode`.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.status.fields

```
crm.status.fields()
```

Возвращает описание полей справочника.

Параметры

Без параметров

Пример

```
        BX24.callMethod(
                "crm.status.fields",
                {},
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
        );
```

crm.status.entity.types

```
crm.status.entity.types()
```

Возвращает описание типов справочников. Результат - массив вида `array(array("ID"=>"символьный идентификатор справочника", "NAME":"название справочника")[, ...])`.

Параметры

Без параметров

Пример

```
        BX24.callMethod(
                "crm.status.entity.types",
                {},
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.dir(result.data());
```

```
                        }
            );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.status.entity.items

```
crm.status.entity.items()
```

Возвращает элементы справочника по его символьному идентификатору, упорядоченные по полю "SORT". Метод аналогичен crm.status.list за исключением того, что в последнем можно определить правила сортировки.

Параметры

Параметр	Описание
entityId	Символьный идентификатор справочника (может быть получен вызовом метода crm.status.entity.types). Обязательный

Пример

```
var id = prompt('Введите ID');
BX24.callMethod(
        "crm.status.entity.items",
        {
                entityId: id
        },
```

```
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
        );
```

crm.status.add

```
crm.status.add(fields)
```

Создаёт новый элемент в указанном справочнике.

Если добавляется стадия для пользовательского направления сделок, то к идентификатору статуса будет автоматически добавлен префикс направления. Это нужно, чтобы определить направление по идентификатору стадии.

Параметры

Параметр	Описание
fields	Набор полей - массив вида `array("поле"=>"значение"[, ...])`, содержащий значения полей справочника. (Обязательный)

Внимание! С версии модуля CRM 20.5.500 вводится ограничение на длину и формат значения поля STATUS_ID для некоторых ENTITY_ID:

- STATUS (статус лида). Макс. длина: 21, может содержать только латинские буквы, цифры, знаки тире и подчеркивания.

- QUOTE_STATUS (статус счета). Макс. длина: 22, может содержать только латинские буквы, цифры, знаки тире и подчеркивания.
- DEAL_STAGE (статус сделки). Макс. длина: 22, может содержать только латинские буквы, цифры, знаки тире и подчеркивания.
- DEAL_STAGE_xx (статус сделки в направлениях не по-умолчанию. xx - идентификатор направления). Макс. длина: 19 если xx меньше 10; Макс. длина: 18 если xx меньше 100 и т.д. Может содержать только латинские буквы, цифры, знаки тире и подчеркивания
- Для остальных ENTITY_ID, максимальная длина STATUS_ID - 50 символов, содержать может любые символы.

Пример

```
        BX24.callMethod(
                "crm.status.add",
                {
                        fields:
                        {
                                "ENTITY_ID":
"DEAL_STAGE",
                                "STATUS_ID":
"DECISION",
                                "NAME":
"Принятие решения",
                                "SORT": 70
                        }
                },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else
```

```
console.info("Создан элемент справочника с
ID " + result.data());
                }
        );
```

```
BX24.callMethod(
        "crm.status.add",
        {
           fields:
           {
               "ENTITY_ID": "DEAL_STAGE_1",
               "STATUS_ID": "DECISION",
               "NAME": "Принятие решения",
               "SORT": 70
           }
       },
       function(result)
       {
           if(result.error())

console.error(result.error());
           else
               console.info("Создан элемент
справочника с ID " + result.data());
       }
   );
```

В втором примере поле STATUS_ID будет сохранено как `C1:DECISION`. То есть будет добавлен префикс "C1:", где 1 - идентификатор направления сделок, к которому принадлежит справочник DEAL_STAGE_1.

© «Битрикс», 2001-2008, «1С-

crm.status.delete

```
crm.status.delete(id, params)
```

Удаляет элемент справочника.

Параметры

Параметр	Описание
id	Идентификатор элемента справочника. (Обязательный)
params	Набор параметров. FORCED - флаг принудительного удаления системных элементов. По умолчанию - N. Если удаляемый элемент является системным, то он не будет удалён. Если будет передано значение Y, то этот элемент будет удалён в любом случае. Для удаления системного элемента используйте второй пример в описании.

Пример

Пример удаления простого элемента

```
        var id = prompt("Введите ID
пользовательского элемента");
        BX24.callMethod(
                "crm.status.delete",
                { id: id },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.info(result.data());
                }
        );
```

Пример удаления системного элемента

```
var id = prompt("Введите ID
пользовательского или системного элемента");
BX24.callMethod(
    "crm.status.delete",
    { id: id, params:{ FORCED: "Y" } },
    function(result)
    {
       if(result.error())
            console.error(result.error());
       else
            console.info(result.data());
    }
);
```

crm.status.get

```
crm.status.get(id)
```

Возвращает элемент справочника по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор элемента справочника. (Обязательный)

Пример

```
        var id = prompt("Введите ID");
        BX24.callMethod(
                "crm.status.get",
                { id: id },
                function(result)
                {
                        if(result.error())

console.error(result.error());
```

```
                                else

console.dir(result.data());
                    }
          );
```

crm.status.list

```
crm.status.list()
```

Возвращает список элементов справочника по фильтру. Является реализацией списочного метода для элементов справочников. Обратите внимание, что в данной реализации параметры "select" и "navigation" не поддерживаются.

Параметры

Смотри описание списочных методов.

Пример

```
        BX24.callMethod(
                "crm.status.list",
                {
                        order: { "SORT": "ASC" },
                        filter: { "ENTITY_ID": "STATUS" }
                },
                function(result)
                {
                        if(result.error())
```

```
console.error(result.error());
                        else
                        {

console.dir(result.data());

if(result.more())

result.next();
                        }
                }
        );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.status.update

```
crm.status.update(id, fields)
```

Обновляет существующий элемент справочника.

Параметры

Параметр	Описание
id	Идентификатор элемента справочника. (Обязательный)
fields	Набор полей - массив вида `array("обновляемое поле"=>"значение"[, ...])`, где "обновляемое поле" может принимать значения из возвращаемых методом crm.status.fields. (Обязательный)

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.status.update",
        {
```

```
                        id: id,
                        fields:
                        {
                                "SORT": 75
                        }
                },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else
                        {

console.info(result.data());
                        }
                }
        );
```

crm.vat.add

```
crm.vat.add(fields)
```

Создаёт новую ставку НДС.

Параметры

Параметр	Описание
fields	Набор полей - массив вида array("поле"=>"значение"[, ...]), содержащий значения ставки НДС.

Пример

```
        var current = new Date();
        var date2str = function(d)
        {
                return d.getFullYear() + '-'
+ paddatepart(1 + d.getMonth()) + '-' +
paddatepart(d.getDate()) + 'T' +
paddatepart(d.getHours())
                        + ':' +
```

```
paddatepart(d.getMinutes()) + ':' +
paddatepart(d.getSeconds()) + '+03:00';
        };
        var paddatepart = function(part)
        {
                return part >= 10 ?
part.toString() : '0' + part.toString();
        };
        BX24.callMethod(
                "crm.vat.add",
                {
                        "fields":
                        {

"TIMESTAMP_X": date2str(current),
                                "ACTIVE":
"Y",
                                "C_SORT":
110,
                                "NAME": "НДС
18%",
                                "RATE":
18.00
                        }
                },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.info("Создана новая ставка НДС с ID
" + result.data());
                }
        );
```

crm.vat.delete

```
crm.vat.delete(id)
```

Удаляет ставку НДС.

Параметры

Параметр	Описание
id	Идентификатор ставки НДС.

Пример

```
        var id = prompt("Введите ID");
        BX24.callMethod(
                "crm.vat.delete",
                { "id": id },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else
```

```
console.info(result.data());
                }
        );
```

crm.vat.fields

```
crm.vat.fields()
```

Возвращает описание полей ставки НДС.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(
                "crm.vat.fields",
                {},
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
        );
```

crm.vat.get

```
crm.vat.get(id)
```

Возвращает ставку НДС по идентификатору.

Параметры

Параметр	Описание
id	Идентификатор ставки НДС.

Пример

```
        var id = prompt("Введите ID");
        BX24.callMethod(
                "crm.vat.get",
                { "id": id },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else
```

```
console.dir(result.data());
                }
        );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

crm.vat.list

Возвращает список ставок НДС по фильтру. Является реализацией списочного метода для ставок НДС.

Параметры

См. описание списочных методов.

Пример

```
        BX24.callMethod(
                "crm.vat.list",
                {
                        "order": { "ID":
"ASC" },
                        "filter": {
"ACTIVE": "Y" },
                        "select": [ "ID",
"NAME", "RATE" ]
                },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else
                        {
```

```
console.dir(result.data());

if(result.more())

result.next();
                        }
                }
        );
```

crm.vat.update

```
crm.vat.update(id, fields)
```

Обновляет существующую ставку НДС.

Параметры

Параметр	Описание
id	Идентификатор ставки НДС.
fields	Набор полей - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.vat.fields.

Пример

```
var id = prompt("Введите ID");
BX24.callMethod(
        "crm.vat.update",
        {
                "id": id,
```

```
                        "fields":
                        {
                                "ACTIVE":
"N"
                        }
                },
                function(result)
                {
                        if(result.error())

console.error(result.error());
                        else
                        {

console.info(result.data());
                        }
                }
        );
```

crm.livefeedmessage.add

Описание

```
crm.livefeedmessage.add(fields)
```

Добавляет сообщение в ленту CRM.

Параметры

Параметр	Описание
POST_TITLE	Заголовок сообщения.
MESSAGE	Текст сообщения.
SPERM	Права на просмотр сообщения, пример: `SPERM": { "CRMCONTACT": ["CRMCONTACT3", "CRMCONTACT7"], // контакты CRM "CRMCOMPANY": ["CRMCOMPANY1", "CRMCOMPANY3"], // компании CRM "CRMDEAL": ["CRMDEAL3", "CRMDEAL5"], // сделки CRM "CRMLEAD": ["CRMLEAD9", "CRMLEAD11"], // лиды CRM "SG": ["SG5", "SG9"], // рабочие группы соцсети "U": ["U1", "U3"], // пользователи "DR": ["DR1", "DR7"], // подразделения с подотделами }`

ENTITYTYPEID	Тип сущности (число), в которой опубликовано сообщение: ▪ 1 - лид; ▪ 2 - сделка; ▪ 3 - контакт; ▪ 4 - компания.
ENTITYID	ID конкретного лида/сделки/контакта/компании, в которой опубликовано сообщение.
FILES	Файлы сообщения.

Примеры

```
BX24.callMethod(
   "crm.livefeedmessage.add",
   {
      fields:
      {
         "POST_TITLE": "Немного о сервисе",
         "MESSAGE": "Битрикс24 создан на
базе платформы Bitrix Framework.",
         "SPERM": {
            "CRMCONTACT": ["CRMCONTACT3",
"CRMCONTACT7"],
            "CRMCOMPANY": ["CRMCOMPANY1",
"CRMCOMPANY3"],
            "CRMDEAL": ["CRMDEAL3",
"CRMDEAL5"],
            "CRMLEAD": ["CRMLEAD9",
"CRMLEAD11"],
            "SG": ["SG5", "SG9"],
            "U": ["U1", "U3"],
```

```
            "DR": ["DR1", "DR7"],
         },
         "ENTITYTYPEID": 3,
         "ENTITYID": 3,
      }
   },
   function(result)
   {
      if(result.error())
         console.error(result.error());
      else
         console.info("Создано сообщение с
ID " + result.data());
   }
);
```

```
BX24.callMethod(
   "crm.livefeedmessage.add",
   {
      fields:
      {
         "POST_TITLE": "POST_TITLE",
         "MESSAGE": "MESSAGE",
         "SPERM": {
            "CRMLEAD": ["CRMLEAD9",
"CRMLEAD11"],
            "U": ["U1"],
         },
         "ENTITYTYPEID": 1,
         "ENTITYID": 56374,
         "FILES": [
    ["1.gif",
"R0lGODlhAQABAIAAAP///wAAACH5BAEAAAAALAAAAAA
BAAEAAAICRAEAOw=="],
```

```
     ["2.gif", "..."]
 ],
       }
    },
    function(result)
    {
       if(result.error())
          console.error(result.error());
       else
          console.info("Создано сообщение с
 ID " + result.data());
    }
 );
```

crm.settings.mode.get

```
crm.settings.mode.get()
```

Возвращает текущие настройки режима работы CRM:

1 - классический режим работы (с лидами).

2 - режим работы без лидов.

То есть метод `crm.settings.mode.get` возвращает значение, определённое в crm.enum.settings.mode.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

pull.application.config.get

Описание

Метод для получения информации о подключении к real-time серверам и организации мгновенных коммуникаций в рамках приложений.

Благодаря подключению к RT-серверам вы сможете:

- создать действительно интерактивное приложение,
- менять состояния,
- мгновенно обновлять интерфейс без необходимости обновления страницы в режиме реального времени.

Обратите внимание: метод вернет данные о подключении к каналам, созданных специально для вашего rest-приложения. В рамках этих каналов вы будет получать только ваши события.

Параметры

Параметр	Пример	Обязательный	Описание
CACHE	Y / N	Нет	Возвращать кешированные данные или нет, по умолчанию Y.

Примеры

JavaScript

```
BX24.callMethod('pull.application.config.get
', {
        'CACHE': 'Y',
}, function(result){
        if(result.error())
        {

console.error(result.error().ex);
        }
        else
        {
                console.log(result.data());
        }
});
```

PHP

```
$result =
restCommand('pull.application.config.get', [
        'CACHE': 'Y',
], $_REQUEST["auth"]);
```

Пример ответа

```
{
    "result": {
        "server": {
            "version": 4,
            "server_enabled": true,
            "long_polling":
"http://rt.bitrix24.com/sub/",
```

```
            "long_polling_secure":
"https://rt.bitrix24.com/sub/",
            "websocket_enabled": true,
            "websocket":
"ws://rt.bitrix24.com/sub/",
            "websocket_secure":
"wss://rt.bitrix24.com/sub/"
            "publish_enabled": true,
                        "publish":
"http://rt.bitrix24.com/pubweb/",
                        "publish_secure":
"https://rt.bitrix24.com/pubweb/"
        },
        "channels": {
            "shared": {
                "id":
"46a437d2336d4a88e4e9b3cd956ecf45.7910bb25e6
60bf211fdec15e33c5e25e4c3b644a",
                "start": "2017-06-
28T12:04:00+02:00",
                "end": "2017-06-
29T00:04:00+02:00",
                "type": "shared"
            },
            "private": {
                "id":
"925153cd80b6b5a4dbf8659d5be21d1:abe9e696453
2000ab8b7acf092ba627b.605ea91793ad24be3f9745
d662713b23a5803a94",
                "public_id":
"abe9e6964532000ab8b7acf092ba627b.057ac8625a
e4ac0da4ed093a19950f9dab7e29d0",
                "start": "2017-06-
28T09:57:48+02:00",
                "end": "2017-06-
28T21:57:48+02:00",
                "type": "private"
```

```
                    }
                }
            }
        }
```

Объект **server** описывает конфигурацию сервера и пути для подключения к real-time каналу. Ключи объекта:

- **version** - версия установленного сервера,
- **server_enabled** - активирована или нет работа с сервером,
- **websocket_enabled** - доступна или нет работа с веб сокетами.
- **long_pooling** и **websocket** - пути подключения,
- **long_pooling_secure** и **websocket_secure** - пути подключения при использовании протокола https,
- **publish_enabled** - доступна или нет [dw]возможность публикации сообщения[/dw][di]Доступно начиная с 4-й версии сервера очередей.[/di] со стороны клиента,
- **publish** и **publish_secure** - пути для публикации сообщений со стороны клиента.
- **clientId** - уникальный идентификатор портала на облачном push-сервере. Возвращается в случае, если на портале используется [ds]облачный push-сервер[/ds][di]Push-уведомления – это небольшие всплывающие окна, которые появляются на экране мобильного телефона или обычного компьютера и сообщают о важных событиях и обновлениях. То есть инициатором уведомлений являются сайты, на которые подписан пользователь. Противоположностью Push-технологии является технология Pull, где информация запрашивается самим пользователем.

  Подробнее...[/di].

Объект **channels** описывает данные для подключения пользователя к каналам. Ключи:

- **shared** - общий канал портала. На этом канале публикуются команды для всех пользователей портала (в том числе пользователей экстранет).
- **private** - приватный канал пользователя. На этом канале публикуются команды только для текущего пользователя.

Массив канала, содержит:

- **id** - идентификатор канала;
- **public_id** - публичный [dw]идентификатор канала[/dw] [di]Доступен только для 4-й версии сервера очередей и только для приватных каналов[/di];
- **start** - время создания канала (в формате ATOM);
- **end** - время окончания работы канала (в формате ATOM);
- **type** - тип канала.

Пример ответа при возникновения ошибки

```
{
    "error": "SERVER_ERROR",
    "error_description": "Push & Pull server
is not configured"
}
```

Ключи:

- **error** - код возникшей ошибки
- **error_description** - краткое описание возникшей ошибки

Возможные коды ошибок

Код	Описание
SERVER_ERROR	На портале не настроен модуль **Push & Pull** на работу с сервером очередей.
WRONG_AUTH_TYPE	Метод можно использовать только в рамках OAuth 2.0 или через веб-хуки .

Смотрите также

- Интерактивность в приложениях

pull.application.event.add

Описание

Метод для отправки событий в RT-канал приложения.

Параметры

Параметр	Пример	Обязательный	Описание
COMMAND	'test'	Да	Тип события, строка.
PARAMS	{JSON Object}	Нет	Произвольный JSON массив с данными.
MODULE_ID	'application'	Нет	Если отправляются команды из разных подсистем приложения, можно это указать через модуль.
USER_ID	1 или [1,2,3]	Нет	Если не указывать USER_ID, то данные будут отправлены в общий канал. Если указать ID пользователя,

			то данные будут отправлены в приватный канал. Администратор может отправлять одновременно нескольким пользователям и в общий канал, пользователь без прав - только себе или в общий канал.

Примеры

JavaScript

```
BX24.callMethod('pull.application.event.add'
, {
        'COMMAND': 'test',
        'PARAMS': '{"param1":"value1"}',
}, function(result){
        if(result.error())
        {

console.error(result.error().ex);
        }
        else
        {
                console.log(result.data());
        }
});
```

PHP

```
$result =
restCommand('pull.application.event.add', [
        'COMMAND': 'test',
        'PARAMS': ['param1' => 'value1'],
], $_REQUEST["auth"]);
```

Пример ответа

```
{
    "result": true
}
```

Пример ответа при возникновения ошибки

```
{
    "error": "WRONG_AUTH_TYPE",
    "error_description": "Get access to
application config available only for
application authorization."
}
```

Ключи:

- **error** - код возникшей ошибки
- **error_description** - краткое описание возникшей ошибки

Возможные коды ошибок

Код	Описание
COMMAND_ERROR	Формат поля MODULE_ID не верный. Разрешены английские буквы в смешанном регистре, цифры, символ подчеркивания, точка и тире.
MODULE_ID_ERROR	Формат поля MODULE_ID не верный. Разрешены английские буквы в нижнем регистре, цифры, точка и знак подчеркивания.
USER_ID_ACCESS_ERROR	Указывать произвольных пользователей может только пользователь с правами администратора.
PARAMS_ERROR	Передан не корректный JSON обьект.
WRONG_AUTH_TYPE	Метод можно использовать только в рамках OAuth 2.0.

Смотрите также

- Интерактивность в приложениях

pull.application.push.add

Описание

Метод для отправки push-уведомления на мобильное устройство в рамках приложения Битрикс24.

Параметры

Параметр	Пример	Обязат
USER_ID	1 или [1,2,3]	Да
TEXT	'Hello, world!'	Нет
AVATAR	'https://files.shelenkov.com/images/avatar-ivanov.jpg'	Нет

Примеры

JavaScript

```
BX24.callMethod('pull.application.push.add',
{
        'USER_ID': [1,2,3],
        'TEXT': 'Hello, world!',
```

```
        'AVATAR':
'https://files.shelenkov.com/images/avatar-
ivanov.jpg',
}, function(result){
        if(result.error())
        {

console.error(result.error().ex);
        }
        else
        {
                console.log(result.data());
        }
});
```

PHP

```
$result =
restCommand('pull.application.push.add', [
        'USER_ID': [1,2,3],
        'TEXT': 'Hello, world!',
        'AVATAR':
'https://files.shelenkov.com/images/avatar-
ivanov.jpg',
], $_REQUEST["auth"]);
```

Пример ответа

```
{
    "result": true
}
```

Пример ответа при возникновения ошибки

```
{
    "error": "WRONG_AUTH_TYPE",
    "error_description": "Send push
notifications available only for application
authorization."
}
```

Ключи:

- **error** - код возникшей ошибки
- **error_description** - краткое описание возникшей ошибки

Возможные коды ошибок

Код	Описание
TEXT_ERROR	Не передан текст сообщения.
EMPTY_APP_NAME	Ошибка возникает если у вашего приложения не задано название.
ACCESS_ERROR	Метод может использовать только пользователь с правами администратора.
WRONG_AUTH_TYPE	Метод можно использовать только в рамках OAuth 2.0.

Смотрите также

- Интерактивность в приложениях

Частые кейсы::Добавить активити, создающую счет на основании лида или сделки

Данный пример показывает как можно сделать свою активити в битрикс24 создающую счет на основании лида или сделки

Внимание! Для использования данного примера необходимо настроить работу класса CRest и подключить файл crest.php в файлах, где используется данный класс подробнее.

файл регистрирующий активити необходимо изменить путь **$handlerUrl** на ваш путь до файла выполняющего активити :

```
<?
$handlerUrl = 'https://yourdomain.yyy/handler.php';
$result = CRest::call(
        'bizproc.activity.add',
        [
                'CODE' => 'activityAccount',
                'HANDLER' => $handlerUrl,
                'AUTH_USER_ID' => 1,
                'NAME' => 'ActivityAccount',
                'DESCRIPTION' => 'description',
                'PROPERTIES' => [
                        'account_title' => [
                                'Name' => 'Format account
title',
                                'Description' => '',
                                'Type' => 'string',
                                'Required' => 'Y',
                                'Multiple' => 'N',
                                'Default' => 'Account
title',
```

```
                        ],

                        'my_company_id' => [
                                'Name' => 'My Company id',
                                'Description' => '',
                                'Type' => 'int',
                                'Required' => 'Y',
                                'Multiple' => 'N',
                                'Default' => '1',
                        ],

                        'pay_system_id' => [
                                'Name' => 'Pay system id',
                                'Description' => '',
                                'Type' => 'int',
                                'Required' => 'Y',
                                'Multiple' => 'N',
                                'Default' => '1',
                        ],
                ]
        ]
);
?>
```

обработчик активити путь до которого вы указали в переменной **$handlerUrl** выше:

```
<?
$my_company_id = intVal($_REQUEST['properties']
['my_company_id']);
$pay_system_id = intVal($_REQUEST['properties']
['pay_system_id']);//some in
CRest::call('sale.paysystem.list')
$account_title = htmlspecialchars($_REQUEST['properties']
['account_title']);
$arDocument = $_REQUEST['document_id'];
$iDealID = 0;
$iLeadID = 0;
if (is_array($arDocument))
{
        foreach ($arDocument as $param)
        {//search id
                if (strpos($param, 'DEAL_') !== false)
                {
                        $iDealID = intVal(substr($param,
strlen('DEAL_')));
                        break;
                }
```

```
                elseif (strpos($param, 'LEAD_') !== false)
                {
                        $iLeadID = intVal(substr($param,
strlen('LEAD_')));
                        break;
                }
        }
}
if ($iDealID > 0)
{
        $result = CRest::call(
                'crm.deal.get',
                [
                        'id' => $iDealID
                ]
        );
        if (!empty($result['result']))
        {
                $arData = $result['result'];
                $resultProduct = CRest::call(
                        'crm.deal.productrows.get',
                        [
                                'id' => $iDealID
                        ]
                );
        }
}
elseif ($iLeadID > 0)
{
        $result = CRest::call(
                'crm.lead.get',
                [
                        'id' => $iLeadID
                ]
        );
        if (!empty($result['result']))
        {
                $arData = $result['result'];
                $resultProduct = CRest::call(
                        'crm.lead.productrows.get',
                        [
                                'id' => $iLeadID
                        ]
                );
        }
}
if (!empty($arData['COMPANY_ID']) ||
!empty($arData['CONTACT_ID']))
{
        if (empty($resultProduct['result']))
        {//if the deal or lead has no products
```

```
                $resultProduct['result'][] = [
                        'ID' => 0,
                        'PRODUCT_ID' => 0,
                        'PRODUCT_NAME' => $account_title,
                        'QUANTITY' => 1,
                        'PRICE' =>
($arData['OPPORTUNITY'])?:0,
                ];
        }
        $arProduct = [];
        foreach ($resultProduct['result'] as $product)
        {
                $arProduct[] = [
                        'ID' => $product['ID'],
                        'PRODUCT_ID' =>
$product['PRODUCT_ID'],
                        'PRODUCT_NAME' =>
$product['PRODUCT_NAME'],
                        'QUANTITY' => $product['QUANTITY'],
                        'PRICE' => $product['PRICE']
                ];
        }

        $resultInvoice = CRest::call(
                'crm.invoice.add',
                [
                        'fields' => [
                                'ORDER_TOPIC' =>
$account_title,
                                'UF_COMPANY_ID' =>
$arData['COMPANY_ID'],
                                'UF_CONTACT_ID' =>
$arData['CONTACT_ID'],
                                'UF_DEAL_ID' =>
$arData['ID'],
                                'UF_MYCOMPANY_ID' =>
$my_company_id,
                                'PERSON_TYPE_ID' =>
($arData['COMPANY_ID'] > 0) ? 1 : 2,//1 is company, 2 is
contact in CRest::call('crm.persontype.list')
                                'PAY_SYSTEM_ID' =>
$pay_system_id,
                                "STATUS_ID" => "N",
                                'DATE_INSERT' =>
date(DATE_ATOM),
                                'DATE_BILL' =>
date(DATE_ATOM),
                                'DATE_PAY_BEFORE' =>
date(DATE_ATOM, time() + 3600 * 24 * 20),//20 day pay
                                'PRODUCT_ROWS' =>
$arProduct,
```

```
                                ]
                        ]
            );
}

?>
```

Частые кейсы::Добавить робота crm, создающего счет на основании лида или сделки

Данный пример показывает как можно сделать робота в битрикс24 создающего счет на основании лида или сделки

Внимание! Для использования данного примера необходимо настроить работу класса CRest и подключить файл crest.php в файлах, где используется данный класс подробнее.

файл регистрирующий активити необходимо изменить путь **$handlerUrl** на ваш путь до файла выполняющего активити :

```
<?
$handlerUrl = 'https://yourdomain.yyy/handler.php';
$result = CRest::call(
        'bizproc.robot.add',
        [
                'CODE' => 'robotAccount',
                'HANDLER' => $handlerUrl,
                'AUTH_USER_ID' => 1,
                'NAME' => 'RobotAccount',
                'PROPERTIES' => [
                        'account_title' => [
                                'Name' => 'Format account
title',
                                'Description' => '',
                                'Type' => 'string',
                                'Required' => 'Y',
                                'Multiple' => 'N',
                                'Default' => 'Account
title',
                        ],
```

```
                        'my_company_id' => [
                                'Name' => 'My Company id',
                                'Description' => '',
                                'Type' => 'int',
                                'Required' => 'Y',
                                'Multiple' => 'N',
                                'Default' => '1',
                        ],

                        'pay_system_id' => [
                                'Name' => 'Pay system id',
                                'Description' => '',
                                'Type' => 'int',
                                'Required' => 'Y',
                                'Multiple' => 'N',
                                'Default' => '1',
                        ],
                ]
        ]
);
?>
```

обработчик активити путь до которого вы указали в переменной **$handlerUrl** выше:

```
<?

$my_company_id = intVal($_REQUEST['properties']
['my_company_id']);
$pay_system_id = intVal($_REQUEST['properties']
['pay_system_id']);//some in
CRest::call('sale.paysystem.list')
$account_title = htmlspecialchars($_REQUEST['properties']
['account_title']);
$arDocument = $_REQUEST['document_id'];
$iDealID = 0;
$iLeadID = 0;
if (is_array($arDocument))
{
        foreach ($arDocument as $param)
        {//search id
                if (strpos($param, 'DEAL_') !== false)
                {
                        $iDealID = intVal(substr($param,
strlen('DEAL_')));
                        break;
                }
```

```
                elseif(strpos($param, 'LEAD_') !== false)
                {
                        $iLeadID = intVal(substr($param,
strlen('LEAD_')));
                        break;
                }
        }
}

if ($iDealID > 0)
{
        $result = CRest::call(
                'crm.deal.get',
                [
                        'id' => $iDealID
                ]
        );
        if (!empty($result['result']))
        {
                $arData = $result['result'];
                $resultProduct = CRest::call(
                        'crm.deal.productrows.get',
                        [
                                'id' => $iDealID
                        ]
                );
        }
}
elseif($iLeadID > 0)
{
        $result = CRest::call(
                'crm.lead.get',
                [
                        'id' => $iLeadID
                ]
        );
        if (!empty($result['result']))
        {
                $arData = $result['result'];
                $resultProduct = CRest::call(
                        'crm.lead.productrows.get',
                        [
                                'id' => $iLeadID
                        ]
                );
        }
}

if(!empty($arData['COMPANY_ID']) ||
!empty($arData['CONTACT_ID']))
{
```

```
        if (empty($resultProduct['result']))
        {//if the deal or lead has no products
                $resultProduct['result'][] = [
                        'ID' => 0,
                        'PRODUCT_ID' => 0,
                        'PRODUCT_NAME' => $account_title,
                        'QUANTITY' => 1,
                        'PRICE' =>
($arData['OPPORTUNITY'])?:0,
                ];
        }
        $arProduct = [];
        foreach ($resultProduct['result'] as $product)
        {
                $arProduct[] = [
                        'ID' => $product['ID'],
                        'PRODUCT_ID' =>
$product['PRODUCT_ID'],
                        'PRODUCT_NAME' =>
$product['PRODUCT_NAME'],
                        'QUANTITY' => $product['QUANTITY'],
                        'PRICE' => $product['PRICE']
                ];
        }
        $resultInvoice = CRest::call(
                'crm.invoice.add',
                [
                        'fields' => [
                                'ORDER_TOPIC' =>
$account_title,
                                'UF_COMPANY_ID' =>
$arData['COMPANY_ID'],
                                'UF_CONTACT_ID' =>
$arData['CONTACT_ID'],
                                'UF_DEAL_ID' =>
$arData['ID'],
                                'UF_MYCOMPANY_ID' =>
$my_company_id,
                                'PERSON_TYPE_ID' =>
($arData['COMPANY_ID'] > 0) ? 1 : 2,//1 is company, 2 is
contact in CRest::call('crm.persontype.list')
                                'PAY_SYSTEM_ID' =>
$pay_system_id,
                                "STATUS_ID" => "N",
                                'DATE_INSERT' =>
date(DATE_ATOM),
                                'DATE_BILL' =>
date(DATE_ATOM),
                                'DATE_PAY_BEFORE' =>
date(DATE_ATOM, time() + 3600 * 24 * 20),//20 day pay
                                'PRODUCT_ROWS' =>
```

```
$arProduct,
                            ]
                    ]
            );
}
?>
```

Частые кейсы::Настройка REST роботов (действий) через приложение (встройку)

Описание

С версии **20.0.600** модуля **Бизнес-процессы** появилась возможность настраивать робота/действие, используя интерфейс приложения. Реализована стандартным механизмом встройки rest.placement.

Реализация на примере приложения

Рассмотрим реализацию на конкретном примере приложения (полный код приложения представлен в следующем табе страницы).

В нашем примере приложение добавляет робота, у которого есть 2 параметра типа **Строка**:

```
var params = {
   'CODE': 'robot',
   'HANDLER': 'http://handler.com',
```

```
    'AUTH_USER_ID': 1,
    'NAME': 'Пример робота-встройки',
    'USE_PLACEMENT': 'Y',
    'PLACEMENT_HANDLER':
 'http://handler.com',
    'PROPERTIES': {
       'string': {
          'Name': 'Параметр 1',
          'Type': 'string'
       },
       'stringm': {
          'Name': 'Параметр 2',
          'Type': 'string',
          'Multiple': 'Y',
          'Default': ['value 1', 'value 2']
       },
    }
 };

 BX24.callMethod(
    'bizproc.robot.add',
    params,
    function(result)
    {
       if(result.error())
          alert("Ошибка: " + result.error());
    }
 );
```

Для того, чтобы параметры можно было настраивать через приложение, при добавлении робота передаем параметры `USE_PLACEMENT=Y` и обработчик `PLACEMENT_HANDLER`.

Останется написать обработчик встройки, задача которого отрисовать параметры и сохранить их значения. Для этого в обработчик в `PLACEMENT_OPTIONS` мы передаем данные:

- **code** – код вашего робота при регистрации;

- **activity_name** – идентификатор действия в шаблоне бизнес-процесса;
- **properties** – список свойств и их описание;
- **current_values** – текущие значения свойств;
- **document_type** – тип документа, над которым идет настройка;
- **document_fields** – список полей документа.

В приложение это приходит так:

```
Array
(
    [code] => robot
    [activity_name] =>
A72788_31169_37133_27365
    [properties] => Array
        (
            [string] => Array
                (
                    [NAME] => Параметр 1
                    [TYPE] => string
                )

            [stringm] => Array
                (
                    [NAME] => Параметр 2
                    [TYPE] => string
                    [MULTIPLE] => Y
                    [DEFAULT] => Array
                        (
                            [0] => value 1
                            [1] => value 2
                        )

                )

        )
```

```
    [current_values] => Array
        (
            [string] => 1
            [stringm] => Array
                (
                    [0] => 2
                )

        )

    [document_type] => Array
        (
            [0] => crm
            [1] => CCrmDocumentDeal
            [2] => DEAL
        )

    [document_fields] => Array
        (
            [ID] => Array
                (
                    [Name] => ID
                    [Type] => int
                    [Filterable] => 1
                    [Editable] =>
                    [Required] =>
                    [BaseType] => int
                )

            [TITLE] => Array
                (
                    [Name] => Название
                    [Type] => string
                    [Filterable] => 1
                    [Editable] => 1
                    [Required] => 1
                    [BaseType] => string
```

```
                    )

                [...]
            )
)
```

Осталось сделать верстку и научиться сохранять параметры непосредственно в Робота. Для этого мы используем функцию `setPropertyValue`, которая доступна нам через BX24.placement.call.

```
BX24.placement.call(
   'setPropertyValue',
   {string: 'test string'}
)
```

В качестве параметров передаются ID свойства и значения. Можно передавать несколько значений свойств, например:

```
BX24.placement.call(
   'setPropertyValue',
   {string: 'test string', stringm:
['test2', 'test3']}
)
```

Далее пользователь сохраняет робота как обычно.

Так это выглядит в Роботах:

А так в дизайнере бизнес-процессов:

Примечание: Белая область - встройка (фрейм приложения).

Полный код приложения

```
<?php
header('Content-Type: text/html;
charset=UTF-8');

$protocol = $_SERVER['SERVER_PORT'] == '443'
? 'https' : 'http';
$host = explode(':', $_SERVER['HTTP_HOST']);
$host = $host[0];

define('BP_APP_HANDLER',
$protocol.'://'.$host.explode('?',
$_SERVER['REQUEST_URI'])[0]);
?>
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title></title>
</head>
<body>
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/boo
tstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQU
OhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
<script
src="https://stackpath.bootstrapcdn.com/boot
strap/4.3.1/js/bootstrap.min.js"
```

```
integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJG
zIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
<script src="//api.bitrix24.com/api/v1/">
</script>
<?if (!isset($_POST['PLACEMENT']) ||
$_POST['PLACEMENT'] === 'DEFAULT'):?>
<h1>Робот-встройка</h1>
<div class="container-fluid">
<div class="container-fluid">
                <h2>Робот</h2>
                <button
onclick="installRobot();" class="btn btn-
primary">Установить</button> вЂ“
                <button
onclick="uninstallRobot();" class="btn btn-
danger">Удалить</button>
        </div>
        <hr/>
        <div class="container-fluid">
                <button onclick="getList();"
class="btn btn-light">Получить список
установленных роботов</button>
        </div>
 </div>
 <script type="text/javascript">
        document.body.style.display =
'none';
        BX24.init(function()
        {
                document.body.style.display
= '';
        });

        function installRobot()
        {
```

```
                var params = {
                        'CODE': 'robot',
                        'HANDLER': '<?
=BP_APP_HANDLER?>',
                        'AUTH_USER_ID': 1,
                        'NAME': 'Пример
робота-встройки',
                        'USE_PLACEMENT':
'Y',
                        'PLACEMENT_HANDLER':
'<?=BP_APP_HANDLER?>',
                        'PROPERTIES': {
                                'string': {

'Name': 'Параметр 1',

'Type': 'string'
                                },
                                'stringm': {

'Name': 'Параметр 2',

'Type': 'string',

'Multiple': 'Y',

'Default': ['value 1', 'value 2']
                                },
                        }
                };

                BX24.callMethod(
                        'bizproc.robot.add',
                        params,
                        function(result)
                        {
```

```
if(result.error())

alert("Ошибка: " + result.error());
                                else

alert("Успешно");
                        }
                );
        }

        function uninstallRobot()
        {
                BX24.callMethod(

'bizproc.robot.delete',
                                {

'CODE': 'robot'
                                },

function(result)
                                {

if(result.error())

alert('Ошибка: ' + result.error());
                                        else

alert("Успешно");
                                }
                );
        }

        function getList()
        {

                BX24.callMethod(
```

```
'bizproc.robot.list',
                        {},
                        function(result)
                        {

if(result.error())

alert("Ошибка: " + result.error());
                                else

alert("Коды установленных роботов: " +
result.data().join(', '));
                        }
                );
        }

 </script>
 <?php else:?>
        <form name="props" class="container-
fluid">
 <?php
 $options =
json_decode($_POST['PLACEMENT_OPTIONS'],
true);

 foreach ($options['properties'] as $id =>
$property)
 {
        $multiple =
isset($property['MULTIPLE']) &&
$property['MULTIPLE'] === 'Y';
        $val = (array)
$options['current_values'][$id];
        if (!$val)
        {
                $val[] = '';
```

```
        }

        if ($multiple)
        {
                $val[] = '';
        }

        $name = $multiple ? $id.'[]' : $id;
        ?>
        <div class="form-group">
                <label><?
=htmlspecialchars($property['NAME'])?>:
</label>
                <?foreach ($val as $v):?>
                <p><input name="<?=$name?>"
value="<?=htmlspecialchars((string)$v)?>"
class="form-control"
onchange="setPropertyValue('<?=$id?>',
this.name, <?=(int)$multiple?>)"></p>
                <?endforeach;?>
        </div>
        <?
 }
 ?>
                <script>
                        function
setPropertyValue(name, inputName, multiple)
                        {
                                var form =
new FormData(document.forms.props);
                                var value =
multiple? form.getAll(inputName) :
form.get(inputName);
                                var params =
{};
                                params[name]
= value;
```

```
BX24.placement.call(

'setPropertyValue',

params
                                        )
                                  }
                        </script>
                </form>
 <?php endif;?>
 </body>
</html>
```

bizproc.workflow.instance.list

bizproc.workflow.instance.list - метод возвращает список запущенных бизнес-процессов. Алиас метода bizproc.workflow.instance, параметры и ограничения полностью совпадают.

Пример

```
BX24.callMethod(
   'bizproc.workflow.instance.list',
   {
      select: [
         'ID',
         'STARTED',
         'STARTED_BY',
         'TEMPLATE_ID',
         'DOCUMENT_ID'
      ],
      order: {MODIFIED: 'DESC'},
      filter: {MODULE_ID: 'crm', ENTITY:
'CCrmDocumentLead'}
   },
   function(result)
   {
      if(result.error())
         alert("Error: " + result.error());
      else
```

```
        {
            console.log(result.data());
        }
    }
)
```

bizproc.workflow.instances

Описание и пример

bizproc.workflow.instances - метод возвращает список запущенных бизнес-процессов.

Важно: метод доступен только администратору.

Пример

```
BX24.callMethod(
   'bizproc.workflow.instances',
   {
      select: ['ID', 'MODIFIED',
'OWNED_UNTIL', 'MODULE_ID', 'ENTITY',
'DOCUMENT_ID', 'STARTED', 'STARTED_BY',
'TEMPLATE_ID'],
      order: {STARTED: 'DESC'},
      filter: {'>STARTED_BY': 0}
   },
   function(result)
   {
      if(result.error())
         alert("Error: " + result.error());
      else
         console.log(result.data());
```

```
    }
);
```

Параметры

Параметр	Описание	Значение по умолчанию
SELECT	Массив полей записей, которые будут возвращены методом. Можно указать только те поля, которые необходимы. Доступные поля: `ID` - идентификатор бизнес-процесса; `OWNED_UNTIL` - время блокировки бизнес-процесса. Процесс считается зависшим, если разница времени блокировки с текущим временем более 5 минут; `MODULE_ID` - идентификатор модуля (по документу); `ENTITY` - идентификатор сущности (по документу); `DOCUMENT_ID` - идентификатор документа; `STARTED` - дата запуска бизнес-процесса;	`['ID', 'MODIFIED', 'OWNED_UNTIL']`

	STARTED_BY - кем запущен бизнес-процесс; TEMPLATE_ID - идентификатор шаблона бизнес-процесса.	
FILTER	Массив вида {"фильтруемое_поле": "значение фильтра" [, ...]}. Список фильтруемых полей такой же, как для параметра SELECT. Перед названием фильтруемого поля может указать тип фильтрации: ▪ "!" - не равно; ▪ "<" - меньше; ▪ "<=" - меньше либо равно; ▪ ">" - больше; ▪ ">=" - больше либо равно.	
ORDER	Массив для сортировки результата. Массив вида {"поле_сортировки": 'направление сортировки' [, ...]}. Список полей для сортировки такой же, как для параметра SELECT. Направление сортировки может принимать значения: ▪ asc - по возрастанию; ▪ desc - по убыванию.	{'MODIFIED': 'desc'}

bizproc.workflow.kill

bizproc.workflow.kill - метод удаляет запущенный бизнес-процесс.

Параметры

Параметр	Описание
ID	Идентификатор бизнес-процесса.

Пример

```
function killWf(id, cb)
{
   var params = {ID: id};

   BX24.callMethod(
      'bizproc.workflow.kill',
      params,
      function(result)
      {
         if(result.error())
            alert("Error: " +
result.error());
         else if (cb)
```

```
                cb();
            }
        );
    }
```

bizproc.workflow.start

bizproc.workflow.start - метод запускает Бизнес-процесс

Параметры

Параметр	Описание
TEMPLATE_ID	Идентификатор шаблона БП
DOCUMENT_ID	Идентификатор документа БП
PARAMETERS	Значения параметров БП (если шаблон с параметрами)

Примеры

```
function startWf(leadId, tplId, cb)
{
   if (!leadId)
   {
      alert('Lead not selected');
      return;
   }

   var params = {
      TEMPLATE_ID: tplId,
```

```
      DOCUMENT_ID: ['crm',
'CCrmDocumentLead', leadId],
      PARAMETERS: null
   };

   BX24.callMethod(
      'bizproc.workflow.start',
      params,
      function(result)
      {
         if(result.error())
            alert("Error: " +
result.error());
         else if (cb)
            cb();
      }
   );
}
```

Примеры подстановки в параметр DOCUMENT_ID:

```
['crm', 'CCrmDocumentLead', 'LEAD_777'] -
Лид
['crm', 'CCrmDocumentCompany',
'COMPANY_777'] - Компания
['crm', 'CCrmDocumentContact',
'CONTACT_777'] - Контакт
['crm', 'CCrmDocumentDeal', 'DEAL_777'] -
Сделка
['disk', 'Bitrix\Disk\BizProcDocument',
'777'] - файл Диска
['lists', 'BizprocDocument', '777'] -
документ Процессов в ленте (в новостях)
['lists',
'Bitrix\Lists\BizprocDocumentLists', '777']
- документ Списков
```

Пример DOCUMENT_ID для смарт-процесса:

```
DOCUMENT_ID = ['crm',
'Bitrix\Crm\Integration\BizProc\Document\Dyn
amic', 'DYNAMIC_147_1']
```

Где 147 - это ID смарт-процесса, 1 - ID элемента.

Пример подстановки в параметр DOCUMENT_ID для новых счетов:

```
Bitrix\Crm\Integration\BizProc\Document\Smar
tInvoice

SMART_INVOICE_<ID элемента>
```

bizproc.workflow.terminate

bizproc.workflow.terminate - метод останавливает активный Бизнес-процесс.

Параметры

Параметр	Описание
ID	Идентификатор БП, который нужно остановить
STATUS	Установить текст статуса. (не обязательный)

Пример

```
function terminateWf(id, cb)
{
   var params = {ID: id, STATUS: 'Terminated
by rest app.'};

   BX24.callMethod(
      'bizproc.workflow.terminate',
      params,
      function(result)
      {
```

```
            if(result.error())
               alert("Error: " +
result.error());
            else if (cb)
               cb();
         }
      );
}
```

bizproc.activity.add

Описание

bizproc.activity.add - добавляет новое действие в бизнес-процесс.

Каждый документ генерирует свой набор типов полей, с которыми он может работать. Например, у CRM есть поле типа **Адрес**, он обозначается как `UF:address`. Чтобы такой тип поля использовать в своих активити, нужно указать, что мы работаем с документом CRM (ключ DOCUMENT_TYPE) и тогда можно описывать свойства такого типа (ключ PROPERTIES).

Параметры

Параметр	Описание
CODE*	Внутренний идентификатор действия, уникальн рамках приложения. Допустимые символы a-z, A-Z, 0-9, точка, дефи нижнее подчеркивание.
HANDLER*	URL, на который действие будет отсылать данн сервер очередей *bitrix24*), когда бизнес-процес до его выполнения. Должен ссылаться на тот же домен, на котором установлено приложение.
AUTH_USER_ID	ID пользователя, токен которого будет передан приложению.
USE_SUBSCRIPTION	Использование подписки. Допустимые значени N. Можно указать, должно ли ожидать действие

	приложения. Если параметр пустой или не указ пользователь может сам настроить этот параме настройках действия в дизайнере бизнес-проце
NAME*	Название действия. Может быть строкой или ассоциативным массивом локализированных ст
DESCRIPTION	Описание действия. Может быть строкой или ассоциативным массивом локализированных ст
PROPERTIES	Массив параметров действия. Список значений аналогичен значениям параметра RETURN_PRO
RETURN_PROPERTIES	Массив возвращаемых значений действия. Пар управляет возможностью действия ожидать отв приложения и работать с данными, которые пр ответе. **Внимание**! Системное название параметра дол начинаться с буквы и может содержать только **a-z**, **A-Z**, **0-9** и нижнее подчеркивание. Каждый параметр обязательно должен содержа ▪ **Name** - строка или массив локализаций. ▪ **Description** - описание параметра, строка массив локализаций. ▪ **Type** - тип параметра. Список базовых па ▪ `bool` (**Да/Нет**) ▪ `date` (**Дата**) ▪ `datetime` (**Дата/Время**) ▪ `double` (**Число**) ▪ `int` (**Целое число**) ▪ `select` (**Список**) массив значений сп ▪ `string` (**Строка**) ▪ `text` (**Текст**) ▪ `user` (**Пользователь**), может иметь з вида: ▪ цифра - ID группы пользователе ▪ user_1 - пользователь с ID 1;

	▪ group_D12 - сотрудники отдела
	другие коды (группы соцсети и
	
	▪ **Options** Только для **TYPE** равному **select**
	`[`
	`'value1' => 'title1',`
	`'value2' => 'title2',`
	`'value3' => 'title3',`
	`'value4' => 'title4',`
	`]`
	▪ **Required(Y/N)** - обязательность парамет
	▪ **Multiple(Y/N)** - множественность параме
	▪ **Default** - значение параметра по-умолчан
	умолчанию тип параметра - **string**, необяз
	немножественный.

DOCUMENT_TYPE	Тип документа, который будет определять типы
	для параметров PROPERTIES и RETURN_PROPER
	Массив из 3 элементов:
	▪ id модуля
	▪ сущность (класс)
	▪ непосредственно тип документа
	Примеры:
	`['crm', 'CCrmDocumentLead', 'LEAD']`
	`['lists', 'BizprocDocument', 'iblock_22']`
	`['disk', 'Bitrix\Disk\BizProcDocument',`
	`'STORAGE_490']`
	`['tasks',`
	`'Bitrix\Tasks\Integration\Bizproc\Documen`
	`'TASK_PROJECT_13']`

FILTER	Правила ограничения действия по типу докуме редакции.
USE_PLACEMENT	Дает возможность открывать дополнительные н действия в слайдере приложения. Принимает з (Y/N).

* - обязательные параметры

Примеры

```
var params = {
   'CODE': 'md5',
   'HANDLER':
'http://yanzh.net16.net/ping.php',
   'AUTH_USER_ID': 1,
   'USE_SUBSCRIPTION': 'Y',
   'NAME': {
      'ru': 'MD5 генератор',
      'en': 'MD5 generator'
   },
   'DESCRIPTION': {
      'ru': 'Действие возвращает MD5 хеш от
входящего параметра',
      'en': 'Activity returns MD5 hash of
input parameter'
   },
   'PROPERTIES': {
      'inputString': {
         'Name': {
            'ru': 'Входящая строка',
            'en': 'Input string'
         },
         'Description': {
            'ru': 'Введите строку, которую
вы хотите хешировать',
            'en': 'Input string for hashing'
         },
         'Type': 'string',
         'Required': 'Y',
         'Multiple': 'N',
         'Default': '{=Document:NAME}'
      }
   },
   'RETURN_PROPERTIES': {
      'outputString': {
```

```
            'Name': {
               'ru': 'MD5',
               'en': 'MD5'
            },
            'Type': 'string',
            'Multiple': 'N',
            'Default': null
         }
      },
      'DOCUMENT_TYPE': ['lists',
   'BizprocDocument', 'iblock_1'],
      'FILTER': {
         INCLUDE: [
            ['lists']
         ]
      }
   };

   BX24.callMethod(
      'bizproc.activity.add',
      params,
      function(result)
      {
         if(result.error())
            alert("Error: " + result.error());
         else
            alert("Success: " + result.data());
      }
   );
```

Пример параметров Бизнес-процесса

```
   select
   'docType': {
           'Name': {
            'ru': 'Тип документа',
```

```
          'en': 'Document type'
        },
        'Required': 'Y',
        'Multiple': 'N',
        'Default': 'PDF',
        'Type': 'select',
        'Options': {
          'pdf': 'PDF',
          'docx': 'DOCX'
        }
      }

bool
'saveDoc': {
         'Name': {
             'ru': 'Сохранить документ',
             'en': 'Save document'
          },
          'Description': {
             'ru': 'Присвоить порядковый
номер',
             'en': 'Assign a sequential
number'
          },
          'Type': 'bool',
          'Required': 'Y',
          'Multiple': 'N',
          'Default': 'Y'
       }

string
'Parameters': {
          'Name': {
             'ru': 'Параметры шаблона',
             'en': 'Template\'s parameters'
          },
          'Description': {
```

```
            'ru': 'ParamID={=ParamValue}',
            'en': 'ParamID={=ParamValue}'
        },
        'Type': 'string',
        'Required': 'N',
        'Multiple': 'Y'

    }
```

bizproc.activity.log

bizproc.activity.log - записывает информацию в лог бизнес-процесса.

Параметр	Описание
EVENT_TOKEN	Уникальный ключ, который необходимо использовать при отправке события бизнес-процессу.
LOG_MESSAGE	Сообщения для записи в лог.

Пример

```
var params = {
   event_token:
'55c1dc1c3f0d75.78875596|A51601_82584_96831_
81132|hsyUws1j4XiwqPqN45eH66CcQtEvpUIP.47dd5
d888e8e549d2c984713e12a4268e6e87d0208ca1f093
ba1075e77f92e90',
   log_message: 'Please wait for answer!'
};

BX24.callMethod(
   'bizproc.activity.log',
   params,
   function(result)
   {
      if(result.error())
         alert("Error: " + result.error());
```

```
        else
            alert("Success: " + result.data());
    }
);
```

bizproc.activity.delete

bizproc.activity.delete - метод удаляет действие.

Внимание! При удалении и при обновлении приложения все действия, связанные с приложением, удаляются!

Параметр	Описание
CODE	Идентификатор действия приложения.

Пример

```
var params = {
   code: 'md5'
};

BX24.callMethod(
   'bizproc.activity.delete',
   params,
   function(result)
   {
      if(result.error())
         alert('Error: ' + result.error());
      else
         alert("Success: " + result.data());
   }
);
```

bizproc.activity.list

bizproc.activity.list - возвращает список установленных приложением действий.

Пример

```
BX24.callMethod(
   'bizproc.activity.list',
   {},
   function(result)
   {
      if(result.error())
         alert("Ошибка: " + result.error());
      else
         alert("Успешно: " +
result.data().join(', '));
   }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

bizproc.activity.update

Метод позволяет обновить поля действия. Параметры метода аналогичны bizproc.activity.add.

Пример

```
function updateActivity1()
{
   var params = {
      'CODE': 'hash',
      'FIELDS': {
         'DOCUMENT_TYPE': '',
         'FILTER': ''
      },
   };

   BX24.callMethod(
      'bizproc.activity.update',
      params,
      function(result)
      {
         if(result.error())
            alert("Error: " +
result.error());
         else
            alert("Успешно: " +
result.data());
      }
```

```
    );
}
```

bizproc.task.complete

Метод осуществляет выполнение заданий БП. В настоящий момент можно выполнить задания Утверждение документа и Ознакомление с документом.

С версии **20.0.800** модуля **Бизнес-процессы** доступно также выполнение задания Запрос доп.информации. Выполнить можно только свое задание и, только то, которое еще не выполнено.

Параметры

Параметр	Описание	С версии
TASK_ID	Идентификатор задания, обязательный	
STATUS	Целевой статус задания, обязательный. Список допустимых значений: ▪ 1 или yes - ответ "Да" (утвержден) ▪ 2 или no - ответ "Нет" (отклонен) ▪ 3 или ok - ответ "Ок" (ознакомлен) ▪ 4 или cancel - ответ "Отмена" (для Запроса доп.информации с отклонением)	

COMMENT	Комментарий пользователя, обязательность зависит от параметров задания	

Пример

```
function completeTask(id, status, comment,
cb)
    {
        var params = {
            TASK_ID: id,
            STATUS: status,
            COMMENT: comment
        };

        BX24.callMethod(
            'bizproc.task.complete',
            params,
            function(result)
            {
                if(result.error())
                    alert("Error: " +
result.error());
                else if (cb)
                    cb();
            }
        );
    }
```

Выполнение задания Запрос дополнительной информации

через REST

С версии **20.0.800** модуля Бизнес-процессы появилась возможность выполнять задания **Запрос доп.информации** через rest метод **bizrpoc.task.complete**.

Для того, чтобы понять, какие поля нужно заполнить, в метод bizproc.task.list в PARAMETERS добавлено новое свойство `Fields` - массив с описанием полей.

```
      "PARAMETERS": {
        "CommentLabel": "Комментарий",
        "CommentRequired": "N",
        "ShowComment": "Y",
        "StatusOkLabel": "Сохранить",
        "Fields": [
          {
            "Type": "datetime",
            "Name": "date",
            "Description": "",
            "Multiple": false,
            "Required": true,
            "Options": null,
            "Settings": null,
            "Default": "2020-07-
08T15:16:12+02:00",
            "Id": "date"
          }
        ]
      }
```

Значения **по умолчанию** хранятся в разделе **Default**. Значения конвертируются во «внешнее» представление (для дат - в формат rest ATOM (ISO-8601), а для файлов - в ссылку на файл).

Далее значения этих полей нужно передать в метод **bizrpoc.task.complete** в параметре **Fields**. Значения конвертируются в этот раз во «внутреннее представление» (т.е.

даты из rest формата конвертируются во внутренний, а файлы из rest сохраняются и прикрепляются к бизнес-процессу).

bizproc.task.list

Описание

bizproc.task.list - метод возвращает список заданий бизнес-процессов. Метод доступен не только для администраторов. Обычный пользователь может запросить задания свои или своего подчиненного. Для запроса своих заданий не-администратору не указывать фильтр по `USER_ID`.

Параметры

Параметр	Описание	Знач умол
SELECT	Массив полей записей, которые будут возвращены методом. Можно указать только те поля, которые необходимы. Доступные поля: `ID` - идентификатор задания; `WORKFLOW_ID` - идентификатор бизнес-процесса; `DOCUMENT_NAME` - название документа; `DESCRIPTION` - описание задания; `NAME` - название задания; `MODIFIED` - дата изменения;	`['ID', 'WORKFI 'DOCUME 'NAME']` **Важно 'MODU 'ENTIT 'DOCUM 'DOCUM** - возвра всегда.

WORKFLOW_STARTED - дата запуска бизнес-процесса;

WORKFLOW_STARTED_BY - кем запущен бизнес-процесс;

OVERDUE_DATE - крайний срок;

WORKFLOW_TEMPLATE_ID - идентификатор шаблона бизнес-процесса;

WORKFLOW_TEMPLATE_NAME - название шаблона бизнес-процесса;

WORKFLOW_STATE - статус бизнес-процесса;

STATUS - статус задания:

- 0 - выполняется;
- 1 - утверждено (ответ **Да**);
- 2 - отклонено (ответ **Нет**);
- 3 - выполнено (ответ **Ок**);
- 4 - таймаут (истек срок выполнения задания).

USER_ID - идентификатор пользователя;

USER_STATUS - ответ пользователя:

- 0 - ожидание ответа;
- 1 - да (утвердил);
- 2 - нет (отклонил);
- 3 - ок (выполнил).

MODULE_ID - идентификатор модуля (по документу);

ENTITY - идентификатор сущности (по документу);

DOCUMENT_ID - идентификатор документа.

`ACTIVITY` - идентификатор типа задания, строка, варианты значений:

- `ApproveActivity` - Утверждение документа
- `ReviewActivity` - Ознакомление с документом
- `RequestInformationActivity` - Запрос дополнительной информации
- `RequestInformationOptionalActivity` - Запрос дополнительной информации (с отклонением)

`ACTIVITY_NAME` - идентификатор действия в шаблоне.

`PARAMETERS` - параметры задания, массив, который может содержать следующую информацию:

- `CommentLabelMessage` - Название поля "Комментарий";
- `CommentRequired` - Обязательность комментария. Допустимые значения N (нет), Y (да), YA (да при утверждении), YR (да при отклонении);
- `ShowComment` - Показывать комментарий, Y/N;
- `TaskButtonMessage` - текст кнопки "Ознакомлен";
- `TaskButton1Message` - текст кнопки "Утвердить";
- `TaskButton2Message` - текст кнопки "Отклонить";
- `Fields` - массив с описанием полей ([dw]с версии 20.0.800[/dw][di]С версии **20.0.800** модуля Бизнес-процессы появилась возможность выполнять задания **Запрос доп.информации** через rest метод bizrpoc.task.complete. Для того, чтобы понять, какие поля нужно заполнить, в метод **bizproc.task.list** в

	PARAMETERS добавлено новое свойство `Fields`.[/di]);	
FILTER	Массив вида `{"фильтруемое_поле": "значение фильтра" [, ...]}`. Список фильтруемых полей такой же, как для параметра `SELECT`. Перед названием фильтруемого поля может указать тип фильтрации: ▪ `"!"` - не равно; ▪ `"<"` - меньше; ▪ `"<="` - меньше либо равно; ▪ `">"` - больше; ▪ `">="` - больше либо равно.	Если в с присутс `USER_ID` проверя суборди пользов Началы запроси заданий подчине Admini может запраш задания ограни
ORDER	Массив для сортировки результата. Массив вида `{"поле_сортировки": 'направление сортировки' [, ...]}`. Список полей для сортировки такой же, как для параметра `SELECT`. Направление сортировки может принимать значения: ▪ `asc` - по возрастанию; ▪ `desc` - по убыванию.	`{'ID':`

Пример

```
BX24.callMethod(
   'bizproc.task.list',
   {
      select: [
```

```
            'ID',
            'WORKFLOW_ID',
            'DOCUMENT_NAME',
            'DESCRIPTION',
            'NAME',
            'MODIFIED',
            'WORKFLOW_STARTED',
            'WORKFLOW_STARTED_BY',
            'OVERDUE_DATE',
            'WORKFLOW_TEMPLATE_ID',
            'WORKFLOW_TEMPLATE_NAME',
            'WORKFLOW_STATE',
            'STATUS',
            'USER_ID',
            'USER_STATUS',
            'MODULE_ID',
            'ENTITY',
            'DOCUMENT_ID'
         ],
         order: {ID: 'DESC'},
         filter: {'USER_ID': 1}
      },
      function(result)
      {
         if(result.error())
            alert("Error: " + result.error());
         else
            console.log(result.data());
      }
   );
```

bizproc.provider.add

bizproc.provider.add - регистрирует новый SMS-провайдер в бизнес-процессе.

Параметры

Параметр	Описание
CODE	Внутренний идентификатор провайдера. Допустимые символы a-z, A-Z, 0-9, точка, дефис и нижнее подчеркивание.
TYPE	Тип провайдера.
HANDLER	URL приложения, на который будут отправлены данные.
NAME	Название провайдера. Может быть строкой или ассоциативным массивом локализированных строк.
DESCRIPTION	Описание провайдера. Может быть строкой или ассоциативным массивом локализированных строк.

Пример

```
var params = {
        CODE: 'provider1',
        TYPE: 'SMS',
        HANDLER: 'http:///',
```

```
                        NAME: 'Провайдер
SMS.ru',
                        DESCRIPTION:
'Провайдер SMS.ru'
                };

                BX24.callMethod(

'bizproc.provider.add',
                        params,
                        function(result)
                        {

if(result.error())

alert("Error: " + result.error());
                                else

alert("Успешно: " + result.data());
                        }
                );
```

bizproc.provider.delete

bizproc.provider.delete - метод удаляет зарегистрированного SMS-провайдера.

Параметр	Описание
CODE	Идентификатор провайдера.

Пример

```
        function uninstallProvider(provider)
        {
                BX24.callMethod(

'bizproc.provider.delete',
                        {
                                'CODE':
provider
                        },
                        function(result)
                        {

if(result.error())

alert('Error: ' + result.error());
                                else

alert("Успешно: " + result.data());
```

```
                                        }
                        );

            }
```

bizproc.provider.list

bizproc.activity.list - возвращает список зарегистрированных приложением SMS-провайдеров.

Пример

```
BX24.callMethod(
'bizproc.provider.list',
{},
function(result)
{
if(result.error())
alert("Error: " + result.error());
else
alert("Успешно: " + result.data().join(', '));
}
);
```

bizproc.robot.update

Метод обновляет поля робота. В массив **FIELDS** передаются параметры, описанные в bizproc.robot.add.

Пример

```
function updateRobot1()
{
   var params = {
      'CODE': 'hash',
      'FIELDS': {
         'DOCUMENT_TYPE': '',
         'FILTER': ''
      },
   };

   BX24.callMethod(
      'bizproc.robot.update',
      params,
      function(result)
      {
         if(result.error())
            alert("Error: " +
result.error());
         else
            alert("Успешно: " +
result.data());
      }
```

```
    );
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

bizproc.robot.add

Описание

bizproc.robot.add - метод регистрирует нового робота.

Параметры

Параметр	Описание
CODE	Внутренний идентификатор робота. Допустимые символы a-z, A-Z, 0-9, точка, дефис и нижнее подчеркивание. Обязательный параметр.
HANDLER	URL приложения, на который будут отправлены данные. Обязательный параметр.
AUTH_USER_ID	ID пользователя, токен которого будет передан приложению.
NAME	Название робота. Может быть строкой или ассоциативным массивом локализированных строк. Обязательный параметр.
USE_SUBSCRIPTION	Использование подписки. Допустимые значения - Y или N. Можно указать, должен ли робот ожидать ответа от приложения. Если параметр пустой или не указан - пользователь может сам настроить этот параметр в

	настройках действия в дизайнере бизнес-процессов.
PROPERTIES	Массив параметров робота. Список значений аналогичен значениям параметра RETURN_PROPERTIES.
USE_PLACEMENT	Дает возможность открывать дополнительные настройки робота в слайдере приложения. Принимает значения (Y/N). Необязательный параметр.
PLACEMENT_HANDLER	URL встройки (обработчик встройки на стороне приложения). Если использовать параметр USE_PLACEMENT со значением "Y", но не указать PLACEMENT_HANDLER, то возникает [dw]ошибка[/dw][di] `"error": "ERROR_WRONG_HANDLER_URL", "error_description": "Wrong handler URL"`[/di].
RETURN_PROPERTIES	Массив возвращаемых значений робота. Параметр управляет возможностью ожидать ответа приложения роботом и работать с данными, которые придут в ответе. **Внимание**! Системное название параметра должно начинаться с буквы и может содержать только символы **a-z**, **A-Z**, **0-9** и нижнее подчеркивание. Каждый параметр обязательно должен содержать: ▪ **Name** - строка или массив локализаций. ▪ **Description** - описание параметра, строка или массив локализаций. ▪ **Type** - тип параметра. Список базовых параметров:

- `bool` (**Да/Нет**)
- `date` (**Дата**)
- `datetime` (**Дата/Время**)
- `double` (**Число**)
- `int` (**Целое число**)
- `select` (**Список**) массив значений списка:
- `string` (**Строка**)
- `text` (**Текст**)
- `user` (**Пользователь**)

- **Options** Только для **TYPE** равному **select**.

```
[
'value1' => 'title1',
'value2' => 'title2',
'value3' => 'title3',
'value4' => 'title4',
]
```

- **Required(Y/N)** - обязательность параметра.
- **Multiple(Y/N)** - множественность параметра.
- **Default** - значение параметра по-умолчанию. По умолчанию тип параметра - **string**, необязательный, немножественный.

Пример

```
var params = {
        'CODE': 'robot',
        'HANDLER':
'http:///robot.php',
        'AUTH_USER_ID': 1,
        'NAME': 'Пример
```

```
робота',
                        'PROPERTIES': {
                                'bool': {

'Name': 'Да/Нет',

'Type': 'bool',

'Required': 'Y',

'Multiple': 'N'
                                },
                                'date': {

'Name': 'Дата',

'Type': 'date'
                                },
                                'datetime':
{

'Name': 'Дата/Время',

'Type': 'datetime'
                                },
                                'double': {

'Name': 'Число',

'Type': 'double',

'Required': 'Y'
                                },
                                'int': {

'Name': 'Целое число',
```

```
'Type': 'int'
                                    },
                                    'select': {

'Name': 'Список',

'Type': 'select',

'Options': {

'one': 'one',

'two': 'two'
                                            }
                                    },
                                    'string': {

'Name': 'Строка',

'Type': 'string',

'Default': 'default string value'
                                    },
                                    'text': {

'Name': 'Текст',

'Type': 'text'
                                    },
                                    'user': {

'Name': 'Пользователь',

'Type': 'user'
                                    }
                            }
                    };
```

```
                BX24.callMethod(
                        'bizproc.robot.add',
                        params,
                        function(result)
                        {

if(result.error())

alert("Error: " + result.error());
                                else

alert("Успешно: " + result.data());
                        }
                );
```

bizproc.robot.delete

bizproc.provider.delete - метод удаляет зарегистрированного робота.

Параметр	Описание
CODE	Идентификатор робота.

Пример

```
var params = {
        'CODE': 'robot'
};

BX24.callMethod(
'bizproc.robot.delete',
params,
function(result)
{
if(result.error())
alert('Error: ' + result.error());
else
alert("Успешно: " + result.data());
```

```
                                                    }
                        );
```

bizproc.robot.list

bizproc.robot.list - возвращает список зарегистрированных приложением роботов.

Пример

```
BX24.callMethod(
'bizproc.robot.list',
{},
function(result)
{
if(result.error())
alert("Error: " + result.error());
else
alert("Успешно: " + result.data().join(', '));
}
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

bizproc.event.send

bizproc.event.send - метод возвращает действию выходные параметры, заданные в описании действия.

Параметр	Описание
EVENT_TOKEN	Уникальный ключ, который необходимо использовать при отправке события бизнес-процессу.
RETURN_VALUES	Массив возвращаемых значений действия. Также будет появляться в форме Вставка значения во вкладке **Дополнительные результаты**.

Пример

```
var params = {
   event_token:
'55c1dc1c3f0d75.78875596|A51601_82584_96831_
81132|hsyUws1j4XiwqPqN45eH66CcQtEvpUIP.47dd5
d888e8e549d2c984713e12a4268e6e87d0208ca1f093
ba1075e77f92e90',
   return_values: {
      outputString:
'846c55d14f552180874a628d2615e285'
   }
};

BX24.callMethod(
   'bizproc.event.send',
```

```
    params,
    function(result)
    {
       if(result.error())
          alert("Error: " + result.error());
       else
          alert("Success: " + result.data());
    }
 );
```

bizproc.workflow.template.add

Метод добавляет шаблон Бизнес-процесса. Требует прав администратора.

Параметры

Параметр	Описание

DOCUMENT_TYPE	Тип документа - массив из 3 элементов:
	▪ id модуля
	▪ сущность (класс)
	▪ непосредственно тип документа
	Примеры:
	`['crm', 'CCrmDocumentLead', 'LEAD']`
	`['lists', 'BizprocDocument', 'iblock_22']`
	`['disk', 'Bitrix\Disk\BizProcDocument', 'STORAGE_490']`
	`['tasks', 'Bitrix\Tasks\Integration\Bizproc\Document\T`
	`'TASK_PROJECT_13']`

NAME	Название шаблона
DESCRIPTION	Описание шаблона.
TEMPLATE_DATA	Передается контент файла *.bpt (стандартно для р
AUTO_EXECUTE	Флаги автозапуска, может быть: 0, 1 (создание), 2

	(изменение) , 3 (создание + изменение)

Пример

```
function addTemplate()
{
   BX24.callMethod(
      'bizproc.workflow.template.add',
      {
         DOCUMENT_TYPE: ['crm',
'CCrmDocumentLead', 'LEAD'],
         NAME: 'App template',
         DESCRIPTION: 'Template was
generated by rest application.',
         AUTO_EXECUTE: 1,
         TEMPLATE_DATA:
document.getElementById('tpl_file')
      },
      function(result)
      {
         if(result.error())
            alert("Error: " +
result.error());
         console.log(result);
      }
   );
}
```

bizproc.workflow.template.del

Метод удаляет шаблон Бизнес-процесса. Требует прав администратора. Метод удаляет только те шаблоны, которые были созданы методом bizproc.workflow.template.add. Так как такие шаблоны привязываются к приложению и только их можно удалить.

Параметры

Параметр	Описание	С версии
ID	Идентификатор удаляемого шаблона.	

Пример

```
function deleteTemplate(id)
{
   BX24.callMethod(
      'bizproc.workflow.template.delete',
      {ID: id},
      function(result)
      {
         if(result.error())
            alert("Error: " +
```

```
result.error());
            console.log(result);
        }
    );
}
```

bizproc.workflow.template.list

Описание

bizproc.workflow.template.list - метод возвращает список шаблонов Бизнес-процессов, установленных на сайте. Использование метода требует прав администратора.

Параметры

Метод	Описание
ID	Идентификатор бизнес-процесса.
MODULE_ID	Идентификатор модуля (по документу)
ENTITY	Идентификатор сущности (по документу)
DOCUMENT_TYPE	Тип документа
AUTO_EXECUTE	правило автозапуска: 0 - без автозапуска 1 - при добавлении 2- при изменении 3 - добавление + изменение
NAME	Название шаблона
TEMPLATE	Шаблон БП (массив с описанием структуры действий).
PARAMETERS	Параметры шаблона, массив с описанием

	свойств.
VARIABLES	Переменные шаблона, массив с описанием свойств.
CONSTANTS	Константы шаблона, массив с описанием свойств.
MODIFIED	Дата последнего изменения.
IS_MODIFIED	[Y\N] Флаг был ли изменен. Актуально для шаблонов, поставляемых "в коробке" (у которых есть системный код)
USER_ID	идентификатор пользователя кто создал/изменил шаблон.
SYSTEM_CODE	системный код шаблона. применяется для идентификации типовых шаблонов, Процессов в ленте, шаблонов автоматизации и т.п.

Пример

```
function getTemplates()
{
   BX24.callMethod(
      'bizproc.workflow.template.list',
      {
         select: [
            'ID',
            // 'MODULE_ID',
            // 'ENTITY',
            // 'DOCUMENT_TYPE',
            // 'AUTO_EXECUTE',
            'NAME',
            // 'TEMPLATE',
            // 'PARAMETERS',
```

```
            // 'VARIABLES',
            // 'CONSTANTS',
            'MODIFIED',
            'IS_MODIFIED',
            'USER_ID',
            'SYSTEM_CODE'
         ],
         filter: {MODULE_ID: 'crm', ENTITY:
'CCrmDocumentLead'}
      },
      function(result)
      {
         if(result.error())
            alert("Error: " +
result.error());
         else
         {
            var templates = result.data();

            console.log(templates);
         }
      }
   );
}
```

bizproc.workflow.template.upd

Метод изменяет шаблон Бизнес-процесса. Требует прав администратора. Метод обновляет только те шаблоны, которые были созданы методом bizproc.workflow.template.add. Так как такие шаблоны привязываются к приложению и только их можно обновлять.

Параметры

Параметр	Описание	С версии
ID	Идентификатор изменяемого шаблона.	
FIELDS	Массив изменяемых параметров. Обновить можно поля: NAME, DESCRIPTION, AUTO_EXECUTE, TEMPLATE_DATA. При попытке обновить другие поля возвращаемые bizproc.workflow.template.list - ошибок не возникнет, но и обновлено ничего не будет.	

Пример

```
function renameTemplate(id, name)
{
   BX24.callMethod(
      'bizproc.workflow.template.update',
      {ID: id, FIELDS: {'NAME': name}},
      function(result)
      {
         if(result.error())
            alert("Error: " +
result.error());
         console.log(result);
      }
   );
}
```

BX24.placement.info

```
BX24.placement.info();
```

Получение информации о контексте вызова.

Результат:

```
{"placement":"CRM_LEAD_LIST_MENU","options":
{"ID":"1348"}}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

JS методы, доступные в карточке CRM

В карточке CRM доступна дополнительная команда **reloadData**, которая обновит интерфейсные данные CRM. Эта команда служит исключительно для передачи данных в карточку CRM, следовательно, пока карточка реально не будет сохранена пользователем, значения в базу не попадут.

```
BX24.placement.call('reloadData', function()
{console.log('reload call')});
```

js-интерфейс служит исключительно для передачи данных в форму. Пока форма не будет сохранена пользователем, значение в базу не попадет.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

placement.bind

Метод установит обработчик места встраивания.

Обработчик ограничен условиями:

- запрос исполняется с авторизацией администратора,
- обработчик лежит в том же домене, что и зарегистрированный **redirect_uri** приложения
- каждое места встраивания привязано к праву доступа (scope). Чтобы установить обработчик места встраивания в интерфейсе CRM, дайте приложению доступ к CRM.

После установки приложений с интерфейсом необходимо выполнить метод BX24.installFinish после запроса. В противном случае кнопки не отобразятся в местах встраивания. Для серверных локальных приложений без интерфейса в Битрикс24 плейсменты отображаются сразу.

С версии **rest 21.200.0** метод поддерживает многоязычность.

Параметры

Параметр	Описание
PLACEMENT	Идентификатор требуемого места встраивания. Обязателен.
HANDLER	URL обработчика места встраивания. Обязателен.
LANG_ALL	Массив параметров для каждого языка. Ключи массива:

	▪ TITLE - заголовок обработчика, будет выводиться по месту встраивания. Не обязателен, по умолчанию - название приложения ▪ DESCRIPTION - Описание обработчика, может выводиться по месту встраивания. Не обязателен. ▪ GROUP_NAME - Название группы, который позволяет группировать обработчики в месте встраивания. Не обязателен.

Пример

```
https://portal.bitrix24.com/rest/placement.bind/?auth=sode3flffcmv500fuagrprhllx3soi72
    &PLACEMENT=CRM_CONTACT_LIST_MENU

&HANDLER=http%3A%2F%2Fwww.applicationhost.com%2Fplacement%2F
    &TITLE=Demo

HTTP/1.1 200 OK

{
    "result": true
}
```

Пример с поддержкой нескольких языков:

```
CRest::call(
   'placement.bind',
   [
      'PLACEMENT' => 'PLACEMENT_CODE',
```

```
        'HANDLER' =>
'https://example.com/place.php',
        'LANG_ALL' => [
            'en' => [
                'TITLE' => 'title',
                'DESCRIPTION' => 'description',
                'GROUP_NAME' => 'group',
            ],
            'ru' => [
                'TITLE' => 'заголовок',
                'DESCRIPTION' => 'описание',
                'GROUP_NAME' => 'группа',
            ],
        ],
    ]
);
```

placement.get

Метод для получения списка зарегистрированных обработчиков мест встраивания.

Параметры

Без параметров.

Пример

```
GET
https://sometestportal.bitrix24.com/rest/placement.get?
auth=7e623a5a0000cd710000cd5b00000001000000a8b1dbe022e2de93198634e9526b00f7 HTTP/1.1

HTTP/1.1 200 OK

{
    "result": [
        {
            "description": "",
            "handler":
"https://www.myapplicationhost.com/placement/",
            "placement":
"CRM_LEAD_DETAIL_TAB",
```

```
                "title": "My test handler"
            }
        ]
    }
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

placement.list

Метод для получения списка доступных приложению мест встраивания.

Параметры

Параметр	Описание	С версии
SCOPE	Ограничение списка одним из прав доступа приложения	

Пример

```
GET
https://sometestportal.bitrix24.com/rest/placement.list?
auth=7e623a5a0000cd710000cd5b00000001000000a8b1dbe022e2de93198634e9526b00f7 HTTP/1.1

HTTP/1.1 200 OK

{
    "result": [
        "CRM_LEAD_LIST_MENU",
```

```
            "CRM_DEAL_LIST_MENU",
            "CRM_INVOICE_LIST_MENU",
            "CRM_QUOTE_LIST_MENU",
            "CRM_CONTACT_LIST_MENU",
            "CRM_COMPANY_LIST_MENU",
            "CRM_ACTIVITY_LIST_MENU",
            "CRM_LEAD_DETAIL_TAB",
            "CRM_DEAL_DETAIL_TAB",
            "CRM_CONTACT_DETAIL_TAB",
            "CRM_COMPANY_DETAIL_TAB",
            "CRM_LEAD_DETAIL_ACTIVITY",
            "CRM_DEAL_DETAIL_ACTIVITY",
            "CRM_CONTACT_DETAIL_ACTIVITY",
            "CRM_COMPANY_DETAIL_ACTIVITY"
        ]
}
```

placement.unbind

Метод удаляет зарегистрированный обработчик места встраивания. Должен исполняться с правами администратора портала.

Параметры

Параметр	Описание	С версии
PLACEMENT	Идентификатор места встраивания. Обязателен.	
HANDLER	URL обработчика места встраивания. Если не указывать URL обработчика, будут удалены все обработчики указанного места встраивания, зарегистрированные приложением.	

Метод возвращает количество удаленных обработчиков.

Пример

```
GET
https://sometestportal.bitrix24.com/rest/placement.unbind?
```

```
auth=7e623a5a0000cd710000cd5b000000010000000a
8b1dbe022e2de93198634e9526b00f7&placement=CR
M_LEAD_DETAIL_TAB&handler=https%3A%2F%2Fwww.
myapplicationhost.com%2Fplacement%2F
HTTP/1.1

HTTP/1.1 200 OK

{
    "result": {
        "count": 1
    }
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

userfieldtype.add

Регистрация нового типа пользовательских полей. Метод возвращает true или ошибку с описанием причины.

Параметры

Параметр	Тип	Описание	Ограничения
USER_TYPE_ID	Строка	Строковой код типа. Обязательный параметр.	a-z0-9
HANDLER	URL	Адрес обработчика пользовательского типа. Обязательный параметр.	Должен быть в том же домене, что и основной адрес приложения.
TITLE	Строка	Текстовое название типа. Будет выводиться в административном интерфейсе настройки пользовательских полей.	
DESCRIPTION	Строка	Текстовое описание типа. Будет выводиться	

		в административном интерфейсе настройки пользовательских полей.	
OPTIONS	Массив	Дополнительные настройки. На данный момент доступен один ключ: **height** - указывает высоту пользовательского поля по умолчанию в пикселях. [dw]По умолчанию - 0[/dw][di]При указании значения 0 - будет использована высота стандартная для отображения данной встройки. [/di]. Применится любое положительное значение.	

Примеры

Пример вызова

```
BX24.callMethod(
   'userfieldtype.add',
   {
      USER_TYPE_ID: 'test',
```

```
      HANDLER:
'https://www.myapplication.com/handler/',
      TITLE: 'Test type',
      DESCRIPTION: 'Test userfield type for
documentation'
   }
);
```

Пример запроса

```
POST
https://sometestportal.bitrix24.com/rest/use
rfieldtype.add HTTP/1.1

USER_TYPE_ID=test&HANDLER=https%3A%2F%2Fwww.
myapplication.com%2Fhandler%2F&TITLE=Test+ty
pe&DESCRIPTION=Test+userfield+type+for+docum
entation&auth=63t6r4z9cugaciaxocrh2r47zlodp1
2y

HTTP/1.1 200 OK

{
    "result": true
}
```

Пример с использованием параметра OPTIONS:

```
CRest::call(
    'userfieldtype.add',
    [
        'USER_TYPE_ID' => 'custom_type',
        'HANDLER' =>
'https://example.com/field.php',
```

```
        'TITLE' => 'title',
        'OPTIONS' => [
            'height' => 60,
        ],
    ]
);
```

userfieldtype.delete

Удаление зарегистрированного приложением типа пользовательских полей. Метод возвращает *true* или ошибку с описанием причины.

Параметры

Параметр	Тип	Описание	Ограничения
USER_TYPE_ID	Строка	Строковой код типа. Обязательный параметр.	a-z0-9

Примеры

Пример вызова

```
BX24.callMethod(
    'userfieldtype.delete',
    {
        USER_TYPE_ID: 'test'
    }
);
```

Пример запроса

```
POST
https://sometestportal.bitrix24.com/rest/use
rfieldtype.delete HTTP/1.1

USER_TYPE_ID=test&auth=63t6r4z9cugaciaxocrh2
r47zlodp12y

HTTP/1.1 200 OK

{
    "result": true
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

userfieldtype.list

Получение списка зарегистрированных приложением типов пользовательских полей. Списочный метод. На выход отдается список типов полей с постраничной навигацией.

Параметры

Входных параметров нет.

Примеры

Пример вызова:

```
BX24.callMethod(
   'userfieldtype.list',
   {},
   function(result)
   {
      console.log(result.data());
   }
);
```

Пример запроса

```
POST
https://sometestportal.bitrix24.com/rest/use
```

```
rfieldtype.list HTTP/1.1

auth=63t6r4z9cugaciaxocrh2r47zlodp12y

HTTP/1.1 200 OK

{
    "result": [
        {
            "DESCRIPTION": "Test userfield
type for documentation",
            "HANDLER":
"https://www.myapplication.com/handler/",
            "TITLE": "Test type",
            "USER_TYPE_ID": "test"
        }
    ],
    "total": 1
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

userfieldtype.update

Изменение настроек зарегистрированного приложением типа пользовательских полей. Метод возвращает *true* или ошибку с описанием причины.

Параметры

Параметр	Тип	Описание	Ограничения
USER_TYPE_ID	Строка	Строковой код типа. Обязательный параметр.	a-z0-9
HANDLER	URL	Адрес обработчика пользовательского типа. Обязательный параметр.	Должен быть в том же домене, что и основной адрес приложения.
TITLE	Строка	Текстовое название типа. Будет выводиться в административном интерфейсе настройки пользовательских полей.	
DESCRIPTION	Строка	Текстовое	

		описание типа. Будет выводиться в административном интерфейсе настройки пользовательских полей.	

Примеры

Пример вызова

```
BX24.callMethod(
   'userfieldtype.update',
   {
      USER_TYPE_ID: 'test',
      TITLE: 'Updated test type',
      DESCRIPTION: 'Test userfield type for
documentation with updated description'
   }
);
```

Пример запроса

```
POST
https://sometestportal.bitrix24.com/rest/use
rfieldtype.update HTTP/1.1

USER_TYPE_ID=test&TITLE=Updated+test+type&DE
SCRIPTION=Test+userfield+type+for+documentat
ion+with+updated+description&auth=63t6r4z9cu
gaciaxocrh2r47zlodp12y

HTTP/1.1 200 OK
```

```
{
    "result": true
}
```

Идентификаторы entityId

Принадлежность поля к конкретной сущности определяется значением поля **entityId**. Ниже приведены идентификаторы для разных сущностей:

CRM

- Лид - CRM_LEAD
- Контакт - CRM_CONTACT
- Компания - CRM_COMPANY
- Сделка - CRM_DEAL
- Предложение - CRM_QUOTE
- Счет - CRM_INVOICE
- Смарт-процесс - `CRM_ + {ID}`, где ID - это идентификатор смарт-процесса (не идентификатор типа)

RPA

В модуле rpa поле строится по правилу `RPA_ + {ID}`, где ID - это идентификатор процесса.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

userfieldconfig.add

Описание и параметры

```
userfieldconfig.add({moduleId: string,
field: {}})
```

Метод добавит новое пользовательского поле.

Параметры

Параметр	Описание	С версии
moduleId	Строковый идентификатор модуля. Обязательный.	
field	Список с полями настройки нового поля: ▪ entityId - строковый идентификатор сущности. Обязательное. ▪ fieldName - код поля. Должен быть сформирован по шаблону `UF_ + {идентификатор сущности} + _ + {произвольная`	

строка в UPPER_CASE}. Код поля не может быть больше 50 символов. Обязательное.

- userTypeId - строковый идентификатор [dw]типа поля[/dw][di]Типы полей идентичны полям для главного модуля Подробнее...[/di]. Обязательное.
- xmlId - внешний идентификатор.
- sort - индекс сортировки.
- multiple - флаг множественности (N или Y), по умолчанию N. Этот флаг можно указать только при создании поля.
- mandatory - флаг обязательности (N или Y), по умолчанию N.
- showFilter - флаг показа поля в фильтре (N или Y), по умолчанию N.
- showInList - флаг показа поля в списке (N или Y), по умолчанию Y.
- editInList - флаг разрешения редактирования поля в списке (N или Y), по умолчанию Y.
- isSearchable - флаг наличия значения поля в [dw]полнотекстовом индексе[/dw] [di]Добавляйте в поиск только нужные поля. На построение индекса уходит время при изменении каждого значения поля, что может существенно замедлить работы при большом количестве таких

	полей.[/di] (N или Y), по умолчанию N.
	▪ settings - список с дополнительными настройками поля.
	▪ editFormLabel - список с языкозависимыми названиями поля, где ключ - идентификатор языка, а значение - фраза.
	▪ enum - массив с вариантами значений для свойств типа "список":
	  ▪ value - значение варианта. Обязательное
	  ▪ def - флаг значения по умолчанию (N или Y), по умолчанию N. Только один может быть вариантом по умолчанию
	  ▪ sort - индекс сортировки. Если не задан, генерируется автоматически на основе порядка передачи вариантов значений
	  ▪ xmlId - внешний идентификатор варианта	

Возвращаемое значение и пример

Возвращаемое значение

Метод вернет такие же данные, как метод userfieldconfig.get на только что созданном поле.

Примеры

Пример простого запроса. Этого запроса достаточно для создания поля типа "строка".

```
{
    "moduleId": "rpa",
    "field": {
        "entityId": "RPA_1",
        "fieldName":
"UF_RPA_1_NEW_REST_STRING",
        "userTypeId": "string",
    }
}
```

Создание поля типа "список"

```
{
    "moduleId": "rpa",
    "field": {
        "entityId": "RPA_1",
        "fieldName":
"UF_RPA_1_NEW_REST_STRING",
        "userTypeId": "enumeration"
    }
}
```

userfieldconfig.delete

```
userfieldconfig.delete({moduleId: string,
id: number})
```

Метод удалит настройки поля с идентификатором id.

Параметры

Параметр	Описание	С версии
moduleId	Строковый идентификатор модуля. Обязательный.	
id	Идентификатор настроек поля. Обязательный.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

userfieldconfig.get

Описание и параметры

```
userfieldconfig.get({id: number, moduleId:
string})
```

Метод вернет данные о настройках пользовательского поля с идентификатором **id**.

Параметры

Параметр	Описание	С версии
id	Идентификатор настроек поля. Обязательный.	
moduleId	Строковый идентификатор модуля. Обязательный.	

Пример

Пример ответа.

```
{
    "field": {
        "id": "165",
        "entityId": "RPA_1",
        "fieldName": "UF_RPA_1_1585069397",
        "userTypeId": "file",
        "xmlId": null,
        "sort": "100",
        "multiple": "Y",
        "mandatory": "N",
        "showFilter": "E",
        "showInList": "Y",
        "editInList": "Y",
        "isSearchable": "Y",
        "settings": {
            "SIZE": 20,
            "LIST_WIDTH": 0,
            "LIST_HEIGHT": 0,
            "MAX_SHOW_SIZE": 0,
            "MAX_ALLOWED_SIZE": 0,
            "EXTENSIONS": []
        },
        "languageId": {
            "en": "en",
            "ru": "ru"
        },
        "editFormLabel": {
            "en": "",
            "ru": "Множественный файл"
        },
        "listColumnLabel": {
            "en": null,
            "ru": null
        },
        "listFilterLabel": {
            "en": null,
            "ru": null
```

```
            },
            "errorMessage": {
                "en": null,
                "ru": null
            },
            "helpMessage": {
                "en": null,
                "ru": null
            }
        }
    }
```

Где:

- id - идентификатор
- entityId - строковый идентификатор сущности
- fieldName - код поля
- userTypeId - строковый идентификатор типа поля
- xmlId - внешний идентификатор
- sort - индекс сортировки
- multiple - флаг множественности
- mandatory - флаг обязательности
- showFilter - флаг показа поля в фильтре
- showInList - флаг показа поля в списке
- editInList - флаг разрешения редактирования поля в списке
- isSearchable - флаг наличия значения поля в полнотекстовом индексе
- settings - список дополнительных настроек поля, зависит от его типа
- languageId - список идентификаторов языков, для которых есть фразы
- editFormLabel - список с языкозависимыми названиями поля, где ключ - идентификатор языка, а значение - фраза
- listColumnLabel - аналогичный список фраз для подписи поля в списке (не используется)
- listFilterLabel - аналогичный список фраз для подписи поля в фильтре (не используется)
- errorMessage - аналогичный список фраз для сообщения об ошибке (не используется)

- helpMessage - аналогичный список подсказок (не используется)
- enum - массив с вариантами значений для свойств типа список enumeration:
  - id - идентификатор варианта
  - userFieldId - идентификатор настроек поля
  - value - значение
  - def - флаг по умолчанию (N или Y), по умолчанию N. Только один может быть вариантом по умолчанию.
  - sort - индекс сортировки
  - xmlId - внешний идентификатор варианта, генерируется автоматически

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

userfieldconfig.getTypes

```
userfieldconfig.getTypes({moduleId: string})
```

Метод вернет набор доступных типов пользовательских полей для модуля moduleId.

Параметры

Параметр	Описание	С версии
moduleId	Идентификатор модуля. Обязательный.	

Пример

Пример ответа

```
{
    "types": {
        "employee": {
            "userTypeId": "employee",
            "description": "Привязка к
```

```
сотруднику"
        },
        "money": {
            "userTypeId": "money",
            "description": "Деньги"
        },
        "string": {
            "userTypeId": "string",
            "description": "Строка"
        },
        "integer": {
            "userTypeId": "integer",
            "description": "Целое число"
        },
        ...
    }
}
```

userfieldconfig.list

```
userfieldconfig.list({moduleId: string,
select: ?{}, order: ?{}, filter: ?{}, start:
number = 0})
```

Метод вернет список настроек пользовательских полей.

Параметры

Параметр	Описание	С версии
moduleId	Строковый идентификатор модуля. Обязательный.	
select	Массив с полями, которые надо показать. По умолчанию выводятся все, кроме вариантов для списка и языковых фраз. Чтобы получить фразы в списке, необходимо передать идентификатор языка по ключу **language**	
order	Список для определения порядка отображения, где ключ - название поля, а значение - ASC или DESC	

filter	Список для фильтрации	

Пример

Пример запроса

Найти все множественные настройки пользовательских полей из модуля *rpa*, отсортированные по убыванию *id*, со всеми полями и языковыми фразами для языка *ru*.

```
{
    "moduleId": "rpa",
    "select": {
        "0": "*",
        "language": "ru"
    },
    "order": {
        "id": "DESC"
    },
    "filter": {
        "multiple": "Y"
    }
}
```

Формат ответа в части языковых фраз немного отличается, так как здесь фразы только для одного языка.

```
{
    "fields": [
        {
            "id": "165",
            "entityId": "RPA_1",
            "fieldName":
"UF_RPA_1_1585069397",
```

```
            "userTypeId": "file",
            "xmlId": null,
            "sort": "100",
            "multiple": "Y",
            "mandatory": "N",
            "showFilter": "E",
            "showInList": "Y",
            "editInList": "Y",
            "isSearchable": "Y",
            "settings": {
                "SIZE": 20,
                "LIST_WIDTH": 0,
                "LIST_HEIGHT": 0,
                "MAX_SHOW_SIZE": 0,
                "MAX_ALLOWED_SIZE": 0,
                "EXTENSIONS": []
            },
            "languageId": {
                "ru": "ru"
            },
            "editFormLabel": {
                "ru": "Множественный файл"
            },
            "listColumnLabel": null,
            "listFilterLabel": null,
            "errorMessage": null,
            "helpMessage": null
        },
        {
            ...
        }
    ]
}
```

© «Битрикс», 2001-2008, «1С-

userfieldconfig.update

Описание и параметры

```
userfieldconfig.update({moduleId: string, id: number, field: {}})
```

Метод изменяет значение поля.

Параметры

Параметр	Описание	С версии
moduleId	Строковый идентификатор модуля. Обязательный.	
id	Идентификатор настроек поля. Обязательный.	
field	Список с полями настройки нового поля. Аналогично методу userfieldconfig.add, но: ▪ fieldName - не может быть изменен	

	▪ userTypeId - не может быть изменен ▪ entityId - не может быть изменен ▪ multiple - не может быть изменен ▪ isSearchable - изменение этого флага не вызовет автоматическое перестроение поискового индекса. Перестроение происходит при изменении сущностей, к которым привязаны поля ▪ enum - полный список всех вариантов значений для свойства типа "список". Чтобы это поле учитывалось, в fields должен присутствовать **userTypeId** ▪ id - идентификатор варианта. Должен присутствовать, если надо обновить вариант	

Будьте внимательны при работе с вариантами значений для списков.

Возвращаемое значение и пример

Возвращаемое значение

Метод вернет такие же данные, как метод userfieldconfig.get на измененном поле.

Примеры

Обновление флагов и языковых фраз, без изменения вариантов значений

```
{
    "moduleId": "rpa",
    "id": 170,
    "field": {
        "mandatory": "Y",
        "editFormLabel": {
            "ru": "Новое название поля"
        }
    }
}
```

Пример удаление всех вариантов значений.

```
{
    "moduleId": "rpa",
    "id": 170,
    "field": {
        "userTypeId": "enumeration",
        "enum": [
            [""]
        ]
    }
}
```

Пример частичного обновления вариантов значений.

```
{
    "moduleId": "rpa",
```

```
    "id": 170,
    "field": {
        "userTypeId": "enumeration",
        "enum": [
            {
                "id": 29,
            },
            {
                "id": 30,
                "value": "Обновленное
значение"
            },
            {
                "value": "Новое значение"
            }
        ]
    }
}
```

В этом примере:

- вариант значения с id=29 останется без изменений,
- у варианта с id=30 поменяется значение
- добавится новый вариант "Новое значение"
- все остальные варианты будут удалены

Примеры генерации документа

Здесь изложен полностью процесс генерации документа. Все примеры написаны на php через веб-хук.

Создание нумератора

Для начала создадим новый нумератор для реста с помощью `documentgenerator.numerator.add`.

```
 \Bitrix\Main\Loader::includeModule('rest');
 $client = new
\Bitrix\Main\Web\HttpClient();
 $webHookUrl =
'http://mycrm.bitrix24.com/rest/1/webhookkey
/';
 $prefix = 'documentgenerator';

 $data = [
        'fields' => [
                'name' => 'Rest Numerator',
                'template' => '{NUMBER}',
                'settings' => [

'Bitrix_Main_Numerator_Generator_SequentNumb
erGenerator' => [
                'start' => '0',
                'step' => '1',
            ],
```

```
                ],
        ],
 ];

 $url =
$webHookUrl.$prefix.'.numerator.add/';
 $answer = $client->post($url, $data);
 try
 {
        $result =
\Bitrix\Main\Web\Json::decode($answer);
 }
 catch(Exception $e)
 {
        var_dump($answer);
 }

 print_r($result);
```

Ответ

```
[result] => Array
    (
        [numerator] => Array
            (
                [name] => new rest numerator
                [template] => {NUMBER}
                [id] => 68
                [settings] => Array
                    (

[Bitrix_Main_Numerator_Generator_SequentNumb
erGenerator] => Array
                            (
                                [start] => 0
                                [step] => 1
```

```
                                    [periodicBy]
=>
                                    [timezone]
=>

[isDirectNumeration] =>
                              )

                     )

            )
    )
```

Из ответа получаем `id` нумератора и можем дальше его использовать.

Загрузка шаблона

Сначала необходимо загрузить шаблон. Весь код как в предыдущем примере, но другие входные данные и название метода:

`documentgenerator.template.add`

```
$data = [
        'fields' => [
                'name' => 'Rest Template',
                'file' =>
base64_encode(file_get_contents($_SERVER['DO
CUMENT_ROOT'].'/upload/rest_template.docx'))
,
                'numeratorId' => 1,
                'region' => 'ru',
                'users' => ['UA'],
                'providers' =>
['Bitrix\\DocumentGenerator\\DataProvider\\R
```

```
est'],
        ],
];

$url = $webHookUrl.$prefix.'.template.add/';
```

Ответ

```
[result] => Array
    (
        [template] => Array
            (
                [id] => 203
                [name] => Rest Template
                [region] => ru
                [code] =>
                [download] =>
http://mycrm.bitrix24.com/bitrix/services/ma
in/ajax.php?
action=documentgenerator.template.download&i
d=203&ts=1539173306
                [active] => Y
                [moduleId] => rest
                [numeratorId] => 1
                [withStamps] => N
                [providers] => Array
                    (

[bitrix\documentgenerator\dataprovider\rest]
=>
bitrix\documentgenerator\dataprovider\rest
                    )

                [users] => Array
                    (
                        [UA] => UA
```

```
                    )

                [isDeleted] => N
                [sort] => 500
                [createTime] => 2018-10-
10T14:08:26+02:00
                [updateTime] => 2018-10-
10T14:08:26+02:00
                [downloadMachine] =>
http://mycrm.bitrix24.com/rest/1/webhookkey/
documentgenerator.template.download/?
token=documentgenerator%7CYWN0a // тут
длинная ссылка
            )

    )
```

Список полей

Проверим, что поля шаблона распознались правильно. Выполним метод `documentgenerator.template.getfields`:

```
$data = [
        'id' => 203,
        'providerClassName' =>
'\\Bitrix\\DocumentGenerator\\DataProvider\\
Rest',
        'value' => 1,
];
$url =
$webHookUrl.$prefix.'.template.getfields/';
```

Ответ

```
[result] => Array
    (
        [templateFields] => Array
            (
                [DocumentNumber] => Array
                    (
                        [title] => Номер
                        [value] => 1
                        [group] => Array
                            (
                                [0] =>
Документ
                            )

                        [default] => 1
                    )
                [SomeDate] => Array
                    (
                        [value] =>
                        [default] =>
                    )
                [SomeName] => Array
                    (
                        [value] =>
                        [default] =>
                    )
                [Stamp] => Array
                    (
                        [value] =>
                        [default] =>
                    )
                [Image] => Array
                    (
                        [value] =>
                        [default] =>
                    )
                [TableItemImage] => Array
```

```
                        (
                            [value] =>
                            [default] =>
                        )
                    [TableItemName] => Array
                        (
                            [value] =>
                            [default] =>
                        )
                    [TableItemPrice] => Array
                        (
                            [value] =>
                            [default] =>
                        )
                    [TableIndex] => Array
                        (
                            [value] =>
                            [default] =>
                        )
                )
    )
```

Как видим, все поля документа на месте.

Генерация простого документа

Если в документ необходимо вставить только простые текстовые данные, то надо в метод `documentgenerator.document.add` передать только массив `values` с текстовыми значениями:

```
$data = [
        'templateId' => 203,
        'providerClassName' =>
'Bitrix\\DocumentGenerator\\DataProvider\\Re
st',
```

```
        'value' => 1,
        'values' => [
                'SomeDate' => '14.02.2018',
                'SomeName' => 'Горелкин
Владислав',
        ],
];
$url = $webHookUrl.$prefix.'.document.add/';
```

Но это самый простой случай. Теперь рассмотрим вариант, когда нам надо вставить картинки, печать, использовать модификатор для даты и имени, и заполнить таблицу несколькими значениями

Генерация документа с изображениями и печатями

Чтобы в документ передать сложные данные, не только в строковом формате, необходимо правильно сформировать параметр `fields`. Для начала, заполним изображение и печать.

```
$data = [
        'templateId' => 203,
        'providerClassName' =>
'Bitrix\\DocumentGenerator\\DataProvider\\Re
st',
        'value' => 1,
        'values' => [
                'SomeDate' => '14.02.2018',
                'SomeName' => 'Горелкин
Владислав',
                'Stamp' =>
'http://myrestapp.com/upload/stamp.png', //
внешний путь к файлу печати
                'Image' =>
'http://myrestapp.com/upload/image.jpg', //
внешний путь к файлу изображения
```

```
        ],
        'fields' => [
        'Stamp' => ['TYPE' => 'STAMP'], //
тип поля - печать
        'Image' => ['TYPE' => 'IMAGE'], //
тип поля - изображение
    ]
];
$url = $webHookUrl.$prefix.'.document.add/';
```

В массиве `fields` можно указать тип поля, по ключу `TYPE`.

- Для полей "Изображение" тип - `STAMP`
- Для полей "Печать или подпись" тип - `IMAGE`

В массиве `values` в качестве значений необходимо указать абсолютный путь к файлу. Файл будет скачан по этому адресу и вставлен в документ.

Генерация документа с модификаторами даты и имени

Про сами модификаторы можно почитать здесь.

Использование модификаторов для дат и имён через REST не обязательно - входные данные могут быть заранее отформатированы самим приложением. Тем не менее, это может быть удобно. Поэтому в REST есть возможность использовать модификаторы. Для этого необходимо передать соответствующий тип поля в `fields`. Также в `fields` по ключу `FORMAT` можно передать формат по умолчанию. Если в самом шаблоне у этого поля будет указан модификатор - будет применен модификатор шаблона.

Дата

Чтобы модификаторы работали для полей типа дата, необходимо передать их в следующем виде:

- в `values` дата должна быть передана в формате atom (как во всех rest-методах)
- в `fields` `TYPE` = `DATE`
- также в `fields` по ключу `FORMAT['format']` можно передать модификатор по умолчанию (также, как в шаблоне)

Имя

Чтобы модификаторы работали для полей типа имя необходимо передать их в следующем виде:

- в `values` имя должно быть передано в виде массива

```
[
    'NAME' => 'Игорь', // имя
    'LAST_NAME' => 'Иванов', // фамилия
    'SECOND_NAME' => 'Петрович', // отчество
    'GENDER' => 'M', // пол
]
```

По ключу `GENDER` можно указать пол в явном виде (`M` - мужской, `F` - женский). Если пол не указан, то модуль попытается определить его по отчеству. Если отчество не указано - пол не будет определен и склонение не будет работать.

- в `fields` `TYPE` = `NAME`
- в `fields` по ключу `FORMAT['format']` можно передать формат по умолчанию
- в `fields` по ключу `FORMAT['case']` можно передать падеж по умолчанию

```
$data = [
        'templateId' => 203,
        'providerClassName' =>
'Bitrix\\DocumentGenerator\\DataProvider\\Re
st',
```

```
        'value' => 1,
        'values' => [
                'SomeDate' => '2018-10-
10T14:30:18+02:00', // значение передано в
формате atom
                'SomeName' => [ // имя
передано в виде массива
                    'NAME' => 'Владислав',
                    'LAST_NAME' =>
'Горелкин',
                    'GENDER' => 'M',
                ],
        ],
        'fields' => [
        'SomeDate' => [
            'TYPE' => 'DATE',
            'FORMAT' => [
                'format' => 'd f Y H:i', //
формат вывода
            ],
        ], // тип поля - дата
        'SomeName' => [
            'TYPE' => 'NAME',
            'FORMAT' => [ // здесь можно
передать формат поля по умолчанию
                'case' => 0, // код падежа
                'format' => '#NAME#
#LAST_NAME#' // формат вывода
            ]
        ], // тип поля - имя
    ]
];
$url = $webHookUrl.$prefix.'.document.add/';
```

Генерация документа с данными в виде массива

В нашем тестовом шаблоне есть таблица, куда надо вставить картинку, название и цену товара. Модификаторы для цен нельзя использовать в ресте генератора документов, поэтому цену придётся передавать в виде сформированной строки.

Ниже рабочий пример кода, после него подробности.

```
$data = [
        'templateId' => 203,
        'providerClassName' =>
'\\Bitrix\\DocumentGenerator\\DataProvider\\
Rest',
        'value' => 1,
        'values' => [
                'Table' => [
                        [
                                'Name' =>
'Item name 1',
                                'Price' =>
'$111.23',
                                'Image' =>
'http://myrestapp.com/upload/stamp.png'
                        ],
                        [
                                'Name' =>
'Item name 2',
                                'Price' =>
'$222.34',
                                'Image' =>
'http://myrestapp.com/upload/stamp.png'
                        ],
                ],
                'TableItemName' =>
'Table.Item.Name',
                'TableItemImage' =>
'Table.Item.Image',
                'TableItemPrice' =>
'Table.Item.Price',
```

```
                'TableIndex' =>
'Table.INDEX',
        ],
        'fields' => [
                'Table' => [
                        'PROVIDER' =>
'Bitrix\\DocumentGenerator\\DataProvider\\Ar
rayDataProvider',
                        'OPTIONS' => [
                                'ITEM_NAME'
=> 'Item',

'ITEM_PROVIDER' =>
'Bitrix\\DocumentGenerator\\DataProvider\\Ha
shDataProvider',
                        ],
                ],
                'TableItemImage' => ['TYPE'
=> 'IMAGE'],
        ],
];
$url = $webHookUrl.$prefix.'.document.add/';
```

Напоминаем, что в шаблон вставлена таблица, в таблице три поля **{TableItemName}**, **{TableItemImage}**, **{TableItemPrice}**. Для начала посмотрим, как заполняется массив полей `fields`.

1. Передаем настройки поля `Table`. Такого поля в явном виде нет в шаблоне, но оно нам нужно, чтобы передать по этому ключу массив значений для таблицы.
2. Для этого поля нужно обязательно указать провайдера `fields['Table']['PROVIDER'] = 'Bitrix\\DocumentGenerator\\DataProvider\\ArrayDataProvider'`. Таким образом мы указываем, что по этому ключу придёт массив значений.
3. Надо заполнить массив настроек провайдера. `fields['Table']['OPTIONS']['ITEM_NAME'] = 'Item'`. Здесь мы передали внутренний ключ, по которому провайдер будет обращаться к элементам массива.

4. `fields['Table']['OPTIONS']['ITEM_PROVIDER'] = 'Bitrix\\DocumentGenerator\\DataProvider\\HashDataProvider'` - здесь указывается, что каждый элемент массива поля `Table` по ключу `Item` - это простой хеш.
5. Указываем, что поле `TableItemImage` является изображением - здесь всё как обычно.

Теперь про массив `values`.

1. По ключу `Table` передаем простой массив (с индексными последовательными ключами). Каждый элемент массива - это ассоциативный массив, где ключ - это правая часть поля, за исключением `Table` (название поля-массива) и `Item` (название внутреннего ключа).
2. В качестве значения для самих полей таблицы передаем цепочку получения значения из провайдера. Строится она как последовательные коды провайдеров, разделенные точкой. В нашем случае это будет `Table` - название поля-массива, откуда получаются значения. Потом точка. Дальше `Item` - название внутреннего ключа провайдера `Table`, по которому провайдер отдает элементы массива. Потом точка. Дальше идёт ключ внутреннего ассоциативного массива из элементов `Table`.
3. У провайдера `ArrayDataProvider` есть внутренняя переменная `INDEX`, которая указывает на текущий номер элемента (начиная с 1). Чтобы в поле **{TableIndex}** внутри таблицы подставился порядковый номер, было указано `values['TableIndex'] = 'Table.INDEX'`.

Если указать в качестве значения поля обычную строку, то она вставится в таблицу как есть во все строки.

Копирование таблиц

Начиная с версии **18.7.200** модуля **documentgenerator** появилась возможность создавать документы с вложенными списками, т.е. когда одно из значений списка первого уровня - это ещё один список (**Внимание:** работает только через **REST** и **PHP API**, через интерфейс такое не сделать).

Это может быть полезно, когда необходимо вставить несколько таблиц одинаковой структуры, но с разным количеством строк.

Основная идея заключается в том, что **список первого уровня** должен в качестве значений отдавать **названия полей** для внутренних списков. Все описания и значения полей должны быть переданы сразу. В массиве полей внешний список должен идти первым (т.к. поля обрабатываются в том же порядке, в котором они переданы в описании)

Этот способ можно использовать в схеме **Таблица внутри повторяющихся блоков** или **Повторяющиеся блоки внутри повторяющихся блоков**, но нельзя вставить таблицу внутри таблицы.

Пример для схемы Таблица внутри повторяющихся блоков

Пример для схемы Повторяющиеся блоки внутри повторяющихся блоков

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.document.

```
documentgenerator.document.add(templateId,
value, values = [], stampsEnabled = 0,
fields = [])
```

Метод создает новый документ на основании шаблона. В случае успешного выполнения в результате придёт структура, аналогичная методу documentgenerator.document.get() на новом документе.

Параметры

Параметр	Описание
templateId	ID шаблона.
value	Внешний идентификатор. Параметр value нужен только для интерфейса приложения, как идентификатор внешнего источника. Это строковый параметр, в него можно передать полноценный внешний код. Например, "PARTNER_APP_10_BILL_133”.
values	Набор значений полей документа. В простом случае это одномерный массив, где ключ - название поля, а значение - строка для вставки в документ.

stampsEnabled	1 (поставить), 0 (убрать) печати и подписи.
fields	Описание полей документа. Данный параметр - массив, где ключ - название поля, а значение - описание. Коды типов простых полей: ▪ **TYPE** — тип поля; ▪ **FORMAT** — формат; ▪ **PROVIDER** — провайдер; ▪ **TITLE** — заголовок; ▪ **IMAGE** — изображение; ▪ **STAMP** — печать / подпись; ▪ **DATE** — дата / время; ▪ **NAME** — имя; ▪ **PHONE** — номер телефона.

Примеры

Здесь можно посмотреть пример генерации документа.

Почему в результате document.add нет ссылки на pdf?

Конвертация в **pdf** - операция асинхронная. На момент окончания генерации документа ещё нет pdf-файла.

Если для документа очень нужен pdf, то сейчас единственный вариант - сделать повторный запрос **documentgenerator.document.get** через 20-30 секунд, чтобы

считать ссылку на pdf. Если она там не появилась - попробовать повторить.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.document.

```
documentgenerator.document.delete(id)
```

Удаляет документ. Ответ пустой.

Параметры

Параметр	Описание
id	идентификатор документа.

Примеры

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.document.

```
documentgenerator.document.enablepublicurl(id, status = 1)
```

Включает / выключает публичную ссылку на документ.

Параметры

Параметр	Описание
id	идентификатор документа.
status	1 (включить), 0 (выключить) публичную ссылку на документ.

Ответ

```
"publicUrl": "" // публичная ссылка
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.document.

```
documentgenerator.document.get(id)
```

Возвращает информацию о документе по его идентификатору.

Параметры

Параметр	Описание
id	ID документа

Ответ

```
"document": {
    "id": 1929, // id документа
    "title": "Rest Template 1", // название
    "number": "1", // номер
    "createTime": "2018-06-
05T16:04:40+02:00", // дата создания
    "updateTime": "2018-06-
05T16:04:40+02:00", // дата изменения
```

```
    "createdBy": "1", // ид пользователя,
кто создал документ
    "updatedBy": "1", // ид пользователя,
кто обновил документ
    "stampsEnabled": true // вставлены ли
печати и подписи
    "downloadUrl": "", // ссылка на
скачивание docx файла пользователем
    "downloadUrlMachine": "", // ссылка на
скачивание docx файла приложением
    "imageUrl": "", // ссылка на скачивание
картинки пользователем
    "imageUrlMachine": "", // ссылка на
скачивание картинки приложением
    "pdfUrl": "", // ссылка на скачивание
pdf пользователем
    "pdfUrlMachine": "", // ссылка на
скачивание pdf приложением
    "publicUrl": "" // публичная ссылка
(если есть)
    "isTransformationError": false, // была
ли ошибка конвертации
    "templateId": "202", // ид шаблона
    "provider":
"Bitrix\DocumentGenerator\DataProvider\Rest"
, // код провайдера
    "value": "1", // внешний идентификатор
    "values": { // значения полей
        "SomeDate": "2018-02-
21T16:33:00+03:00",
    }
}
```

documentgenerator.document.

```
documentgenerator.document.getfields(id,
values = [])
```

Возвращает список полей документа с их описанием. Возвращаемые значения абсолютно идентичны методу documentgenerator.template.getfields().

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.document.

```
documentgenerator.document.list(select =
['*'], order = [], filter = [], start = 0)
```

Возвращает список документов по фильтру.

Параметры

Параметр	Описание
select	массив полей для вывода.
order	массив для указания порядка вывода `{"id": "desc"}`.
filter	массив для фильтрации.
start	offset для постраничной навигации.

Пример фильтра

```
"filter": {
  "value": 5
}
```

В результате будет список документов, сформированный для внешнего идентификатора = 5.

Ответ

```
"documents": [
    "0": {
        "id": 1929, // id документа
        "title": "Rest Template 1", //
название
        "number": "1", // номер
        "createTime": "2018-06-
05T16:04:40+02:00", // дата создания
        "updateTime": "2018-06-
05T16:04:40+02:00", // дата изменения
        "stampsEnabled": true // вставлены
ли печати и подписи
        "downloadUrl": "", // ссылка на
скачивание docx файла пользователем
        "downloadUrlMachine": "", // ссылка
на скачивание docx файла приложением
        "imageUrl": "", // ссылка на
скачивание картинки пользователем
        "imageUrlMachine": "", // ссылка на
скачивание картинки приложением
        "pdfUrl": "", // ссылка на
скачивание pdf пользователем
        "pdfUrlMachine": "", // ссылка на
скачивание pdf приложением
```

```
        "publicUrl": "" // публичная ссылка
(если есть)
        "isTransformationError": false, //
была ли ошибка конвертации
        "templateId": "202", // ид шаблона
        "provider":
"Bitrix\DocumentGenerator\DataProvider\Rest"
, // код провайдера
        "value": "1", // внешний
идентификатор
        "values": { // значения полей
            "SomeDate": "2018-02-
21T16:33:00+03:00",
        }
    }
]
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.document.

```
documentgenerator.document.update(id,
values, stampsEnabled = 1, fields = [])
```

Обновляет существующий документ с новыми значениями. Надо учесть, что обновление документа на удалённом шаблоне невозможно. Работает аналогично documentgenerator.document.add().

Параметры

Параметр	Описание
id	идентификатор документа.
values	массив новых значений полей документа.
stampsEnabled	1 (поставить), 0 (убрать) печати и подписи.
fields	настройки полей.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.template.a

```
documentgenerator.template.add(fields)
```

Метод добавляет новый шаблон. Возвращает те же данные, что и при вызове documentgenerator.template.get() на новом шаблоне.

Контент файла (параметр `file`) можно передать двумя способами:

- В POST-запросе закодированным в *base64* (`fields[file]`);
- Без кодировки в *multipart/form-data* (просто `file`);

Т.к. интерфейс должен быть реализован самостоятельно, то настройки видимости (параметр `fields[users]`) нужны только самому приложению. Аналогично индекс сортировки (параметр `fields[sort]`) и активность (параметр `fields[active]`).

Все созданные через этот метод шаблоны привязаны к модулю rest и единственному провайдеру `\Bitrix\DocumentGenerator\DataProvider\Rest`.

Параметры

Параметр	Описание
fields	массив полей шаблона, среди которых: ▪ `fields[name]` - название шаблона (обязательное).

- `fields[file]` - контент файла, закодированный в *base64* (обязательное). Как альтернативу, контент файла можно передать в *multipart / form-data*. В этом случае его не надо кодировать в *base64*.
- `fields[code]` - символьный код шаблона.
- `fields[numeratorId]` - идентификатор нумератора (обязательное).
- `fields[region]` - страна (обязательное).
- `fields[users]` - массив видимости. По умолчанию пусто.
- `fields[active]` - Y/N флаг активности. По умолчанию Y.
- `fields[withStamps]` - Y/N ставить печати и подписи. По умолчанию N
- `fields[sort]` - индекс сортировки.

documentgenerator.template.d

```
documentgenerator.template.delete(id)
```

Если есть хотя бы один документ, созданный по этому шаблону, то он не удаляется на самом деле. Запись остается для сохранения привязки документа к модулю. При получении списка шаблонов их можно отфильтровать по параметру `isDeleted=Y`.

Если нет ни одного документа, то запись о шаблоне удаляется полностью.

Параметры

Параметр	Описание
id	идентификатор шаблона

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.template.g

```
documentgenerator.template.get(id)
```

Возвращает информацию о шаблоне по его идентификатору.

Параметры

Параметр	Описание
id	ID шаблона

Ответ

```
"template": {
    "id": "202", // id шаблона
    "name": "Rest Template", // название
    "region": "ru", // страна
    "code": "", // код
    "download": '', // ссылка на скачивание
для пользователя
    "downloadMachine": '', // ссылка на
скачивание для приложения
    "active": "Y", // активность
```

```
    "moduleId": "rest", // ид модуля
    "numeratorId": "20", // ид нумератора
    "withStamps": "N", // ставить печати по
умолчанию
    "isDeleted": "N" // удален или нет
    "sort": "100", // сортировка
    "createTime": "2018-06-
05T13:07:12+02:00"
    "updateTime": "2018-09-
06T14:26:24+02:00"
    "providers": [ // провайдеры

"bitrix\documentgenerator\dataprovider\rest"
:
"bitrix\documentgenerator\dataprovider\rest"
    ],
    "users" [ // привязанные пользователи
      "0": "UA"
    ]
}
```

documentgenerator.template.g

```
documentgenerator.template.getfields(id,
entityTypeId, entityId, values = [])
```

На вход получает идентификатор шаблона и возвращает набор полей шаблона с их описанием.

Параметры

Параметр	Описание
id	ID шаблона

Пример

```
"templateFields": {
    "DocumentNumber": {
        "title": "Номер" // заголовок
        "value": "22" // значение
        "group": [ // иерархия
            0: "Документ"
        ],
        "default": "22" // значение по
```

```
умолчанию
    },
    "SomeDate": {
        "value": "",
        "default": ""
    },
    "SomeName": {
        "value": "",
        "default": ""
    }
}
```

documentgenerator.template.li

Описание и пример

```
documentgenerator.template.list(select =
['*'], filter = [], order = [], start = 0)
```

Возвращает список шаблонов по фильтру.

Пример

```
templates: {
   202: {
        "id": "202",
        "active": "Y",
        "name": "Rest Template",
        "code": "",
        "region": "ru",
        "sort": "100",
        "createTime": "2018-06-
05T13:07:12+02:00",
        "updateTime": "2018-09-
06T14:26:24+02:00",
        "moduleId": "rest",
        "numeratorId": "20",
```

```
        "withStamps": "N",
        "isDeleted": "N",
        "download": "",
        "downloadMachine": "",
        "providers": [

"bitrix\documentgenerator\dataprovider\rest"
:
"bitrix\documentgenerator\dataprovider\rest"
        ],
        "users" [
          "0": "UA"
        ]
    }
}
```

Параметры

Параметр	Описание
select	массив полей для вывода. По умолчанию выводит все поля шаблона, кроме *users* и *providers*. Чтобы они появились, надо добавить дополнительно. Например, `['*', 'providers', 'users']`.
order	массив для указания порядка вывода `{"id": "desc"}`.
filter	массив для фильтрации.
start	offset для постраничной навигации.

Примеры фильтра

```
filter: {
  "numeratorId": "20",
  "region": "ru",
  "active": "Y"
}
```

По умолчанию фильтр имеет следующие значения:

```
filter: {
  "moduleId": "rest", // изменить нельзя,
метод всегда вернет шаблоны только для реста
  "isDeleted": "N" // если нужен список
шаблонов без учета isDeleted, необходимо
передать "@isDeleted": ["Y", "N"]
}
```

Метод вернет список шаблонов с их полями.

documentgenerator.template.u

```
documentgenerator.template.update(id,
fields)
```

Метод обновляет существующий шаблон. Возвращает те же данные, что и при вызове documentgenerator.template.get().

Параметры

Параметр	Описание
id	идентификатор шаблона.
fields	массив полей. Аналогично методу documentgenerator.template.add, только здесь все поля необязательные.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.numerator

```
documentgenerator.numerator.add(fields)
```

Метод добавляет новый нумератор. Возвращает результат, идентичный documentgenerator.numerator.get().

Параметры

Параметр	Описание
name	Имя
template	Шаблон
settings	Настройки генераторов

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.numerator

```
documentgenerator.numerator.delete(id)
```

Метод удаляет нумератор.

Удалить можно только те нумераторы, которые были созданы через documentgenerator.numerator.add().

Ответ пустой.

Параметры

Параметр	Описание
id	ID нумератора

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.numerator

```
documentgenerator.numerator.get(id)
```

Метод возвращает информацию о нумераторе по его идентификатору.

Параметры

Параметр	Описание
id	ID нумератора.

Ответ

```
"numerator": {
    "id": "202", // id шаблона
    "name": "Rest Template", // название
    "template": "{NUMBER}", // шаблон
    "settings": { // настройки генераторов

"Bitrix_Main_Numerator_Generator_SequentNumb
```

```
erGenerator": {
            "start":  20,
            "step":  5,
            "periodicBy": '',
            "timezone": '',
            "isDirectNumeration": ''
        }
    },
}
```

documentgenerator.numerator

```
documentgenerator.numerator.list(start = 0)
```

Метод возвращает список нумераторов.

Параметры

Параметр	Описание
start	offset при запросе, для постраничной навигации.

Ответ

```
"numerators": [
    0: {
        "id": "202", // id шаблона
        "name": "Rest Template", // название
        "template": "{NUMBER}", // шаблон
        "settings": { // настройки
генераторов
```

```
"Bitrix_Main_Numerator_Generator_SequentNumb
erGenerator": {
                "start":  20,
                "step":  5,
                "periodicBy": '',
                "timezone": '',
                "isDirectNumeration": ''
            }
        }
    }
]
```

documentgenerator.numerator

```
documentgenerator.numerator.update(id,
fields)
```

Метод обновляет существующий нумератор с новыми значениями.

Обновить можно только те нумераторы, которые были созданы через documentgenerator.numerator.add().

Возвращает результат, идентичный documentgenerator.numerator.get().

Параметры

Параметр	Описание
id	ID нумератора
ID нумератора	Массив, аналогичный documentgenerator.numerator.add(), только все поля необязательны.

documentgenerator.region.get

```
documentgenerator.region.get(id)
```

Возвращает информацию о регионе по его идентификатору.

Параметры

Параметр	Описание
id	ID страны

Ответ

```
"region": {
    "id": "1",
    "title": "Турция",
    "languageId": "",
    "formatDate": "YYYY-MM-DD",
    "formatDatetime": "YYYY-MM-DD HH:MI:SS",
    "formatName": "#LAST_NAME# #NAME#
#SECOND_NAME#",
    "phrases": {
        "TAX_INCLUDED": "НДС включен в
```

```
цену",
        "TAX_NOT_INCLUDED": "НДС не включен
в цену",
        "TAX_INCLUDED_NOT_VAT": "Налог (не
НДС) включен в цену",
        "TAX_NOT_INCLUDED_NOT_VAT": "Налог
(не НДС) не включен в цену",
        "UF_TYPE_BOOLEAN_YES": "Да",
        "UF_TYPE_BOOLEAN_NO": "Нет",
    }
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.region.list

```
documentgenerator.region.list()
```

Возвращает список регионов, как установленных по умолчанию, так и пользовательских.

Параметры

Без параметров

Ответ

```
"regions": {
   "uk":{ // предустановленный регион
       "code": "uk",
       "title": "Великобритания",
       "languageId": "en",
   },
   "1":{ // пользовательский регион
       "id": "1",
       "title": "Турция asdf",
       "languageId":" ",
       "formatDate": "YYYY-MM-DD",
       "formatDatetime": "YYYY-MM-DD
```

```
HH:MI:SS",
        "formatName": "#LAST_NAME# #NAME#
#SECOND_NAME#",
    }
}
```

documentgenerator.region.dele

```
documentgenerator.region.delete(id)
```

Удаляет регион.

Параметры

Параметр	Описание
id	ID региона

© «Битрикс», 2001-2008, «1С-

documentgenerator.region.add

```
documentgenerator.region.add(fields)
```

Добавляет новый регион. Возвращает те же данные, что и при вызове documentgenerator.region.get() на новом регионе.

Параметры

Параметр	Описание
fields	Массив полей страны, среди которых: ▪ `title` - название страны (обязательное) ▪ `languageId` - двухбуквенный идентификатор языка ▪ `formatDate` - формат даты (обязательное) ▪ `formatDatetime` - формат даты времени (обязательное) ▪ `formatName` - формат имени (обязательное)

Ответ

```
"region": {
    "id": "1",
    "title": "Турция",
    "languageId": "",
    "formatDate": "YYYY-MM-DD",
    "formatDatetime": "YYYY-MM-DD HH:MI:SS",
    "formatName": "#LAST_NAME# #NAME#
#SECOND_NAME#",
    "phrases": {
        "TAX_INCLUDED": "НДС включен в
цену",
        "TAX_NOT_INCLUDED": "НДС не включен
в цену",
        "TAX_INCLUDED_NOT_VAT": "Налог (не
НДС) включен в цену",
        "TAX_NOT_INCLUDED_NOT_VAT": "Налог
(не НДС) не включен в цену",
        "UF_TYPE_BOOLEAN_YES": "Да",
        "UF_TYPE_BOOLEAN_NO": "Нет",
    }
}
```

documentgenerator.region.upd

```
documentgenerator.region.update(id, fields)
```

Метод обновляет существующую страну. Возвращает те же данные, что и при вызове documentgenerator.region.get().

Параметры

Параметр	Описание
id	ID региона
fields	Массив полей. Аналогично методу documentgenerator.region.add, только здесь все поля необязательные.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.role.get

```
documentgenerator.role.get(id)
```

Отдает информацию о роли и её правах доступа.

Параметры

Параметр	Описание
id	Идентификатор роли

Ответ

```
"role": {
    "id": "1",
    "name": "Администратор",
    "code": "ADMIN",
    "permissions": {
        "settings": {
            "modify" : "X",
        },
        "templates": {
```

```
            "modify" : "X",
        },
        "documents": {
            "modify" : "X",
            "view" : "X",
        },
    }
}
```

documentgenerator.role.list

```
documentgenerator.role.list(start = 0)
```

Метод вернет список ролей без их прав доступа.

Параметры

Параметр	Описание
start	Сдвиг для постраничной навигации

Ответ

```
"roles": {
   [
       "id": "1",
       "name": "Администратор",
       "code": "ADMIN",
   ],
   [
      "id": "2",
      "name": "Менеджер",
```

```
        "code": "MANAGER",
    ],
}
```

documentgenerator.role.delete

```
documentgenerator.role.delete(id)
```

Удаляет роль.

Параметры

Параметр	Описание
id	ID роли

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

documentgenerator.role.add

```
documentgenerator.role.add(fields)
```

Метод добавит новую роль. Вернет те же данные, что и при вызове на новой роли documentgenerator.role.get() на новом регионе.

Параметры

Параметр	Описание

fields	Массив полей роли, среди которых:
	▪ `name` - название роли (обязательное)
	▪ `code` - код роли
	▪ `permissions` - разрешения роли. Это массив следующего вида:
	`"permissions": {`
	`  "settings": {`
	`  "modify" : "X",`
	`  },`
	`  "templates": {`
	`  "modify" : "X",`
	`  },`
	`  "documents": {`
	`  "modify" : "X",`
	`  "view" : "X",`
	`  },`
	`}`

Первый ключ - сущность, второй - действие, значение - уровень разрешений. Если передать пустой массив, то у роли будут отсутствовать какие-либо разрешения. Есть следующие уровни: пустое значение - нет разрешения, `A` - своё, `D` - своё и коллег по отделу, `X` - разрешено всё.

Уровни `A` и `D` имеют значение только для `permissions[templates][modify]`.

documentgenerator.role.updat

```
documentgenerator.role.update(id, fields)
```

Метод добавит новую роль. Вернет те же данные, что и при вызове на новой роли documentgenerator.role.get() на новой роли.

Параметры

Параметр	Описание
id	ID роли.
fields	Массив полей. Аналогично методу documentgenerator.role.add, только здесь все поля необязательные.

© «Битрикс», 2001-2008, «1С-

documentgenerator.role.fillacc

```
documentgenerator.role.fillaccesses(accesses
= [])
```

Метод установит набор ролей и их привязок.

Внимание! Данный метод не обновляет существующие привязки. Он всегда перезаписывает полную карту.

Параметры

Параметр	Описание
accesses	Массив привязок ролей. ▪ `accesses[roleId]` - идентификатор роли. ▪ `accesses[accessCode]` - символьный код для привязки.

Пример

```
"accesses": [
    {
        "roleId": "1",
        "accessCode" : "U1", // привязать к
пользователю с ИД 1
    },
    {
        "roleId": "2",
        "accessCode" : "D1", // привязать к
отделу с ИД 1
    }
]
```

disk.version.get

Возвращает версию по идентификатору

Пример ответа:

```
    "result": {
        "ID": "2414", //идентификатор
        "CREATED_BY": "1", //идентификатор
пользователя, который создал версию
        "CREATE_TIME": "2017-05-
22T16:32:32+02:00", //время создания
        "DOWNLOAD_URL":
"https://test.ivandivan/rest/.../download/?
token=...", //ссылка на скачивание контента
        "GLOBAL_CONTENT_VERSION": "1", //
инкрементальный счетчик версии контента
относительно файла
        "NAME": "Screenshot from 2017-05-19
09-13-02.png", //название файла на момент
создания версии
        "OBJECT_ID": "8933", //идентификатор
файла, к которому принадлежит версия
        "SIZE": "39078" //размер версии в
байтах
    }
```

Параметры

Параметр	Описание	С версии
id	Идентификатор версии	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

disk.folder.addsubfolder

Описание

```
disk.folder.addsubfolder
```

Создает дочернюю папку.

В ответе та же структура, как и в disk.folder.get.

Пример ответа:


```
"result":{
    "ID": "13",
    "NAME": "New sub folder",
    "CODE": null,
    "STORAGE_ID": "4",
    "TYPE": "folder",
    "PARENT_ID": "8",
    "DELETED_TYPE": "0",
    "CREATE_TIME": "2015-04-
24T12:39:35+03:00",
    "UPDATE_TIME": "2015-04-
24T12:39:35+03:00",
    "DELETE_TIME": null,
    "CREATED_BY": "1",
    "UPDATED_BY": "1",
```

```
    "DELETED_BY": "0",
    "DETAIL_URL":
"https://test.bitrix24.ru/workgroups/group/3
/disk/path/New/"
}
```

Параметры

Параметр	Описание
id	Идентификатор папки.
data	Массив, описывающий папку. Обязательное поле NAME - имя новой папки.

Пример

```
BX24.callMethod(
                "disk.folder.addsubfolder",
                {
                        id: 8,
                        data: {
                                NAME: 'New
sub folder'
                        }
                },
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else
```

```
console.dir(result.data());
                }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

disk.folder.copyto

```
disk.folder.copyto
```

Копирует папку в указанную папку.

В ответе та же структура, как и в disk.folder.get.

Параметры

Параметр	Описание
id	Идентификатор папки.
targetFolderId	Идентификатор папки.

Пример

```
BX24.callMethod(
                "disk.folder.copyto",
                {
                        id: 8,
                        targetFolderId:
22081990
                },
```

```
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.folder.deletetree

```
disk.folder.deletetree
```

Уничтожает папку и всё её дочерние элементы навсегда.

В ответе "result": true - успешное уничтожение папки.

Параметры

Параметр	Описание
id	Идентификатор папки.

Пример

```
BX24.callMethod(
                "disk.folder.deletetree",
                {
                        id: 8
                },
                function (result)
                {
                        if (result.error())
```

```
console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.folder.get

Описание

```
disk.folder.get
```

Возвращает папку по идентификатору.

Пример ответа:

```
"result": {
    "ID": "8", //идентификатор
    "NAME": "newfolder", //название папки
    "CODE": null, //символьный код
    "STORAGE_ID": "4", //идентификатор
хранилища
    "TYPE": "folder",
    "PARENT_ID": "12",  //идентификатор
родительской папки
    "DELETED_TYPE": "0", //маркер удаления
    "CREATE_TIME": "2015-04-
24T10:41:51+03:00", //время создания
    "UPDATE_TIME": "2015-04-
24T15:52:43+03:00", //время изменения
    "DELETE_TIME": null, //время перемещения
в корзину
    "CREATED_BY": "1", //идентификатор
```

```
пользователя, который создал файл
    "UPDATED_BY": "1", //идентификатор
пользователя, который изменил файл
    "DELETED_BY": "0", //идентификатор
пользователя, который переместил в корзину
файл
    "DETAIL_URL":
"https://test.bitrix24.ru/workgroups/group/3
/disk/path/newfolder"  //ссылка на просмотр
списка файлов папки
}
```

Параметры

Параметр	Описание
id	Идентификатор папки.

Пример

```
BX24.callMethod(
                "disk.folder.get",
                {
                        id: 8
                },
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else
```

```
console.dir(result.data());
                    }
);
```

disk.folder.getchildren

Описание

```
disk.folder.getchildren
```

Возвращает список файлов и папок, которые находятся непосредственно в папке.

В ответе массив объектов, структура которых аналогична disk.folder.get, disk.file.get.

Пример ответа:


```
"result": [
{
    "ID": "13",
    "NAME": "near",
    "CODE": null,
    "STORAGE_ID": "4",
    "TYPE": "folder",
    "PARENT_ID": "8",
    "DELETED_TYPE": "0",
    "CREATE_TIME": "2015-04-
24T12:39:35+03:00",
    "UPDATE_TIME": "2015-04-
24T12:39:35+03:00",
```

```
    "DELETE_TIME": null,
    "CREATED_BY": "1",
    "UPDATED_BY": "1",
    "DELETED_BY": "0",
    "DETAIL_URL":
"https://test.bitrix24.ru/workgroups/group/3
/disk/path/near/"
},
{
    "ID": "10",
    "NAME": "2511.jpg",
    "CODE": null,
    "STORAGE_ID": "4",
    "TYPE": "file",
    "PARENT_ID": "8",
    "DELETED_TYPE": "0",
    "CREATE_TIME": "2015-04-
24T10:41:51+03:00",
    "UPDATE_TIME": "2015-04-
24T15:52:43+03:00",
    "DELETE_TIME": null,
    "CREATED_BY": "1",
    "UPDATED_BY": "1",
    "DELETED_BY": "0",
    "DOWNLOAD_URL":
"https://test.bitrix24.ru/disk/downloadFile/
10/?&ncc=1&filename=2511.jpg&auth=******",
    "DETAIL_URL":
"https://test.bitrix24.ru/workgroups/group/3
/disk/file/2511.jpg"
},
{
    "ID": "11",
    "NAME": "549x700.png",
    "CODE": null,
    "STORAGE_ID": "4",
    "TYPE": "file",
```

```
    "PARENT_ID": "8",
    "DELETED_TYPE": "0",
    "CREATE_TIME": "2015-04-
24T10:58:49+03:00",
    "UPDATE_TIME": "2015-04-
24T12:01:32+03:00",
    "DELETE_TIME": null,
    "CREATED_BY": "1",
    "UPDATED_BY": "1",
    "DELETED_BY": "0",
    "DOWNLOAD_URL":
"https://test.bitrix24.ru/disk/downloadFile/
11/?
&ncc=1&filename=549x700.png&auth=******",
    "DETAIL_URL":
"https://test.bitrix24.ru/workgroups/group/3
/disk/file/549x700.png"
}]
```


Параметры

Параметр	Описание
id	Идентификатор папки.
filter	Необязательный параметр. Поддерживает фильтрацию по полям, которые указаны в disk.folder.getfields как USE_IN_FILTER: true.

См. также описание списочных методов.

Пример

```
BX24.callMethod(
                "disk.folder.getchildren",
                {
                        id: 8,
                        filter: {
                                CREATED_BY:
1
                        }
                },
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.folder.getExternalLink

```
disk.folder.getExternalLink
```

Метод возвращает публичную ссылку по идентификатору папки.

Метод	Описание	С версии
id	Идентификатор папки.	

Пример

```
BX24.callMethod(
        "disk.folder.getExternalLink",
       {
           id: 10
       },
       function (result)
       {
           if (result.error())

console.error(result.error());
           else
               console.dir(result.data());
```

```
            }
);
```

disk.folder.getfields

Описание

```
disk.folder.getfields
```

Возвращает описание полей файла.

- **TYPE** - тип поля;
- **USE_IN_FILTER** - возможность использовать поле при фильтрации выборки;
- **USE_IN_SHOW** - доступно ли это поле при получении ответа.

Параметры

Без параметров.

Пример ответа:

```
"result": {
     "ID": {
"TYPE": "integer",
"USE_IN_FILTER": true,
"USE_IN_SHOW": true
},
"NAME": {
```

```
"TYPE": "string",
"USE_IN_FILTER": true,
"USE_IN_SHOW": true
},
"TYPE": {
"TYPE": "enum",
"USE_IN_FILTER": true,
"USE_IN_SHOW": true
},
"CODE": {
"TYPE": "string",
"USE_IN_FILTER": true,
"USE_IN_SHOW": true
},
"STORAGE_ID": {
"TYPE": "integer",
"USE_IN_FILTER": true,
"USE_IN_SHOW": true
},
"REAL_OBJECT_ID": {
"TYPE": "integer",
"USE_IN_FILTER": false,
"USE_IN_SHOW": true
},
"PARENT_ID": {
"TYPE": "integer",
"USE_IN_FILTER": true,
"USE_IN_SHOW": true
},
"CREATE_TIME": {
"TYPE": "datetime",
"USE_IN_FILTER": true,
"USE_IN_SHOW": true
},
"UPDATE_TIME": {
"TYPE": "datetime",
"USE_IN_FILTER": true,
```

```
"USE_IN_SHOW": true
},
"DELETE_TIME": {
"TYPE": "datetime",
"USE_IN_FILTER": true,
"USE_IN_SHOW": true
},
"CREATED_BY": {
"TYPE": "integer",
"USE_IN_FILTER": false,
"USE_IN_SHOW": true
},
"UPDATED_BY": {
"TYPE": "integer",
"USE_IN_FILTER": false,
"USE_IN_SHOW": true
},
"DELETED_BY": {
"TYPE": "integer",
"USE_IN_FILTER": false,
"USE_IN_SHOW": true
},
"DELETED_TYPE": {
"TYPE": "enum",
"USE_IN_FILTER": true,
"USE_IN_SHOW": true
}
}
```

Пример

```
BX24.callMethod(
                "disk.folder.getfields",
                {},
```

```
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.folder.markdeleted

```
disk.folder.markdeleted
```

Перемещает папку в корзину.

В ответе та же структура, как и в disk.folder.get.

Параметры

Параметр	Описание
id	Идентификатор папки.

Пример

```
BX24.callMethod(
                "disk.folder.markdeleted",
                {
                        id: 8
                },
                function (result)
                {
                        if (result.error())
```

```
console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.folder.moveto

```
disk.folder.moveto
```

Перемещает папку в указанную папку.

В ответе та же структура, как и в disk.folder.get.

Параметры

Параметр	Описание
id	Идентификатор папки.
targetFolderId	Идентификатор папки.

Пример

```
BX24.callMethod(
                "disk.folder.moveto",
                {
                        id: 8,
                        targetFolderId:
22081990
                },
```

```
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.folder.rename

```
disk.folder.rename
```

Переименовывает папку.

В ответе та же структура, как и в disk.folder.get.

Параметры

Параметр	Описание
id	Идентификатор папки.
newName	Новое имя.

Пример

```
BX24.callMethod(
                "disk.folder.rename",
                {
                        id: 8,
                        newName: 'New name'
                },
                function (result)
```

```
                    {
                            if (result.error())

console.error(result.error());
                            else

console.dir(result.data());
                    }
);
```

disk.folder.restore

```
disk.folder.restore
```

Восстанавливает папку из корзины.

В ответе та же структура, как и в disk.folder.get.

Параметры

Параметр	Описание
id	Идентификатор папки.

Пример

```
BX24.callMethod(
                "disk.folder.restore",
                {
                        id: 8
                },
                function (result)
                {
                        if (result.error())
```

```
console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.folder.uploadfile

Описание

```
disk.folder.uploadfile
```

Загружает новый файл в указанную папку.

В случае успеха возвращает структуру, аналогичную disk.file.get.

Параметры

Параметр	Описание
id	Идентификатор папки. В текущем API загружать файл по пути к папке невозможно. Необходимо обязательно вычислить ID папки.
fileContent	Аналогично 'DETAIL_PICTURE' в примере Обработка файлов.
data	Массив, описывающий файл. Обязательное поле NAME - имя нового файла. Доступно отправка файла в виде строки, закодированной в base64.
generateUniqueName	Необязательный, по умолчанию *false*. При указании *true*, для загружаемого файла будет уникализировано имя, добавлением [dw]суффикса (1), (2)

	[/dw][di]Пример: avatar (1).jpg avatar (2).jpg[/di] и т.п.
rights	Необязательный, по умолчанию пустой массив. Массив прав доступа на загружаемый файл.

Примеры

Обратите внимание, что список доступных идентификаторов `TASK_ID` для установки прав можно получить rest-методом disk.rights.getTasks:

```
BX24.callMethod(
        "disk.folder.uploadfile",
        {
            id: 4,
            data: {
                NAME: "avatar.jpg"
            },
            fileContent:
document.getElementById('test_file_input'),
                        generateUniqueName:
true,
                        rights: [
                                {

TASK_ID: 42,

ACCESS_CODE: 'U35' //доступ для пользователя
с ID=35, для получения названия типа доступа
можно воспользоваться https://dev.1c-
bitrix.ru/rest_help/general/access_name.php
                },
```

```
                                        {

TASK_ID: 38,

ACCESS_CODE: 'U2' //доступ для пользователя
с ID=35, для получения названия типа доступа
можно воспользоваться https://dev.1c-
bitrix.ru/rest_help/general/access_name.php
                    }
            ]
        },
        function (result)
        {
            if (result.error())

console.error(result.error());
            else
                console.dir(result.data());
        }
);
```

Пример прямой загрузки файла на Диск

1. Первым делом вызываем **/rest/disk.folder.uploadFile** и передаем методу только ID папки:

```
disk.folder.uploadFile?
auth=n2423m863oil59f99c9g0bm4918l5erz&id=289
```

2. В ответ получаем параметр **UploadUrl** и параметр **field**:

```
"result": {
        "field": "file",
        "uploadUrl":
"http://b24.sigurd.bx/rest/upload.json?
auth=n2423m863oil59f99c9g0bm4918l5erz&token=disk%7CaWQ9
Mjg5Jl89QkYzazEzaXNnUjNHcVZQcDJZaGxGRmI4TGhXOG5EZXQ%3D%
7CInVwbG9hZHxkaXNrfGFXUTlNamc1Smw4OVFrWXphekV6YVhOblVqT
khjV
lpRY0RKWmFHeEdSbUk0VEdoWE9HNUVaWFE9fG4yNDIzbTg2M29pbDU5
Zjk5YzlnMGJtNDkxOGw1ZXJ6Ig%3D%3D.Aga709nyY0%2BrFiv3laHj
```

```
fg6XuOO5JT6ttjU%2F53ifphM%3D"
    }
```

3. На полученный **UploadUrl** посылаем POST-запрос в **multipart/form-data**, в котором передаем файл в поле с именем, полученном в параметре **field**:

```
http --form POST
"http://b24.sigurd.bx/rest/upload.json?
auth=n2423m863oil59f99c9g0bm4918l5erz&token=disk%7CaWQ9
Mjg5Jl89QkYzazEzaXNnUjNHcVZQcDJZaGxGRmI4TGhXOG5EZXQ%3D%
7CInVwbG9hZHxkaXNrfGFXUTlNamc1Smw4OVFrWXp
hekV6YVhOblVqTkhjVlpRY0RKWmFHeEdSbUk0VEdoWE9HNUVaWFE9fG
4yNDIzbTg2M29pbDU5Zjk5YzlnMGJtNDkxOGw1ZXJ6Ig%3D%3D.Aga7
09nyY0%2BrFiv3laHjfg6XuOO5JT6ttjU%2F53ifphM%3D"
file@~/somelongfile.log
```

4. В ответ получаем данные о загруженном файле:

```
"result": {
        "CODE": null,
        "CREATED_BY": "1",
        "CREATE_TIME": "2016-03-30T14:30:41+02:00",
        "DELETED_BY": null,
        "DELETED_TYPE": 0,
        "DELETE_TIME": null,
        "DETAIL_URL":
"http://b24.sigurd.bx/company/personal/user/1/disk/file
/Тестируем REST/somelongfile.log",
        "DOWNLOAD_URL":
"http://b24.sigurd.bx/rest/download.json?
auth=n2423m863oil59f99c9g0bm4918l5erz&token=disk%7CaWQ9
MjkwJl89ZTI4MG9TcDZCQno2MDAwVmV3cnRkbWxLM2hLN0JweEs%3D%
7CImRvd25sb2FkfGRpc2t8YVdROU1qa3dKbDg5WlRJNE1HOVRjRFpD
UW5vMk1EQXdWbVYzY25Sa2JXeExNMmhMTjBKd2VFcz18bjI0MjNtODY
zb2lsNTlmOTljOWcwYm00OTE4bDVlcnoi.QlpUpx4mG9sxeyMyholPf
dgkoXgc9kK9gtbOagqSo7s%3D",
        "FILE_ID": 209,
        "GLOBAL_CONTENT_VERSION": 1,
        "ID": 290,
        "NAME": "somelongfile.log",
        "PARENT_ID": "289",
        "SIZE": "496136787",
        "STORAGE_ID": "1",
        "TYPE": "file",
        "UPDATED_BY": "1",
        "UPDATE_TIME": "2016-03-30T14:30:43+02:00"
    }
```

Как загрузить файл через *UploadUrl* на php

```
 $folderId,
                ]
        );
        if (!empty($file['result']
['uploadUrl']))
        {
                $info = pathinfo($path);
                if ($info['basename'])
                {
                        $delimiter = '------
-------' . uniqid('', true);
                        $name =
$info['basename'];
                        $mime =
mime_content_type($path);
                        $content =
file_get_contents($path);

                        $body = '--' .
$delimiter. "\r\n";
                        $body .= 'Content-
Disposition: form-data; name="file"';
                        $body .= ';
filename="' . $name . '"' . "\r\n";
                        $body .= 'Content-
Type: ' . $mime . "\r\n\r\n";
                        $body .= $content .
"\r\n";
                        $body .= "--" .
$delimiter . "--\r\n";

                        $ch = curl_init();
                        curl_setopt($ch,
CURLOPT_URL, $file['result']['uploadUrl']);
                        curl_setopt($ch,
CURLOPT_POST, 1);
                        curl_setopt($ch,
```

```
CURLOPT_RETURNTRANSFER, true);
                        curl_setopt($ch,
CURLOPT_POSTFIELDS, $body);
                        curl_setopt(
                                $ch,

CURLOPT_HTTPHEADER,
                                [

'Content-Type: multipart/form-data;
boundary=' . $delimiter,

'Content-Length: ' . strlen($body),
                                ]
                        );
                        $out =
curl_exec($ch);
                        try
                        {
                                $result =
json_decode($out, true, 512,
JSON_THROW_ON_ERROR);
                        }
                        catch (JsonException
$e)
                        {
                                $result = [

'error' => $e->getMessage(),
                                ];
                        }
                }
        }
}

echo '<pre>';
```

```
        print_r($result);
echo '</pre>';
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

disk.rights.getTasks

Метод позволяет получить список уровней доступов, которые можно использовать в назначении прав.

Возвращает доступные уровни доступа.

Поля	Описание	С версии
ID	Идентификатор уровня доступа	
NAME	Символьный код	
TITLE	Название	

Примеры

Вызов:

```
BX24.callMethod(
        "disk.rights.getTasks",
        {},
        function (result) {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
```

```
                    }
)
```

Пример ответа:

```
{
        "result": [
                {
                        "ID": "42",
                        "NAME":
"disk_access_full",
                        "TITLE": "Полный
доступ"
                },
                {
                        "ID": "40",
                        "NAME":
"disk_access_edit",
                        "TITLE":
"Редактирование"
                },
                {
                        "ID": "38",
                        "NAME":
"disk_access_read",
                        "TITLE": "Чтение"
                }
        ]
}
```

disk.attachedObject.get

```
disk.attachedObject.get
```

Возвращает информацию о прикрепленном файле через пользовательское свойство по идентификатору привязки.

Примеры

Пример ответа:

```
result: {
ID: "318",
OBJECT_ID: "13215", //идентификатор файла из
Диска
MODULE_ID: "blog", //модуль, который владеет
пользовательским свойством
ENTITY_TYPE: "blog_comment", //тип сущности
ENTITY_ID: "157", //идентификатор сущности,
к которой идет прикрепление
CREATE_TIME: "2018-10-31T10:57:35+02:00", //
время создания
CREATED_BY: "1", //идентификатор
пользователя, который создал привязку
DOWNLOAD_URL:
"https://test.bitrix24.ru/bitrix/tools/disk/
uf.php?
```

```
attachedId=318&auth%5Baplogin%5D=1&auth%5Bap
%5D=******&action=download&ncc=1",
NAME: "Test.docx", //имя файла
SIZE: "3867" //размер файла в байтах
}
```

Пример вызова:

```
BX24.callMethod(
        "disk.attachedObject.get",
        {
            id: 318
        },
        function (result)
        {
            if (result.error())

console.error(result.error());
            else
                console.dir(result.data());
        }
);
```

disk.file.copyto

```
disk.file.copyto
```

Копирует файл в указанную папку.

В ответе та же структура, как и в disk.file.get.

Параметры

Параметр	Описание
id	Идентификатор файла.
targetFolderId	Идентификатор папки.

Пример

```
BX24.callMethod(
                "disk.file.copyto",
                {
                        id: 10,
                        targetFolderId: 226
                },
                function (result)
```

```
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.file.delete

```
disk.file.delete
```

Уничтожает файл навсегда.

В ответе "result": true - успешное уничтожение файла.

Параметры

Параметр	Описание
id	Идентификатор файла.

Пример

```
BX24.callMethod(
                "disk.file.delete",
                {
                        id: 10
                },
                function (result)
                {
                        if (result.error())
```

```
console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.file.get

Описание

```
disk.file.get
```

Возвращает файл по идентификатору.

Важно. Ссылка на загрузку файла из параметра `DOWNLOAD_URL` содержит токен авторизации и предназначена для скачивания файла от имени приложения. Нельзя "раздавать" эту ссылку или использовать для публичных интерфейсов.

Пример ответа:

```
"result": {
    "ID": "10", //идентификатор
    "NAME": "2511.jpg", //название файла
    "CODE": null, //символьный код
    "STORAGE_ID": "4", //идентификатор
хранилища
    "TYPE": "file",
    "PARENT_ID": "8", //идентификатор
родительской папки
    "DELETED_TYPE": "0", //маркер удаления
    "CREATE_TIME": "2015-04-
```

```
24T10:41:51+03:00", //время создания
    "UPDATE_TIME": "2015-04-
24T15:52:43+03:00", //время изменения
    "DELETE_TIME": null, //время перемещения
в корзину
    "CREATED_BY": "1", //идентификатор
пользователя, который создал файл
    "UPDATED_BY": "1", //идентификатор
пользователя, который изменил файл
    "DELETED_BY": "0", //идентификатор
пользователя, который переместил в корзину
файл
    "DOWNLOAD_URL":
"https://test.bitrix24.ru/disk/downloadFile/
10/?&ncc=1&filename=2511.jpg&auth=******",
//возвращает url для скачивания файла
приложением
    "DETAIL_URL":
"https://test.bitrix24.ru/workgroups/group/3
/disk/file/2511.jpg"
//ссылка на страницу детальной информации о
файле
}
```

Параметры

Параметр	Описание
id	Идентификатор файла.

Пример

```
BX24.callMethod(
                "disk.file.get",
                {
                        id: 10
                },
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.file.getExternalLink

```
disk.file.getExternalLink(
   id
)
```

Метод возвращает публичную ссылку по идентификатору файла. Публичные ссылки отдают файл на скачивание только, если пользователь зашел в карточку публичной ссылки.

Пример ответа:

```
"result": "https://test.bitrix24.ru/~Fjruf2"
```

Параметры

Параметр	Описание
id	Идентификатор файла.

Пример

```
BX24.callMethod(
        "disk.file.getExternalLink",
       {
           id: 10
       },
       function (result)
       {
           if (result.error())

console.error(result.error());
           else
               console.dir(result.data());
       }
);
```

disk.file.getfields

```
disk.file.getfields
```

Возвращает описание полей файла.

- **TYPE** - тип поля;
- **USE_IN_FILTER** - возможность использовать поле при фильтрации выборки;
- **USE_IN_SHOW** - доступно ли это поле при получении ответа.

Параметры

Без параметров.

Пример

```
BX24.callMethod(
                "disk.file.getfields",
                {},
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else
```

```
console.dir(result.data());
                }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

disk.file.getVersions

Возвращает список версий файла. Версии упорядочены по убыванию даты создания. В ответе список версий, структура которых аналогична disk.version.get.

Параметры

Метод	Описание	С версии
id	Идентификатор файла	
filter	Необязательный параметр. Поддерживает фильтрацию по полям, которые указаны в *disk.version.getfields* как `USE_IN_FILTER: true`.	

См. также описание списочных методов.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

disk.file.markdeleted

```
disk.file.markdeleted
```

Перемещает файл в корзину.

В ответе та же структура, как и в disk.file.get.

Параметры

Параметр	Описание
id	Идентификатор файла.

Пример

```
BX24.callMethod(
                "disk.file.markdeleted",
                {
                        id: 10
                },
                function (result)
                {
                        if (result.error())
```

```
console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.file.moveto

```
disk.file.moveto
```

Перемещает файл в указанную папку.

В ответе та же структура, как и в disk.file.get.

Параметры

Параметр	Описание
id	Идентификатор файла.
targetFolderId	Идентификатор папки.

Пример

```
BX24.callMethod(
                "disk.file.moveto",
                {
                        id: 10,
                        targetFolderId: 226
                },
                function (result)
```

```
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.file.rename

```
disk.file.rename
```

Переименовывает файл.

В ответе та же структура, как и в disk.file.get.

Параметры

Параметр	Описание
id	Идентификатор файла.
newName	Новое имя.

Пример

```
BX24.callMethod(
                "disk.file.rename",
                {
                        id: 10,
                        newName: 'Newname
for file.png'
                },
```

```
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.file.restore

```
disk.file.restore
```

Восстанавливает файл из корзины.

В ответе та же структура, как и в disk.file.get.

Параметры

Параметр	Описание
id	Идентификатор файла.

Пример

```
BX24.callMethod(
                "disk.file.restore",
                {
                        id: 10
                },
                function (result)
                {
                        if (result.error())
```

```
console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.file.restoreFromVersion

Восстанавливает файл из конкретной версии. В ответе вернется обновленный файл, структура ответа аналогична disk.file.get

Параметры

Метод	Описание	С версии
id	идентификатор файла	
versionId	идентификатор версии	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

disk.file.uploadversion

```
disk.file.uploadversion
```

Загружает новую версию файла.

В ответе та же структура, как и в disk.file.get.

Параметры

Параметр	Описание
id	Идентификатор файла.
fileContent	Аналогично 'DETAIL_PICTURE' в примере Обработка файлов.

Пример

```
BX24.callMethod(
                "disk.file.uploadversion",
                {
                        id: 4,
                        fileContent:
document.getElementById('test_file_input')
```

```
                },
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.storage.addfolder

Описание

```
disk.storage.addfolder
```

Создает папку в корне хранилища.

Возвращаемая структура аналогична приведенной в disk.folder.get.

Пример ответа:


```
"result":{
    "ID": "13",
    "NAME": "New",
    "CODE": null,
    "STORAGE_ID": "4",
    "TYPE": "folder",
    "PARENT_ID": "8",
    "DELETED_TYPE": "0",
    "CREATE_TIME": "2015-04-
24T12:39:35+03:00",
    "UPDATE_TIME": "2015-04-
24T12:39:35+03:00",
    "DELETE_TIME": null,
    "CREATED_BY": "1",
    "UPDATED_BY": "1",
```

```
    "DELETED_BY": "0",
    "DETAIL_URL":
"https://test.bitrix24.ru/workgroups/group/3
/disk/path/New/"
}
```

Параметры

Параметр	Описание
id	Идентификатор хранилища.
data	Массив, описывающий папку. Обязательное поле NAME - имя новой папки.

Пример

```
BX24.callMethod(
                "disk.storage.addfolder",
                {
                        id: 4,
                        data: {'NAME':
'New'}
                },
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
```

```
);
```

disk.storage.get

```
disk.storage.get
```

Возвращает хранилище по идентификатору.

Пример ответа:

```
"result": {
        "ID": "2", //идентификатор
        "NAME": "Маркетинг и реклама", //
название
        "CODE": null, //символьный код
        "MODULE_ID": "disk",
        "ENTITY_TYPE": "group", //тип
сущности (см. disk.storage.gettypes)
        "ENTITY_ID": "1", //идентификатор
сущности
        "ROOT_OBJECT_ID": "2" //
идентификатор корневой папки
}
```

Параметры

Параметр	Описание
id	Идентификатор хранилища.

Пример

```
        BX24.callMethod(
                "disk.storage.get",
                {id: 2},
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.storage.getchildren

Описание

```
disk.storage.getchildren
```

Возвращает список файлов и папок, которые находятся непосредственно в корне хранилища.

В ответе массив объектов, структура которых аналогична disk.folder.get, disk.file.get.

Пример ответа:


```
"result": [
{
    "ID": "13",
    "NAME": "near",
    "CODE": null,
    "STORAGE_ID": "4",
    "TYPE": "folder",
    "PARENT_ID": "8",
    "DELETED_TYPE": "0",
    "CREATE_TIME": "2015-04-
24T12:39:35+03:00",
    "UPDATE_TIME": "2015-04-
24T12:39:35+03:00",
```

```
    "DELETE_TIME": null,
    "CREATED_BY": "1",
    "UPDATED_BY": "1",
    "DELETED_BY": "0",
    "DETAIL_URL":
"https://test.bitrix24.ru/workgroups/group/3
/disk/path/near/"
},
{
    "ID": "10",
    "NAME": "2511.jpg",
    "CODE": null,
    "STORAGE_ID": "4",
    "TYPE": "file",
    "PARENT_ID": "8",
    "DELETED_TYPE": "0",
    "CREATE_TIME": "2015-04-
24T10:41:51+03:00",
    "UPDATE_TIME": "2015-04-
24T15:52:43+03:00",
    "DELETE_TIME": null,
    "CREATED_BY": "1",
    "UPDATED_BY": "1",
    "DELETED_BY": "0",
    "DOWNLOAD_URL":
"https://test.bitrix24.ru/disk/downloadFile/
10/?&ncc=1&filename=2511.jpg&auth=******",
    "DETAIL_URL":
"https://test.bitrix24.ru/workgroups/group/3
/disk/file/2511.jpg"
},
{
    "ID": "11",
    "NAME": "549x700.png",
    "CODE": null,
    "STORAGE_ID": "4",
    "TYPE": "file",
```

```
    "PARENT_ID": "8",
    "DELETED_TYPE": "0",
    "CREATE_TIME": "2015-04-
24T10:58:49+03:00",
    "UPDATE_TIME": "2015-04-
24T12:01:32+03:00",
    "DELETE_TIME": null,
    "CREATED_BY": "1",
    "UPDATED_BY": "1",
    "DELETED_BY": "0",
    "DOWNLOAD_URL":
"https://test.bitrix24.ru/disk/downloadFile/
11/?
&ncc=1&filename=549x700.png&auth=******",
    "DETAIL_URL":
"https://test.bitrix24.ru/workgroups/group/3
/disk/file/549x700.png"
}]
```


Параметры

Параметр	Описание
id	Идентификатор хранилища.
filter	Необязательный параметр. Поддерживает фильтрацию по полям, которые указаны в disk.folder.getfields как USE_IN_FILTER: true.

См. также описание списочных методов.

Пример

```
BX24.callMethod(
                "disk.storage.getchildren",
                {
                        id: 4,
                        filter: {
                                CREATED_BY:
1
                        }
                },
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.storage.getfields

```
disk.storage.getfields
```

Возвращает описание полей хранилища.

- **TYPE** - тип поля;
- **USE_IN_FILTER** - возможность использовать поле при фильтрации выборки;
- **USE_IN_SHOW** - доступно ли это поле при получении ответа.

Параметры

Без параметров.

Пример

```
        BX24.callMethod(
                "disk.storage.getfields",
                {},
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else
```

```
console.dir(result.data());
                }
);
```

disk.storage.getforapp

```
disk.storage.getforapp
```

Возвращает описание хранилища, с которым может работать приложение для хранения своих данных (файлов и папок).

Возвращаемая структура аналогична приведенной в disk.storage.get.

Пример ответа:

```
"result": {
        "ID": "221990",
        "NAME": "bitrix.restapi",
        "CODE": null,
        "MODULE_ID": "disk",
        "ENTITY_TYPE": "restapp",
        "ENTITY_ID": "1",
        "ROOT_OBJECT_ID": "2"
}
```

Параметры

Без параметров.

Пример

```
BX24.callMethod(
                "disk.storage.getforapp",
                {},
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.storage.getlist

```
disk.storage.getlist
```

Возвращает список доступных хранилищ.

Параметры

Параметр	Описание
filter	Необязательный параметр. Поддерживает фильтрацию по полям, которые указаны в disk.storage.getfields как USE_IN_FILTER: true.

В ответе массив объектов, структура которых аналогична disk.storage.get.

См. также описание списочных методов.

Пример

```
//поиск хранилища группы с именем содержащем
"Фут"
BX24.callMethod(
                "disk.storage.getlist",
```

```
                {
                        filter: {

'ENTITY_TYPE': 'group',
                                '%NAME':
'Фут'
                        }
                },
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.storage.gettypes

```
disk.storage.gettypes
```

Возвращает список типов хранилищ.

Пример ответа:

```
"result": [
    "user", //хранилище пользователей
    "common", //хранилище общих документов
    "group" //хранилище социальных групп
]
```

Параметры

Без параметров.

Пример

```
BX24.callMethod(
                "disk.storage.gettypes",
                {},
```

```
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.storage.rename

```
disk.storage.rename
```

Переименовывает хранилище. Допустимо переименование только хранилища приложения (см. disk.storage.getforapp).

Пример ответа:

```
"result": {
        "ID": "2", //идентификатор
        "NAME": "Маркетинг и реклама", //
название
        "CODE": null, //символьный код
        "MODULE_ID": "disk",
        "ENTITY_TYPE": "group", //тип
сущности (см. disk.storage.gettypes)
        "ENTITY_ID": "1", //идентификатор
сущности
        "ROOT_OBJECT_ID": "2" //
идентификатор корневой папки
}
```

Параметры

Параметр	Описание
id	Идентификатор хранилища.
newName	Новое имя.

Пример

```
        BX24.callMethod(
                "disk.storage.rename",
                {
                        id: 2,
                        newName: 'New name
for storage'
                },
                function (result)
                {
                        if (result.error())

console.error(result.error());
                        else

console.dir(result.data());
                }
);
```

disk.storage.uploadfile

Описание

```
disk.storage.uploadfile
```

Загружает новый файл в корне хранилища.

В случае успеха возвращает структуру, аналогичную приведенной в disk.file.get.

Пример ответа:


```
"result": {
    "ID": "10",
    "NAME": "2511.jpg",
    "CODE": null,
    "STORAGE_ID": "4",
    "TYPE": "file",
    "PARENT_ID": "8",
    "DELETED_TYPE": "0",
    "CREATE_TIME": "2015-04-
24T10:41:51+03:00",
    "UPDATE_TIME": "2015-04-
24T15:52:43+03:00",
    "DELETE_TIME": null,
    "CREATED_BY": "1",
```

```
    "UPDATED_BY": "1",
    "DELETED_BY": "0",
    "DOWNLOAD_URL":
"https://test.bitrix24.ru/disk/downloadFile/
10/?&ncc=1&filename=2511.jpg&auth=******",
    "DETAIL_URL":
"https://test.bitrix24.ru/workgroups/group/3
/disk/file/2511.jpg"
}
```


Параметры

Параметр	Описание
id	Идентификатор хранилища.
fileContent	Аналогично 'DETAIL_PICTURE' в примере Обработка файлов.
data	Массив, описывающий файл. Обязательное поле NAME - имя нового файла.
generateUniqueName	Необязательный, по умолчанию *false*. При указании *true*, для загружаемого файла будет уникализировано имя, добавлением [dw]суффикса (1), (2) [/dw][di]Пример: avatar (1).jpg avatar (2).jpg[/di] и т.п.
rights	Необязательный, по умолчанию пустой массив. Массив прав доступа на загружаемый файл.

Пример

Обратите внимание, что список доступных идентификаторов `TASK_ID` для установки прав можно получить rest-методом disk.rights.getTasks.

```
BX24.callMethod(
        "disk.storage.uploadFile",
        {
            id: 4,
            data: {
                NAME: "avatar.jpg"
            },
            fileContent:
document.getElementById('test_file_input'),
                        generateUniqueName:
true,
                        rights: [
                                {

TASK_ID: 42,

ACCESS_CODE: 'U35' //доступ для пользователя
с ID=35, для получения названия типа доступа
можно воспользоваться https://dev.1c-
bitrix.ru/rest_help/general/access_name.php
                },
                                {

TASK_ID: 38,

ACCESS_CODE: 'U2' //доступ для пользователя
с ID=35, для получения названия типа доступа
можно воспользоваться https://dev.1c-
bitrix.ru/rest_help/general/access_name.php
                }
```

```
            ]
        },
        function (result)
        {
            if (result.error())

console.error(result.error());
            else
                console.dir(result.data());
        }
);
```

log.blogpost.add

Описание

Добавляет в **Живую Ленту** сообщение от имени текущего пользователя.

Запрос:

```
https://my.bitrix24.ru/rest/log.blogpost.add
.json?
POST_MESSAGE=Hello%2C%20world!&auth=d9a76e29
29b7bc1ff21aee9c0ce7e3e2
```

Ответ:

```
{"result":true}
```

Параметры

Параметр	Описание
USER_ID	ID автора записи (опционально, по умолчанию - текущий пользователь,

	другое значение доступно только администратору в коробочной версии).
POST_MESSAGE	Текст сообщения.
POST_TITLE	Заголовок сообщения (опционально).
DEST	Права на просмотр сообщения (опционально), по умолчанию - `array("UA")` - всем авторизованным пользователям. Можно использовать значения: ▪ **SG<X>** - рабочая группа, например SG1 - рабочая группа с идентификатором 2; ▪ **U<X>** - пользователь, например U45 - пользователь с идентификатором 45; ▪ **DR<X>** - отдел, включая подразделы, например DR23 - раздел с идентификатором 23; ▪ **UA** - все авторизованные пользователи. ▪ **G<X>** - группа пользователей, например G2 - группа пользователей с идентификатором 2.
SPERM	Права на просмотр сообщения (опционально), по умолчанию - `array("UA")` - всем авторизованным пользователям. Данный параметр поддерживается, но удобнее применять **DEST**.
FILES	Файлы, массив значений, описанный по правилам, приведенным тут.
IMPORTANT	По умолчанию N. Сообщение ленты публикуется как "важное".

IMPORTANT_DATE_END	Указывается значение даты/времени, до которого сообщение будет считаться важным.

Пример

```
BX.rest.callMethod('log.blogpost.add',
{POST_TITLE: 'заголовок',
POST_MESSAGE: 'текст', DEST: ['SG1', 'U2']
}).
then()
```

```
BX24.callMethod('log.blogpost.add',
{POST_MESSAGE: 'Hello, world!'}, function(r)
{
        if(!r.error())
                {
                alert('OK!');
                }
        else
                {
                throw r.error();
                }
});
```

Смотрите также

Использование метода REST API log.blogpost.add (блог разработчика)

log.blogpost.getusers.importar

Отдает массив ID пользователей, прочитавших **Важное сообщение** на портале.

Параметры функции

Параметр	Описание
POST_ID	ID поста блога, являющегося Важным сообщением.

Пример

```
BX24.callMethod(
    'log.blogpost.getusers.important',
    {
        POST_ID: 345
    },
    function(result)
    {
        console.log(result.data());
    }
);
```

Запрос:

```
https://my.bitrix24.ru/rest/log.blogpost.get
users.important.json?
POST_ID=345&auth=xxxxxxx
```

Ответ:

```
{"result":["1","2","3"]}
```

log.blogpost.get

Возвращает массив с доступными текущему пользователю сообщениями **Живой ленты**. Каждое из сообщений представляет собой массив значений полей (включая пользовательские поля).

Параметры функции

Параметр	Описание
POST_ID	Числовой ID сообщения для фильтрации (опционально).
LOG_RIGHTS	Фильтрация по получателю. Значением фильтра может быть как строка (конкретное значение прав), так и массив. Права на просмотр сообщения (опционально), по умолчанию - `array("UA")` - всем авторизованным пользователям. Можно использовать значения: ▪ **SG<X>** - рабочая группа с ID=X; ▪ **U<X>** - пользователь с ID=X; ▪ **DR<X>** - отдел с ID=X, включая подразделы; ▪ **UA** - все авторизованные пользователи. ▪ **G** - группа пользователей (например, G2).

В ответе значение поля DATE_PUBLISH отдается в формате ISO 8601.

Пример

```
BX24.callMethod('log.blogpost.get', {
POST_ID: 755 });
```

Запрос:

```
https://my.bitrix24.ru/rest/log.blogpost.get
.xml?auth=xxxxxxx
```

Ответ:


```
{"result":
[{"ID":"4","BLOG_ID":"1","PUBLISH_STATUS":"P
","TITLE":"sdfsdfsd","AUTHOR_ID":"1","ENABLE
_COMMENTS":"Y","NUM_COMMENTS":"0",
"CODE":null,"MICRO":"Y","DETAIL_TEXT":"sdfsd
fsd","DATE_PUBLISH":"21.10.2015 17:00:50",
"CATEGORY_ID":"","HAS_SOCNET_ALL":"Y","HAS_T
AGS":"N","HAS_IMAGES":"N","HAS_PROPS":"N","H
AS_COMMENT_IMAGES":null},
{"ID":"3","BLOG_ID":"1","PUBLISH_STATUS":"P"
,"TITLE":"test1","AUTHOR_ID":"1","ENABLE_COM
MENTS":"Y","NUM_COMMENTS":"0",
"CODE":null,"MICRO":"Y","DETAIL_TEXT":"test1
","DATE_PUBLISH":"31.07.2015 17:03:54",
"CATEGORY_ID":"","HAS_SOCNET_ALL":"Y","HAS_T
AGS":"N","HAS_IMAGES":"N","HAS_PROPS":"Y","H
AS_COMMENT_IMAGES":null,
```

"UF_BLOG_POST_DOC":
{"ID":"1","ENTITY_ID":"BLOG_POST","FIELD_NAM
E":"UF_BLOG_POST_DOC","USER_TYPE_ID":"file",
"XML_ID":"UF_BLOG_POST_DOC","SORT":"100","MU
LTIPLE":"Y","MANDATORY":"N","SHOW_FILTER":"N
","SHOW_IN_LIST":"N","EDIT_IN_LIST":"Y",
"IS_SEARCHABLE":"Y","SETTINGS":
{"SIZE":20,"LIST_WIDTH":0,"LIST_HEIGHT":0,"M
AX_SHOW_SIZE":0,"MAX_ALLOWED_SIZE":0,"EXTENS
IONS":[]},
"EDIT_FORM_LABEL":null,"LIST_COLUMN_LABEL":n
ull,"LIST_FILTER_LABEL":null,"ERROR_MESSAGE"
:null,"HELP_MESSAGE":null,
"USER_TYPE":
{"USER_TYPE_ID":"file","CLASS_NAME":"CUserTy
peFile",
"DESCRIPTION":"\u0424\u0430\u0439/u043b","BA
SE_TYPE":"file"},"VALUE":false,"ENTITY_VALUE
_ID":3},
"UF_GRATITUDE":
{"ID":"3","ENTITY_ID":"BLOG_POST","FIELD_NAM
E":"UF_GRATITUDE","USER_TYPE_ID":"integer",
"XML_ID":"UF_GRATITUDE","SORT":"100","MULTIP
LE":"N",
"MANDATORY":"N","SHOW_FILTER":"N","SHOW_IN_L
IST":"N","EDIT_IN_LIST":"Y","IS_SEARCHABLE":
"N",
"SETTINGS":
{"SIZE":20,"MIN_VALUE":0,"MAX_VALUE":0,"DEFA
ULT_VALUE":""},
"EDIT_FORM_LABEL":null,"LIST_COLUMN_LABEL":n
ull,"LIST_FILTER_LABEL":null,"ERROR_MESSAGE"
:null,"HELP_MESSAGE":null,
"USER_TYPE":
{"USER_TYPE_ID":"integer","CLASS_NAME":"CUse
rTypeInteger",
"DESCRIPTION":"\u0426\u0435\u043b\u043e\u043

5
\u0447\u0438\u0441\u043b\u043e","BASE_TYPE":
"int"},
"VALUE":null,"ENTITY_VALUE_ID":3},
"UF_BLOG_POST_FILE":
{"ID":"8","ENTITY_ID":"BLOG_POST","FIELD_NAM
E":"UF_BLOG_POST_FILE","USER_TYPE_ID":"disk_
file",
"XML_ID":"UF_BLOG_POST_FILE","SORT":"100","M
ULTIPLE":"Y","MANDATORY":"N","SHOW_FILTER":"
N","SHOW_IN_LIST":"N","EDIT_IN_LIST":"Y",
"IS_SEARCHABLE":"Y","SETTINGS":"IBLOCK_ID":0
,"SECTION_ID":0,"UF_TO_SAVE_ALLOW_EDIT":null
},"EDIT_FORM_LABEL":null,
"LIST_COLUMN_LABEL":null,"LIST_FILTER_LABEL"
:null,"ERROR_MESSAGE":null,"HELP_MESSAGE":nu
ll,
"USER_TYPE":
{"USER_TYPE_ID":"disk_file","CLASS_NAME":"Bi
trix\\Disk\\Uf\\FileUserType",
"DESCRIPTION":"\u0424\u0430\u0439\u043b
(\u0414\u0438\u0441\u043a)",
"BASE_TYPE":"int","TAG":["DISK FILE
ID","DOCUMENT
ID"]},"VALUE":false,"ENTITY_VALUE_ID":3},
"UF_BLOG_POST_IMPRTNT":"ID":"18","ENTITY_ID"
:"BLOG_POST","FIELD_NAME":"UF_BLOG_POST_IMPR
TNT","USER_TYPE_ID":"integer",
"XML_ID":"UF_BLOG_POST_IMPRTNT","SORT":"100"
,"MULTIPLE":"N","MANDATORY":"N","SHOW_FILTER
":"N","SHOW_IN_LIST":"Y",
"EDIT_IN_LIST":"Y","IS_SEARCHABLE":"N","SETT
INGS":
{"SIZE":20,"MIN_VALUE":0,"MAX_VALUE":0,"DEFA
ULT_VALUE":""},
"EDIT_FORM_LABEL":"\u0412\u0430\u0436\u043d\
u043e\u0435

\u0441\u043e\u043e\u0431\u0449\u0435\u043d\u
0438\u0435",
"LIST_COLUMN_LABEL":"\u0412\u0430\u0436\u043
d\u043e\u0435","LIST_FILTER_LABEL":"\u0412\u
0430\u0436\u043d\u043eu0435",
"ERROR_MESSAGE":null,"HELP_MESSAGE":null,"US
ER_TYPE":
{"USER_TYPE_ID":"integer","CLASS_NAME":"CUse
rTypeInteger",
"DESCRIPTION":"\u0426\u0435\u043b\u043e\u043
5
\u0447\u0438\u0441\u043b\u043e","BASE_TYPE":
"int"},"VALUE":"1","ENTITY_VALUE_ID":3},
"UF_BLOG_POST_VOTE":"ID":"35","ENTITY_ID":"B
LOG_POST","FIELD_NAME":"UF_BLOG_POST_VOTE","
USER_TYPE_ID":"vote",
"XML_ID":"UF_BLOG_POST_VOTE","SORT":"100","M
ULTIPLE":"N","MANDATORY":"N",
"SHOW_FILTER":"N","SHOW_IN_LIST":"Y","EDIT_I
N_LIST":"Y","IS_SEARCHABLE":"N","SETTINGS":
{"CHANNEL_ID":1,"UNIQUE":13,
"UNIQUE_IP_DELAY":
{"DELAY":"10","DELAY_TYPE":"D"},"NOTIFY":"I"
},"EDIT_FORM_LABEL":null,
"LIST_COLUMN_LABEL":null,"LIST_FILTER_LABEL"
:null,"ERROR_MESSAGE":null,"HELP_MESSAGE":nu
ll,
"USER_TYPE":
{"USER_TYPE_ID":"vote","CLASS_NAME":"CUserTy
peVote","DESCRIPTION":"\u041e\u043f\u0440\u0
43e\u0441",
"BASE_TYPE":"int"},"VALUE":null,"ENTITY_VALU
E_ID":3}},
{"ID":"2","BLOG_ID":"1","PUBLISH_STATUS":"P"
,"TITLE":"test","AUTHOR_ID":"1","ENABLE_COMM
ENTS":"Y","NUM_COMMENTS":"0",
"CODE":null,"MICRO":"Y","DETAIL_TEXT":"test"

,"DATE_PUBLISH":"31.07.2015 17:03:35",
"CATEGORY_ID":"","HAS_SOCNET_ALL":"Y","HAS_T
AGS":"N","HAS_IMAGES":"N","HAS_PROPS":"N","H
AS_COMMENT_IMAGES":null},
{"ID":"1","BLOG_ID":"1","PUBLISH_STATUS":"P"
,
"TITLE":"\u041e\u0444\u0438\u0446\u0438\u043
0\u043b\u044c\u043d\u043e
\u043e\u0442\u043a\u0440\u044b\u0442
\u043a\u043e\u0440\u043f\u043e\u0440\u0430\u
0442\u0438\u0432
\u043d\u044b\u0439
\u043f\u043e\u0440\u0442\u0430\u043b
\u043d\u0430\u0448\u0435\u0439
\u043a\u043e\u043c\u043f\u0430\u043d\u0438\u
0438",
"AUTHOR_ID":"1","ENABLE_COMMENTS":"Y","NUM_C
OMMENTS":"1","CODE":null,"MICRO":"N",
"DETAIL_TEXT":"\u0421\u0435\u0433\u043e\u043
4\u043d\u044f
\u043e\u0444\u0438\u0446\u0438\u0430\u043b\u
044c\u043d\u043e
\u0437\u0430\u0440\u0430\u0431\u043e\u0442\u
0430
\u043b
\u043a\u043e\u0440\u043f\u043e\u0440\u0430\u
0442\u0438\u0432\u043d\u044b\u0439
\u043f\u043e\u0440\u0442\u0430\u043b
\u043d\u0430\u0448\u0435\u0439 \u043a\u043e
\u043c\u043f\u0430\u043d\u0438\u0438.\n\n\",
"DATE_PUBLISH":"16.07.2015
13:12:35","CATEGORY_ID":"1,2","HAS_SOCNET_AL
L":"Y","HAS_TAGS":"Y",
"HAS_IMAGES":"N","HAS_PROPS":"N","HAS_COMMEN
T_IMAGES":"N"}],"total":4}

log.blogcomment.add

Добавляет комментарий к сообщению Живой ленты

Параметры

Параметр	Описание	Версия
$USER_ID	Автор комментария. Доступно только для администратора. По умолчанию - текущий пользователь.	
$POST_ID	ID сообщения	
$TEXT	Текст комментария	
$FILES	Файлы, массив значений, описанный по правилам, приведенным тут.	

Пример:

```
BX.rest.callMethod('log.blogcomment.add', {
    POST_ID: 10,
    TEXT: 'test comment'
}).then();
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

log.blogpost.delete

Удаляет сообщение **Живой Ленты**.

Параметры функции

Параметр	Описание
USER_ID	ID пользователя, который выполняет действие (опционально, по умолчанию - текущий пользователь, другое значение доступно только администратору в коробочной версии)
POST_ID	Числовой ID сообщения.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

log.blogpost.share

Добавляет получателей в сообщение **Живой Ленты**.

Параметры функции

Параметр	Описание
USER_ID	ID пользователя, добавляющего новых получателей (опционально, по умолчанию - текущий пользователь, другое значение доступно только администратору в коробочной версии).
POST_ID	ID сообщения **Живой ленты**.
DEST	Права на просмотр сообщения. Можно использовать значения: ▪ **SG<X>** - рабочая группа с ID=X; ▪ **U<X>** - пользователь с ID=X; ▪ **DR<X>** - отдел с ID=X, включая подразделы; ▪ **UA** - все авторизованные пользователи.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

log.blogpost.update

Изменяет сообщение **Живой Ленты**.

Параметры функции

Параметр	Описание
POST_ID	ID сообщения **Живой ленты**.
USER_ID	ID автора записи (опционально, по умолчанию - текущий пользователь, другое значение доступно только администратору в коробочной версии).
POST_MESSAGE	Текст сообщения.
POST_TITLE	Заголовок сообщения (опционально).
DEST	Права на просмотр сообщения (опционально), по умолчанию - `array("UA")` - всем авторизованным пользователям. Можно использовать значения: ▪ **SG<X>** - рабочая группа с ID=X; ▪ **U<X>** - пользователь с ID=X; ▪ **DR<X>** - отдел с ID=X, включая подразделы; ▪ **UA** - все авторизованные пользователи.
SPERM	Права на просмотр сообщения (опционально), по умолчанию - `array("UA")` - всем авторизованным пользователям.

	Данный параметр поддерживается, но удобнее применять **DEST**.
FILES	Файлы, массив значений, описанный по правилам, приведенным тут.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

События

Событие	Описание	С версии
ONLIVEFEEDPOSTADD	Прокси к событию PHP OnAfterSocNetLogAdd.	
ONLIVEFEEDPOSTUPDATE	Прокси к событию PHP OnAfterSocNetLogUpdate.	
ONLIVEFEEDPOSTDELETE	Прокси к событию PHP OnSocNetLogDelete.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Структура таблиц

Допустимые поля

Название	Описание	Чтен
TITLE	Название задачи.	+
DESCRIPTION	Описание задачи.	+
DEADLINE	Крайний срок.	+
START_DATE_PLAN	Плановая дата начала.	+
END_DATE_PLAN	Плановая дата завершения.	+
PRIORITY	Приоритет.	+
ACCOMPLICES	Соисполнители (идентификаторы пользователей).	+
ACCOMPLICE	Соисполнители (поле используется для фильтрации).	
AUDITORS	Наблюдатели (идентификаторы пользователей).	+
AUDITOR	Наблюдатели (поле используется для фильтрации).	

TAGS	Теги (при добавлении - просто массив тегов в виде текста). **CTasks::GetList()** не возвращает поля тегов. **CTaskItem::getInstance()->getTags()** возвращает массив имен тегов.	+
TAG	Теги (поле используется для фильтрации).	
ALLOW_CHANGE_DEADLINE	Флаг "Разрешить ответственному менять крайний срок".	+
TASK_CONTROL	Флаг "Принять работу после завершения задачи".	+
PARENT_ID	Идентификатор родительской задачи.	+
DEPENDS_ON	Идентификатор предыдущей задачи.	+
GROUP_ID	Идентификатор рабочей группы.	+
RESPONSIBLE_ID	Идентификатор ответственного.	+
TIME_ESTIMATE	Плановые трудозатраты.	+
ID	Идентификатор задачи. Уникален в рамках базы данных.	+
CREATED_BY	Идентификатор постановщика.	+
DESCRIPTION_IN_BBCODE	Флаг указывающий, что описание задачи хранится в BB-кодах.	+

DECLINE_REASON	Причина отклонения задачи.	+
REAL_STATUS	Истинный статус задачи, который записывается через **STATUS** (см. константы *CTasks::STATE_xxx*). Только для чтения.	+
STATUS	Мета-статус задачи. При записи можно использовать константы *CTasks::STATE_xxx*, однако, при чтении, помимо *CTasks::STATE_xxx*, в результатах можно увидеть *CTasks::METASTATE_xxx*. То есть на самом деле статус задачи может быть *CTasks::STATE_NEW*, а при чтении вернется нам *CTasks::METASTATE_EXPIRED* (для просроченной задачи). В случае если мы хотим узнать истинный статус задачи, следует читать поле **REAL_STATUS**.	+
RESPONSIBLE_NAME	Имя ответственного.	+
RESPONSIBLE_LAST_NAME	Фамилия ответственного.	+
RESPONSIBLE_SECOND_NAME	Отчество ответственного.	+
DATE_START	Дата начала выполнения задачи.	+
DURATION_FACT	Затраченное время на задачу (в минутах).	+
DURATION_PLAN	Планируемая длительность в часах или днях.	+
DURATION_TYPE	Тип единицы измерения в планируемой длительности: days, hours или minutes.	+

CREATED_BY_NAME	Имя постановщика.	+
CREATED_BY_LAST_NAME	Фамилия постановщика.	+
CREATED_BY_SECOND_NAME	Отчество постановщика.	+
CREATED_DATE	Дата создания задачи.	+
CHANGED_BY	Пользователь, изменивший задачу в последний раз (идентификатор пользователя).	+
CHANGED_DATE	Дата последнего изменения задачи.	+
STATUS_CHANGED_BY	Пользователь, изменивший статус задачи (идентификатор пользователя).	+
STATUS_CHANGED_DATE	Дата смены статуса.	+
CLOSED_BY	Кем была завершена задача.	+
CLOSED_DATE	Дата завершения задачи.	+
GUID	Глобально-уникальный идентификатор. С приемлемым уровнем уверенности, данный идентификатор непреднамеренно никогда не будет использован для чего-то ещё даже в других базах данных.	+
MARK	Оценка по задаче (возможные значения P (положительная) и N (отрицательная)).	+
VIEWED_DATE	Дата последнего просмотра	+

	задачи в публичном интерфейсе текущим пользователем (от имени которого делается запрос на получение данных задачи).	
TIME_SPENT_IN_LOGS	Затраченное время на задачу (в секундах).	+
FAVORITE	Присутствие и избранном для текущего пользователя.	+
ALLOW_TIME_TRACKING	Флаг включения учета затраченного времени по задаче.	+
ADD_IN_REPORT	Флаг включения задачи в отчет по эффективности.	+
FORUM_ID	[dw]Идентификатор форума[/dw][di]Форум в данном случае - понятие техническое. Сам форум нигде на портале не используется.[/di], в котором хранятся комментарии к задаче.	+
FORUM_TOPIC_ID	[dw]Идентификатор темы форума[/dw][di]Форум в данном случае - понятие техническое. Сам форум нигде на портале не используется. Комментарии к задаче хранятся в топике форума.[/di], в котором хранятся комментарии к задаче.	+
COMMENTS_COUNT	Число комментариев к задаче.	+
SITE_ID	Идентификатор сайта. По умолчанию в это поле записывается идентификатор	+

	сайта, на котором создается задача.	
SUBORDINATE	Флаг, который показывает, является ли кто-то из участников задачи подчиненным текущего пользователя.	+
FORKED_BY_TEMPLATE_ID	Идентификатор шаблона, на основе которого была автоматически создана задача. Для некоторых старых задач может быть не установлен.	+
MULTITASK	Флаг, означающий, что задача была создана для нескольких ответственных.	+
ONLY_ROOT_TASKS	Поле, позволяющее выбирать только те задачи, у которых либо нет родительской задачи, либо есть, но к этой родительской задаче мы не имеем доступа.	
MATCH_WORK_TIME	Флаг, который показывает, что даты исполнения и крайний срок должны всегда устанавливаться в рабочее время.	+

Примечание: данные поля относятся к методам task.item.*.

Также возможно фильтрация и сортировка по пользовательским полям, в частности:

Название	Описание	Чтение	Запись
UF_TASK_WEBDAV_FILES	Список идентификаторов	+	+

	закрепленных за задачами файлов.		

Примечание: Запись и изменение полей производится согласно бизнес-логике и имеющимся правам пользователя. Т.е. зависит от роли пользователя, настроек прав на группу, иерархии, некоторых флагов в задаче (например, *ALLOW_CHANGE_DEADLINE*), статуса задачи.

Поля даты/времени, которые читаются/записываются в формате ISO 8601

Название
DEADLINE
START_DATE_PLAN
END_DATE_PLAN
DATE_START
CREATED_DATE
CLOSED_DATE
CHANGED_DATE
STATUS_CHANGED_DATE
VIEWED_DATE

Частые вопросы

Решение некоторых частых задач.

Как через http запрос получить отфильтрованный список задач?

Фильтры задач по ID, дате, статусу. Для фильтра `'=ID' => 3` рекомендуется использовать tasks.task.get так как в нём нет постраничной навигации.

```
$filter = [];

//by id
$filter = [
    '>ID' => 3
];

$filter = [
    '=ID' => 3//recommend:
CRest::call('tasks.task.get');
];


//by date
$filter = [
    '<CREATED_DATE' => date(DATE_ATOM,
mktime(12, 22, 37, 7, 25, 2019))
];

//by status
$filter = [
```

```
    '>STATUS' => 2 // 2 is enum value. for
current client: CRest::call(
'tasks.task.getFields');
];


$result = CRest::call(
    'tasks.task.list',
    [
        'filter' => $filter,
        'select' => [
            'ID',
            'TITLE',
            'CREATED_DATE'
        ]
    ]
);

//all fields
$fields = CRest::call(
'tasks.task.getFields');

echo '<pre>';
print_r([$filter, $result, $fields]);
echo '</pre>';



$result = CRest::call(
    'tasks.task.get',
    [
        'taskId' => 3,
        'select' => [
            'ID',
            'TITLE',
            'CREATED_DATE'
        ]
```

```
    ]
);
echo '<pre>';
print_r($result);
echo '</pre>';
```

tasks.task.add

```
tasks.task.add(fields)
```

Метод создает задачу.

Параметры

Параметр	Описание
fields	Поля, соответствующие доступному списку полей tasks.task.getfields.

Примеры

```
BX24.callMethod(
   'tasks.task.add',
   {fields:{TITLE:'task for test',
RESPONSIBLE_ID:1}},
   function(res)
{console.log(res.answer.result);}
);
```

tasks.task.approve

```
tasks.task.approve(taskId)
```

Метод позволяет принять задачу.

Параметры

Параметр	Описание
taskId	Идентификатор задачи.

Примеры

```
BX24.callMethod(
  'tasks.task.approve',
  {taskId:1},
  function(res)
{console.log(res.answer.result);}
);
```

© «Битрикс», 2001-2008, «1С-

tasks.task.complete

```
tasks.task.complete(taskId)
```

Метод переводит задачу в статус «завершена».

Параметры

Параметр	Описание
taskId	Идентификатор задачи.

Примеры

```
BX24.callMethod(
  'tasks.task.complete',
  {taskId:1},
  function(res)
{console.log(res.answer.result);}
);
```

© «Битрикс», 2001-2008, «1С-

tasks.task.counters.get

Описание

```
tasks.task.counters.get(userId, groupId,
type)
```

Метод получения счетчиков пользователя.

Можно отфильтровать по:

- userId
- groupId
- type

Параметры

Параметр	Описание
userId	Идентификатор пользователя (пространство имен). Если `userId` не указан, то берется текущий пользователь.
groupId	Идентификатор группы пользователя.
type	Роль счетчиков: ▪ **view_all** - все роли; ▪ **view_role_responsible** - роль "Делаю";

	▪ **view_role_accomplice** - роль "Помогаю"; ▪ **view_role_auditor** - роль "Наблюдаю"; ▪ **view_role_originator** - роль "Поручил".

Примеры

```
BX24.callMethod('tasks.task.counters.get',
{userId:1, groupId:0,type:'vew_all'},
(res)=>{console.log(res.answer.result);});
```

Результат:


```
{
 "result": {
    "wo_deadline": {
     "counter": 0,
     "code": 10485760
   },
   "expired": {
     "counter": 1,
     "code": 6291456
   },
   "expired_soon": {
     "counter": 0,
     "code": 9437184
   },
   "not_viewed": {
     "counter": 0,
     "code": 1048576
   },
   "wait_ctrl": {
     "counter": 0,
```

```
      "code": 8388608
    }
  },
  "total": 1,
  "time": {
    "start": 1552383141.526606,
    "finish": 1552383141.576861,
    "duration": 0.05025482177734375,
    "processing": 0.0022799968719482 42,
    "date_start": "2019-03-
12T11:32:21+02:00",
    "date_finish": "2019-03-
12T11:32:21+02:00"
  }
}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

tasks.task.defer

```
tasks.task.defer(taskId)
```

Метод переводит задачу в статус «отложена».

Параметры

Параметр	Описание
taskId	Идентификатор задачи.

Примеры

```
BX24.callMethod(
  'tasks.task.defer',
  {taskId:1},
  function(res)
{console.log(res.answer.result);}
);
```

© «Битрикс», 2001-2008, «1С-

tasks.task.delegate

```
tasks.task.delegate(taskId, userId)
```

Метод для делегирования задачи.

Параметры

Параметр	Описание
taskId	Идентификатор задачи.
userId	Идентификатор пользователя, на которого необходимо делегировать задачу.

Примеры

```
BX24.callMethod(
    'tasks.task.delegate',
    {taskId:1, userId: 2},
    function(res)
{console.log(res.answer.result);}
);
```

tasks.task.delete

```
tasks.task.delete(taskId)
```

Метод удаляет задачу.

Параметры

Параметр	Описание
taskId	Идентификатор задачи.

Примеры

```
BX24.callMethod(
    'tasks.task.delete',
    {taskId:1},
    function(res)
{console.log(res.answer.result);}
);
```

© «Битрикс», 2001-2008, «1С-

tasks.task.disapprove

```
tasks.task.disapprove(taskId)
```

Метод позволяет отклонить задачу.

Параметры

Параметр	Описание
taskId	Идентификатор задачи.

Примеры

```
BX24.callMethod(
  'tasks.task.disapprove',
  {taskId:1},
  function(res)
{console.log(res.answer.result);}
);
```

© «Битрикс», 2001-2008, «1С-

tasks.task.favorite.add

```
tasks.task.favorite.add(taskId)
```

Метод добавления задачи в "Избранное".

В случае успешного выполнения возвращает параметр `true` (иначе `false`).

Параметры

Параметр	Описание
taskId	Идентификатор задачи.

Примеры

```
BX24.callMethod('tasks.task.favorite.add',
{taskId: 119}, (res)=>
{console.log(res.answer.result);});
```

Результат:

```
{
 "result": true,
 "time": {
   "start": 1552382402.930095,
   "finish": 1552382403.055257,
   "duration": 0.12516212463378906,
   "processing": 0.09590816497802734,
   "date_start": "2019-03-
12T11:20:02+02:00",
   "date_finish": "2019-03-
12T11:20:03+02:00"
 }
}
```

tasks.task.favorite.delete

```
tasks.task.favorite.delete(taskId)
```

Метод удаления задачи из "Избранного".

В случае успешного выполнения возвращает параметр `true` (иначе `false`).

Параметры

Параметр	Описание
taskId	Идентификатор задачи.

Примеры

```
BX24.callMethod('tasks.task.favorite.delete'
, {taskId: 119}, (res)=>
{console.log(res.answer.result);});
```

Результат:

```
{
 "result": true,
 "time": {
   "start": 1552382402.930095,
   "finish": 1552382403.055257,
   "duration": 0.12516212463378906,
   "processing": 0.09590816497802734,
   "date_start": "2019-03-
12T11:20:02+02:00",
   "date_finish": "2019-03-
12T11:20:03+02:00"
 }
}
```

tasks.task.files.attach

```
tasks.task.files.attach(taskId, fileId, params)
```

Метод для прикрепления загруженного на диск файла к задаче.

Параметры

Параметр	Описание
taskId	Идентификатор задачи.
fileId	Идентификатор загруженного на диск файла.
params	Массив с дополнительными параметрами, по умолчанию пустой. В настоящее время не используется.

Примеры

```
BX24.callMethod(
    'tasks.task.files.attach',
   {
```

```
        taskId: 1,
        fileId: 1065,
    },
   function(res) {
        console.log(res.answer.result);
    }
);
```

tasks.task.get

Описание и пример

Возвращает информацию о конкретной задаче.

Внимание! Необходимо указать поля в **select**, т.к. поля по умолчанию могут быть изменены в будущем.

Пример

```
BX24.callMethod(
        'tasks.task.get',
{taskId:1, select:{'ID','TITLE'}},
function(res)
{console.log(res.answer.result);}
);
```

Параметры

Параметр	Описание
taskId	Идентификатор задачи.
select	Массив полей записей, которые будут возвращены методом. Можно указать только те поля, которые необходимы. Поле может принимать значения:

- **ID** - идентификатор задачи;
- **PARENT_ID** - идентификатор родительской задачи;
- **TITLE** - название задачи;
- **DESCRIPTION** - описание;
- **MARK** - оценка;
- **PRIORITY** - приоритет:
  - **0** - низкий;
  - **1** - средний;
  - **2** - высокий.
- **STATUS** - статус;
- **MULTITASK** - множественная задача;
- **NOT_VIEWED** - непросмотренная задача;
- **REPLICATE** - повторяемая задача;
- **GROUP_ID** - рабочая группа.
- **STAGE_ID** - стадия;
- **CREATED_BY** - постановщик;
- **CREATED_DATE** - дата создания;
- **RESPONSIBLE_ID** - исполнитель;
- **ACCOMPLICE** - идентификатор соисполнителя;
- **AUDITOR** - идентификатор аудитора;
- **CHANGED_BY** - кем изменена задача;
- **CHANGED_DATE** - дата изменения;
- **STATUS_CHANGED_DATE** - кто изменил статус;
- **CLOSED_BY** - кто закрыл задачу;
- **CLOSED_DATE** - дата закрытия задачи;
- **DATE_START** - дата начала;
- **DEADLINE** - крайний срок;
- **START_DATE_PLAN** - плановое начало;
- **END_DATE_PLAN** - плановое завершение;
- **GUID** - GUID (статистически уникальный 128-битный идентификатор);
- **XML_ID** - внешний код;
- **COMMENTS_COUNT** - количество комментариев;

- **NEW_COMMENTS_COUNT** - количество новых комментариев;
- **TASK_CONTROL** - принять в работу;
- **ADD_IN_REPORT** - добавить в отчет;
- **FORKED_BY_TEMPLATE_ID** - создано автоматически из шаблона;
- **TIME_ESTIMATE** - затраченное время;
- **TIME_SPENT_IN_LOGS** - затраченное время из истории изменений;
- **MATCH_WORK_TIME** - пропустить выходные дни;
- **FORUM_TOPIC_ID** - идентификатор темы форума;
- **FORUM_ID** - идентификатор форума;
- **SITE_ID** - идентификатор сайта;
- **SUBORDINATE** - задача подчиненного;
- **FAVORITE** - Избранное;
- **VIEWED_DATE** - дата последнего просмотра;
- **SORTING** - индекс сортировки;
- **DURATION_PLAN** - затрачено (план);
- **DURATION_FACT** - затрачено (фактически);
- **DURATION_TYPE** - Тип продолжительности:
  - 0 - секунды
  - 1 - минуты
  - 2 - часы
  - 3 - дни
  - 4 - недели
  - 5 - месяцы
  - 6 - года
- **UF_CRM_TASK** - привязка к элементам CRM.

По умолчанию будут возвращены все невычисляемые поля основной таблицы запроса.

Для получения пользовательских полей и поля привязки к CRM сущностям (UF_CRM_TASK),

	их нужно будет напрямую указать в SELECT. Список полей можно уточнить, отправив запрос tasks.task.getFields.

tasks.task.getaccess

```
tasks.task.getaccess(taskId, users)
```

Метод для проверки доступа к задаче.

Параметры

Параметр	Описание
taskId	Идентификатор задачи.
users	Массив ID пользователей , которым требуется проверить доступ. По умолчанию подставляется текущий пользователь.

Примеры

```
BX24.callMethod(
    'tasks.task.getaccess',
    {taskId:1, users:[1]},
    function(res)
{console.log(res.answer.result);}
);
```

tasks.task.getFields

Метод возвращает все доступные поля.

Параметры метода

Без параметров.

Примеры

```
BX24.callMethod(
        'tasks.task.getFields',
        {},
        function(result)
        {
                console.info(result.data());
                console.log(result);
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

tasks.task.history.list

```
tasks.task.history.list(taskId)
```

Метод получения истории задачи.

Возвращаем массив данных (см. пример).

Можно фильтровать и сортировать по всем полям (см. tasks.task.list). По умолчанию отдает всю историю без разбивки по страницам.

Параметры

Параметр	Описание
taskId	Идентификатор задачи.

Примеры

1. Вывод истории конкретной задачи без использования фильтров:

```
BX24.callMethod('tasks.task.history.list'
, {taskId: 125}, (res)=>
{console.log(res.answer.result);});
```

Результат:

2. Вывод истории конкретной задачи с использованием фильтра `NEW` (т.е. когда была создана задача):

```
BX24.callMethod('tasks.task.history.list'
, {taskId: 119, filter:{FIELD:'NEW'}},
(res)=>
{console.log(res.answer.result);});
```

Результат:


```
{
 "result": {
   "list": [
     {
       "id": "1230",
       "createdDate": "01.03.2019
15:29:28",
       "field": "NEW",
       "value": {
         "from": null,
         "to": null
       },
       "user": {
         "id": "1",
         "name": "Максим",
         "lastName": "Гречушников",
         "secondName": "",
         "login": "admin"
       }
     }
   ]
 },
 "time": {
   "start": 1552382093.81029,
```

```
    "finish": 1552382093.927268,
    "duration": 0.11697793006896973,
    "processing": 0.018744230270385742,
    "date_start": "2019-03-
12T11:14:53+02:00",
    "date_finish": "2019-03-
12T11:14:53+02:00"
  }
}
```

tasks.task.list

Описание

Возвращает массив задач, каждая из которых содержит массив полей. В отличие от task.item.list параметры в запросе **tasks.task.list** можно указывать в любом порядке, а также можно не указывать ненужные параметры.

Для получения данных по всем задачам пользователь должен обладать админскими правами. Руководитель подразделения получит доступ только к задачам в своей ветке иерархии.

Также можно получить задачи в "Избранном", установив фильтрацию по параметру `$filter[::SUBFILTER-PARAMS][FAVORITE]=Y`.

Внимание! Необходимо указать поля в **select**, т.к. поля по умолчанию могут быть изменены в будущем.

Параметры

Параметр	Описание
order	Массив для сортировки результата. Массив вида *{"поле_сортировки": 'направление сортировки' [, ...]}*. Поле для сортировки может принимать значения: ▪ **ID** - идентификатор задачи; ▪ **TITLE** - название задачи; ▪ **TIME_SPENT_IN_LOGS** - затраченное время из истории изменений; ▪ **DATE_START** - дата старта;

- **CREATED_DATE** - дата создания;
- **CHANGED_DATE** - дата последнего изменения;
- **CLOSED_DATE** - дата завершения;
- **START_DATE_PLAN** - плановое начало;
- **END_DATE_PLAN** - плановое завершение;
- **DEADLINE** - крайний срок;
- **REAL_STATUS** - статус задачи. Константы отражающие статусы задач:
  - STATE_NEW = 1;
  - STATE_PENDING = 2;
  - STATE_IN_PROGRESS = 3;
  - STATE_SUPPOSEDLY_COMPLETED = 4;
  - STATE_COMPLETED = 5;
  - STATE_DEFERRED = 6;
  - STATE_DECLINED = 7;
- **STATUS_COMPLETE** - флаг завершенности задачи;
- **PRIORITY** - приоритет;
- **MARK** - оценка;
- **CREATED_BY_LAST_NAME** - постановщик;
- **RESPONSIBLE_LAST_NAME** - ответственный;
- **GROUP_ID** - рабочая группа.
- **TIME_ESTIMATE** - затраченное время;
- **ALLOW_CHANGE_DEADLINE** - флаг "Разрешить ответственному менять крайний срок";
- **ALLOW_TIME_TRACKING** - флаг включения учета затраченного времени по задаче;
- **MATCH_WORK_TIME** - пропустить выходные дни;
- **FAVORITE** - Избранное;
- **SORTING** - индекс сортировки;
- **MESSAGE_ID** - идентификатор поискового индекса;

	Примечание: В коробочной версии этот список полей для сортировки можно вернуть методом CTasks::getAvailableOrderFields(). Направление сортировки может принимать значения: ▪ **asc** - по возрастанию; ▪ **desc** - по убыванию; Необязательный. По умолчанию фильтруется по убыванию идентификатора задачи.

filter	Массив вида *{"фильтруемое_поле": "значение фильтра" [, ...]}*. Фильтруемое поле может принимать значения:
	
	▪ **ID** - идентификатор задачи;
	▪ **PARENT_ID** - идентификатор родительской задачи;
	▪ **GROUP_ID** - идентификатор рабочей группы;
	▪ **CREATED_BY** - постановщик;
	▪ **STATUS_CHANGED_BY** - пользователь, последним изменивший статус задачи;
	▪ **PRIORITY** - приоритет;
	▪ **FORUM_TOPIC_ID** - идентификатор темы форума;
	▪ **RESPONSIBLE_ID** - ответственный;
	▪ **TITLE** - название задачи (можно искать по шаблону [%_]) ;
	▪ **TAG** - тэг;
	▪ **REAL_STATUS** - статус задачи. Константы отражающие статусы задач:
	
	  ▪ STATE_NEW = 1;
	  ▪ STATE_PENDING = 2;
	  ▪ STATE_IN_PROGRESS = 3;
	  ▪ STATE_SUPPOSEDLY_COMPLETED = 4;
	  ▪ STATE_COMPLETED = 5;
	  ▪ STATE_DEFERRED = 6;
	  ▪ STATE_DECLINED = 7;

- **STATUS** - статус для сортировки. Аналогичен **REAL_STATUS**, но имеет дополнительно три мета-статуса:
  - **-3** - задача почти просрочена;
  - **-2** - не просмотренная задача;
  - **-1** - просроченная задача.
- **MARK** - оценка;
- **SITE_ID** - идентификатор сайта;
- **ADD_IN_REPORT** - задача в отчете (Y|N);
- **DATE_START** - дата начала выполнения;
- **DEADLINE** - крайний срок;
- **CREATED_DATE** - дата создания;
- **CLOSED_DATE** - дата завершения;
- **CHANGED_DATE** - дата последнего изменения;
- **ACCOMPLICE** - идентификатор соисполнителя;
- **AUDITOR** - идентификатор наблюдателя;
- **DEPENDS_ON** - идентификатор предыдущей задачи;
- **ONLY_ROOT_TASKS** - только задачи, которые не являются подзадачами (корневые задачи), а также подзадачи родительской задачи, к которой текущий пользователь доступа не имеет (Y|N).
- **STAGE_ID** - стадия;

Перед названием фильтруемого поля может указать тип фильтрации:

- "!" - не равно
- "<" - меньше
- "<=" - меньше либо равно
- ">" - больше
- ">=" - больше либо равно

"*значения фильтра*" - одиночное значение или массив.

	Необязательный. По умолчанию записи не фильтруются.

select	Массив полей записей, которые будут возвращены методом. Можно указать только те поля, которые необходимы.
	Поле сортировки может принимать значения:
	▪ **ID** - идентификатор задачи;
	▪ **PARENT_ID** - идентификатор родительской задачи;
	▪ **TITLE** - название задачи;
	▪ **DESCRIPTION** - описание;
	▪ **MARK** - оценка;
	▪ **PRIORITY** - приоритет:
	  ▪ **0** - низкий;
	  ▪ **1** - средний;
	  ▪ **2** - высокий.
	▪ **STATUS** - статус;
	▪ **MULTITASK** - множественная задача;
	▪ **NOT_VIEWED** - непросмотренная задача;
	▪ **REPLICATE** - повторяемая задача;
	▪ **GROUP_ID** - рабочая группа.
	▪ **STAGE_ID** - стадия;
	▪ **CREATED_BY** - постановщик;
	▪ **CREATED_DATE** - дата создания;
	▪ **RESPONSIBLE_ID** - исполнитель;
	▪ **ACCOMPLICES** - идентификатор соисполнителя;
	▪ **AUDITORS** - идентификатор аудитора;
	▪ **CHANGED_BY** - кем изменена задача;
	▪ **CHANGED_DATE** - дата изменения;
	▪ **STATUS_CHANGED_DATE** - кто изменил статус;
	▪ **CLOSED_BY** - кто закрыл задачу;
	▪ **CLOSED_DATE** - дата закрытия задачи;
	▪ **DATE_START** - дата начала;

- **DEADLINE** - крайний срок;
- **START_DATE_PLAN** - плановое начало;
- **END_DATE_PLAN** - плановое завершение;
- **GUID** - GUID (статистически уникальный 128-битный идентификатор);
- **XML_ID** - внешний код;
- **COMMENTS_COUNT** - количество комментариев;
- **NEW_COMMENTS_COUNT** - количество новых комментариев;
- **TASK_CONTROL** - принять в работу;
- **ADD_IN_REPORT** - добавить в отчет;
- **FORKED_BY_TEMPLATE_ID** - создано из шаблона;
- **TIME_ESTIMATE** - затраченное время;
- **TIME_SPENT_IN_LOGS** - затраченное время из истории изменений;
- **MATCH_WORK_TIME** - пропустить выходные дни;
- **FORUM_TOPIC_ID** - идентификатор темы форума;
- **FORUM_ID** - идентификатор форума;
- **SITE_ID** - идентификатор сайта;
- **SUBORDINATE** - задача подчиненного;
- **FAVORITE** - Избранное;
- **VIEWED_DATE** - дата последнего просмотра;
- **SORTING** - индекс сортировки;
- **DURATION_PLAN** - затрачено (план);
- **DURATION_FACT** - затрачено (фактически);
- **DURATION_TYPE** - тип единицы измерения в планируемой длительности: days, hours или minutes

По умолчанию будут возвращены все **невычисляемые** поля основной таблицы запроса.

Список полей можно уточнить, отправив запрос **tasks.task.getFields**.

limit	Число записей. Параметр указывается, если нужно получить число записей более значения по умолчанию (50). Все записи одним запросом вернуть нет возможности, это ограничение всех методов REST API. Вы можете несколькими запросами по 50 записей в ответе получить все лиды. Для этого просто передавайте параметр start со значением, кратным 50. Пример:
	`start=0`
	`start=50`
	`start=100`

Примеры

1. Вывод всех задач с названием "task for test", отбор по полям `ID`, `TITLE`, `STATUS`, сортировка по полю `ID` (сортировка по возрастанию):

```
BX24.callMethod(
        'tasks.task.list',
        {filter:{TITLE:'task for test'},
select: ['ID','TITLE','STATUS'], order:
{ID:'asc'}},
        function(res)
{console.log(res.answer.result);}
);
```

   Результат:

2. Вывод всех неповторяющихся задач, добавленных в "Избранное", у которых статус больше 2:

```
BX24.callMethod(
        'tasks.task.list',
```

```
        {filter:{'>STATUS':2,
REPLICATE:'N', '::SUBFILTER-PARAMS':
{FAVORITE:'Y'}}},
        function(res)
{console.log(res.answer.result);}
);
```

Результат:

```
result: {
        tasks: [
                {
                        id: "117",
                        parentId: null,
                        title: "123",
                        description:
"Opisanie.   Ghhhhh",
                        mark: null,
                        priority: "0",
                        status: "6",
                        multitask: "N",
                        notViewed: "N",
                        replicate: "N",
                        groupId: "0",
                        stageId: "0",
                        createdBy: "1",
                        createdDate:
"2019-03-15T15:41:27+02:00",
                        responsibleId:
"1",
                        changedBy: "1",
                        changedDate:
"2019-03-18T14:06:18+02:00",
                        statusChangedBy:
"1",
```

statusChangedDate: "2019-03-
18T14:05:54+02:00",
closedBy: null,
closedDate: null,
dateStart: null,
deadline: null,
startDatePlan:
null,
endDatePlan:
null,
guid: "{aef54f9d-
8157-464b-9069-6ded9745e26d}",
xmlId: null,
commentsCount:
null,
taskControl: "Y",
addInReport: "N",

forkedByTemplateId: null,
timeEstimate:
"0",
timeSpentInLogs:
null,
matchWorkTime:
"N",
forumTopicId:
null,
forumId: null,
siteId: "s1",
subordinate: "Y",
favorite: "Y",
exchangeModified:
null,
exchangeId: null,
outlookVersion:
"7",
viewedDate:

"2019-03-18T14:07:08+02:00",
sorting:
"2564.0000000",
durationPlan:
"0",
durationFact:
null,
durationType:
"days",
descriptionInBbcode: "Y",
ufCrmTask: [ ],
ufTaskWebdavFiles: [
22
],
ufAuto915658270214: null,
ufAuto244510721805: null,
ufAuto637823431651: "0",
ufMailMessage:
null,
ufAuto226929532613: "",
ufAuto187628303463: null,
auditors: [ ],
accomplices: [ ],
newCommentsCount:
0,
subStatus: "6",
creator: {
id: "1",
name:
"Гречушников Максим",

```
                                link:
"/company/personal/user/1/",
                                icon:
"/upload/resize_cache/main/9b0/58_58_2/p2
dVDwA46Nw.png",
                        },
                        responsible: {
                                id: "1",
                                name:
"Гречушников Максим",
                                link:
"/company/personal/user/1/",
                                icon:
"/upload/resize_cache/main/9b0/58_58_2/p2
dVDwA46Nw.png",
                        },
                }
        ]
},
```

Пример отключения постраничной навигации:

```
$result = CRest::call(
    'tasks.task.list',
    [
        'filter' => [
            '>ID' => 50
        ],
        'start' => -1,
    ]
);
```

tasks.task.pause

```
tasks.task.pause(taskId)
```

Метод останавливает выполнение задачи, переводя ее в статус "ждет выполнения".

Параметры

Параметр	Описание
taskId	Идентификатор задачи.

Примеры

```
BX24.callMethod(
   'tasks.task.pause',
   {taskId:1},
   function(res)
{console.log(res.answer.result);}
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

tasks.task.renew

```
tasks.task.renew(taskId)
```

Метод возобновляет задачу после ее завершения.

Параметры

Параметр	Описание
taskId	Идентификатор задачи.

Примеры

```
BX24.callMethod(
  'tasks.task.renew',
  {taskId:1},
  function(res)
{console.log(res.answer.result);}
);
```

© «Битрикс», 2001-2008, «1С-

tasks.task.start

```
tasks.task.start(taskId)
```

Метод переводит задачу в статус «выполняется».

Параметры

Параметр	Описание
taskId	Идентификатор задачи.

Примеры

```
BX24.callMethod(
   'tasks.task.start',
   {taskId:1},
   function(res)
{console.log(res.answer.result);}
);
```

© «Битрикс», 2001-2008, «1С-

tasks.task.startwatch

```
tasks.task.startwatch(taskId)
```

Метод позволяет наблюдать за задачей.

Параметры

Параметр	Описание
taskId	Идентификатор задачи.

Примеры

```
BX24.callMethod(
 'tasks.task.startwatch',
 {taskId:1},
 function(res)
{console.log(res.answer.result);}
);
```

© «Битрикс», 2001-2008, «1С-

tasks.task.stopwatch

```
tasks.task.stopwatch(taskId)
```

Метод останавливает наблюдение за задачей.

Параметры

Параметр	Описание
taskId	Идентификатор задачи.

Примеры

```
BX24.callMethod(
 'tasks.task.stopwatch',
 {taskId:1},
 function(res)
{console.log(res.answer.result);}
);
```

© «Битрикс», 2001-2008, «1С-

tasks.task.update

```
tasks.task.update(taskId, fields)
```

Метод обновляет задачу.

Параметры

Параметр	Описание
taskId	Идентификатор задачи.
fields	Поля, соответствующие доступному списку полей tasks.task.getfields.

Примеры

```
BX24.callMethod(
  'tasks.task.update',
  {taskId:1, fields:{TITLE:'task for test',
RESPONSIBLE_ID:1}},
  function(res)
{console.log(res.answer.result);}
);
```

task.item.add

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Создает новую задачу. Возвращает идентификатор добавленной задачи. Доступны следующие поля.

Параметры функции

Параметр	Описание
TASKDATA	Массив полей данных по задаче (**TITLE**, **DESCRIPTION** и т.д.).

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
// Создадим задачу
var dt = new Date();
BX24.callMethod(
        'task.item.add',
        [{TITLE: 'created via REST API at '
+ dt.toLocaleString(), RESPONSIBLE_ID: 1,
DEADLINE: '2013-05-13T16:06:06+03:00'}],
```

```
        function(result)
        {
                console.info(result.data());
                console.log(result);
        }
);
```

Пример записи значений с CRM:

```
BX24.callMethod(
   'task.item.update',
   [1, {UF_CRM_TASK: ["L_4", "C_7", "CO_5",
"D_10"]}],
   function(result)
        {
       console.info(result.data());
       console.log(result);
        }
);
```

Цифры это ID соответствующих значений. Значение `L_4` означает привязку к задаче лида с ID = 4. Можно задавать несколько связей одного типа, например `L_4, L_5`.

- **L** - лид
- **C** - контакт
- **CO** - компания
- **D** - сделка

task.item.delete

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Удаляет задачу.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.item.delete',
   [13],
   function(result)
        {
       console.info(result.data());
       console.log(result);
```

```
            }
);
```

task.item.getdata

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Возвращает массив данных о задаче (**TITLE**, **DESCRIPTION** и т.д.). Доступны следующие поля.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
// Получим данные по задаче с
идентификатором 2:
BX24.callMethod(
        'task.item.getdata',
        [2],
        function(result)
        {
```

```
                    console.info(result.data());
                    console.log(result);
            }
);
```

task.item.getmanifest

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Возвращает список методов вида **task.item.*** и их описание.

Возвращаемое значение этого метода не предназначено для автоматической обработки, т.к. его формат может быть изменен без предупреждения.

Метод может быть полезен в качестве справочной информации, т.к. всегда содержит актуальную информацию.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
// Получим описание методов и допустимых
полей для класса CTaskItem
BX24.callMethod(
        'task.item.getmanifest',
        [],
        function(result)
        {
                console.info(result.data());
                console.log(result);
        }
);
```

task.item.list

Описание

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Возвращает массив задач, каждая из которых содержит массив полей (аналогичен массиву, возвращаемому task.item.getdata).

Параметры

Параметр	Описание
ORDER	Массив для сортировки результата. Массив вида *{"поле_сортировки": 'направление сортировки' [, ...]}*. Поле для сортировки может принимать значения: ▪ **TITLE** - название задачи; ▪ **DATE_START** - дата старта; ▪ **DEADLINE** - крайний срок; ▪ **STATUS** - статус; ▪ **PRIORITY** - приоритет; ▪ **MARK** - оценка; ▪ **CREATED_BY** - постановщик; ▪ **RESPONSIBLE_ID** - ответственный; ▪ **GROUP_ID** - рабочая группа. Направление сортировки может принимать значения: ▪ **asc** - по возрастанию;

	▪ **desc** - по убыванию; Необязательный. По умолчанию фильтруется по убыванию идентификатора задачи.

FILTER	Массив вида *{"фильтруемое_поле": "значение фильтра" [, ...]}*. Фильтруемое поле может принимать значения:
	
	▪ **ID** - идентификатор задачи;
	▪ **PARENT_ID** - идентификатор родительской задачи;
	▪ **GROUP_ID** - идентификатор рабочей группы;
	▪ **CREATED_BY** - постановщик;
	▪ **STATUS_CHANGED_BY** - пользователь, последним изменивший статус задачи;
	▪ **PRIORITY** - приоритет;
	▪ **FORUM_TOPIC_ID** - идентификатор темы форума;
	▪ **RESPONSIBLE_ID** - ответственный;
	▪ **TITLE** - название задачи (можно искать по шаблону [%_]) ;
	▪ **TAG** - тэг;
	▪ **REAL_STATUS** - статус задачи. Константы отражающие статусы задач:
	  ▪ STATE_NEW = 1;
	  ▪ STATE_PENDING = 2;
	  ▪ STATE_IN_PROGRESS = 3;
	  ▪ STATE_SUPPOSEDLY_COMPLETED = 4;
	  ▪ STATE_COMPLETED = 5;
	  ▪ STATE_DEFERRED = 6;
	  ▪ STATE_DECLINED = 7;
	
	▪ **STATUS** - статус для сортировки. Аналогичен **REAL_STATUS**, но имеет дополнительно два мета статуса:
	  ▪ **-2** - не просмотренная задача;
	  ▪ **-1** - просроченная задача.
	
	▪ **MARK** - оценка;

	▪ **SITE_ID** - идентификатор сайта; ▪ **ADD_IN_REPORT** - задача в отчете (Y|N); ▪ **DATE_START** - дата начала выполнения; ▪ **DEADLINE** - крайний срок; ▪ **CREATED_DATE** - дата создания; ▪ **CLOSED_DATE** - дата завершения; ▪ **CHANGED_DATE** - дата последнего изменения; ▪ **ACCOMPLICE** - идентификатор соисполнителя; ▪ **AUDITOR** - идентификатор аудитора; ▪ **DEPENDS_ON** - идентификатор предыдущей задачи; ▪ **ONLY_ROOT_TASKS** - только задачи, которые не являются подзадачами (корневые задачи), а также подзадачи родительской задачи, к которой текущий пользователь доступа не имеет (Y|N). Перед названием фильтруемого поля может указать тип фильтрации: ▪ "!" - не равно ▪ "<" - меньше ▪ "<=" - меньше либо равно ▪ ">" - больше ▪ ">=" - больше либо равно "*значения фильтра*" - одиночное значение или массив. Необязательный. По умолчанию записи не фильтруются. **Внимание!** Для метода **task.item.list** *обязательно* нужно указывать сортировку для фильтрации. Фильтрация без сортировки возвращает все задачи.
PARAMS	Массив для опций вызова. Элементом является массив NAV_PARAMS вида *{"опция*

	вызова": 'значение' [, ...]}, хранящий следующие опции: ▪ **nPageSize** - количество элементов на странице. В целях ограничения нагрузки на постраничную навигацию наложено ограничение в 50 задач. ▪ **iNumPage** - номер страницы при постраничной навигации.
SELECT	Массив полей записей, которые будут возвращены методом. Можно указать только те поля, которые необходимы. Если в массиве присутствует значение `"*"`, то будут возвращены все доступные поля. Значение по умолчанию - пустой массив `array()` - означает, что будут возвращены все поля основной таблицы запроса.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Однако, если какие-то параметры нужно пропустить, то их все равно нужно передать, но в виде пустых массивов: ORDER[]=&FILTER[]=&PARAMS[]=&SELECT[]=.

Примеры

```
// Пример для работы с JavaScript
// Получим список всех задач (по умолчанию
сработает ограничение — постраничка по 50
элементов)
BX24.callMethod(
        'task.item.list',
        [],
        function(result)
        {
                console.info(result.data());
```

```
                console.log(result);
        }
);
```

```
// Пример для работы с JavaScript
// Получим список задач с идентификаторами
1,2,3,4,5,6. Причем выберем только поля ID и
TITLE.
// Режим постранички — по 2 элемента на
странице, 2-ая страница.
// Сортировка по ID — по убыванию.
BX24.callMethod(
        'task.item.list',
        [
                {ID : 'desc'},          //
Сортировка по ID — по убыванию.
                {ID: [1,2,3,4,5,6]},    //
Фильтр
                {
            NAV_PARAMS: { // постраничка
                        nPageSize : 2,  //
по 2 элемента на странице.
                        iNumPage  : 2   //
страница номер 2
                        }
                }
        ],
        function(result)
        {
                console.info(result.data());
                console.log(result);
        }
);
```

```
// Пример для работы с php
// Получение GET-запроса на выборку данных.
$appParams = array(
     "auth" =>
'92006f4ae0c55d400f1e6e09428af64a',
     "ORDER" => array("DEADLINE" => "desc"),
     "FILTER" => array(">ID" => 1),
     "PARAMS" => array('NAV_PARAMS' =>
array("nPageSize" => 2, 'iNumPage' => 2)),
);

$appRequestUrl = 'http://test-
domain.ru/rest/task.item.list.xml?'.http_bui
ld_query($appParams);

print(urldecode($appRequestUrl));;
```

Метод отдаёт теги, если передать ему параметр LOAD_TAGS:

```
/rest/task.item.list.xml?
auth=31r0ckfy3r2u96yttz4k70g5kv2w534h&O[]=&F
[]=&P[LOAD_TAGS]=Y
```

task.item.update

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Обновляет данные по задаче. Доступны для обновления следующие поля. При обновлении данных по задаче учитывается бизнес-логика и права.

Например: Ответственный не может переименовать задачу - в таком случае будет сгенерирована ошибка.

Рекомендуется перед обновлением данных проверять, допустимо ли данное действие (task.item.isactionallowed).

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.
TASKDATA	Список полей с новыми значениями.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.item.update',
   [1, {TIME_ESTIMATE: 113}],
   function(result)
        {
       console.info(result.data());
       console.log(result);
        }
);
```

Пример записи значений с CRM:

```
BX24.callMethod(
   'task.item.update',
   [1, {UF_CRM_TASK: ["L_4", "C_7", "CO_5",
"D_10"]}],
   function(result)
        {
       console.info(result.data());
       console.log(result);
        }
);
```

Цифры это ID соответствующих значений. Значение `L_4` означает привязку к задаче лида с ID = 4. Можно задавать несколько связей одного типа, например `L_4, L_5`.

- **L** - лид
- **C** - контакт
- **CO** - компания
- **D** - сделка

© «Битрикс», 2001-2008, «1С-

task.item.getdescription

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Возвращает описание задачи.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.
FORMAT	Допустимые значения: ▪ **1** (соответствует PHP-константе CTaskItem::DESCR_FORMAT_RAW) — описание будет возвращено в том формате, в котором хранится в БД (HTML, либо BB-code), обработка санитайзером производиться не будет; ▪ **2** (соответствует PHP-константе CTaskItem::DESCR_FORMAT_HTML) — описание будет возвращено в формате HTML, предварительно будет обработано санитайзером (если это включено в настройках модуля задач); ▪ **3** (соответствует PHP-константе CTaskItem::DESCR_FORMAT_PLAIN_TEXT) — описание будет возвращено в виде «плоского» текста (без HTML-тегов).

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.item.getdescription',
   [13, 1],
   function(result)
    {
       console.info(result.data());
       console.log(result);
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

task.item.getfiles

Возвращает массив, содержащий ссылки на файлы, прикрепленные к задаче.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

task.item.getdependson

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Возвращает массив, содержащий идентификаторы задач, от которых зависит задача (опция **Предыдущие задачи** в форме создания задачи).

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.item.getdependson',
   [13],
   function(result)
        {
       console.info(result.data());
```

```
        console.log(result);
         }
);
```

task.item.getallowedactions

Описание

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Возвращает массив, идентификаторов допустимых действий над задачей (см. констатнты PHP-класса CTaskItem).

Параметры

Параметр	Описание
TASKID	Идентификатор задачи.

Таблица соответствия идентификаторов и допустимых действий над задачей

Идентификатор	Описание
1	ACTION_ACCEPT.
2	ACTION_DECLINE.
3	ACTION_COMPLETE.

4	ACTION_APPROVE.
5	ACTION_DISAPPROVE.
6	ACTION_START.
7	ACTION_DELEGATE.
8	ACTION_REMOVE.
9	ACTION_EDIT.
10	ACTION_DEFER.
11	ACTION_RENEW.
12	ACTION_CREATE.
13	ACTION_CHANGE_DEADLINE.
14	ACTION_CHECKLIST_ADD_ITEMS.
15	ACTION_ELAPSED_TIME_ADD.
16	ACTION_CHANGE_DIRECTOR.
17	ACTION_PAUSE.
18	ACTION_START_TIME_TRACKING.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Примеры

```
BX24.callMethod(
        'task.item.getallowedactions',
```

```
        [13],
        function(result)
        {
                console.info(result.data());
                console.log(result);
        }
);
```

task.item.getallowedtaskactior

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Возвращает массив, ключи которого являются названиями действий (названия соответствуют константам PHP-класса CTaskItem), а значения показывают, допустимо действие (**true**) или нет (**false**).

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(

'task.item.getallowedtaskactionsasstrings',
        [13],
        function(result)
```

```
        {
                console.info(result.data());
                console.log(result);
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

task.item.isactionallowed

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Возвращает **true** если действие разрешено, иначе — **false**.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.
ACTIONID	Идентификатор проверяемого действия (см. констатнты метода task.item.getallowedactions).

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.item.isactionallowed',
   [13, 6],
   function(result)
```

```
    {
        console.info(result.data());
        console.log(result);
    }
);
```

task.item.delegate

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Делегирует задачу новому пользователю.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.
USERID	Идентификатор нового ответственного.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
        'task.item.delegate',
        [13, 3],
        function(result)
        {
                console.info(result.data());
```

```
                    console.log(result);
            }
);
```

task.item.startexecution

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Переводит задачу в статус «выполняется».

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
// Начать задачу с идентификатором 3:
BX24.callMethod(
        'task.item.startexecution',
        [3],
        function(result)
        {
                console.info(result.data());
                console.log(result);
```

```
            }
);
```

task.item.defer

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Переводит задачу в статус «отложена».

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.

Примечание: Для выполнения данной операции, задача должна быть в статусе Выполняется.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
        'task.item.defer',
        [13],
        function(result)
        {
```

```
                    console.info(result.data());
                    console.log(result);
            }
);
```

task.item.complete

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Переводит задачу в статус «завершена» или «условно завершена (ждет контроля исполнителя)».

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
        'task.item.complete',
        [13],
        function(result)
        {
                console.info(result.data());
                console.log(result);
```

```
            }
);
```

task.item.renew

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Переводит задачу в статус «не выполняется».

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.

Примечание: Применимо для задач в статусе Завершена.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
        'task.item.renew',
        [13],
        function(result)
        {
                console.info(result.data());
```

```
                    console.log(result);
            }
);
```

task.item.approve

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Переводит задачу, ожидающую контроля, в статус «завершена».

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
        'task.item.approve',
        [13],
        function(result)
        {
                console.info(result.data());
                console.log(result);
```

```
            }
);
```

task.item.disapprove

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Переводит задачу, ожидающую контроля, в статус «не выполняется».

Параметры метода

Параметр	Описание
TASKID	Идентификатор задачи.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
        'task.item.disapprove',
        [13],
        function(result)
        {
                console.info(result.data());
                console.log(result);
```

```
            }
);
```

task.item.addtofavorite

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Метод добавляет задачу в **Избранное**.

Параметры метода

Параметр	Описание
auth	Токен авторизации.
TASK_ID	Идентификатор задачи.
PARAMS	Параметр содержит ключ *AFFECT_CHILDREN*. Он указывает, добавлять ли в избранное подзадачи данной задачи.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример вызова

```
$request ='http://your-domain.ru/rest/task.item.addtofavorite.xml?
```

```
auth=mqa17fnd5cth4rpwtizyl49tbnzp7omf&TASK_I
D=10&PARAMS[AFFECT_CHILDREN]=Y'
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

task.item.deletefromfavorite

Внимание! Метод устарел и не поддерживается. Рекомендуется использовать методы tasks.task.*.

Метод удаляет задачу из **Избранного**.

Параметры метода

Параметр	Описание
auth	Токен авторизации.
TASK_ID	Идентификатор задачи.
PARAMS	Параметр содержит ключ *AFFECT_CHILDREN*. Он указывает, добавлять ли в избранное подзадачи данной задачи.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример вызова

```
$request = 'http://your-domain.ru/rest/task.item.deletefromfavorite.
```

```
xml?
auth=mqa17fnd5cth4rpwtizyl49tbnzp7omf&TASK_I
D=10&PARAMS[AFFECT_CHILDREN]=Y'
```

task.item.addfile

Метод загружает к задаче файл. Пока реализована загрузка файла через **post** с передачей содержимого файла в параметре `CONTENT`.

Параметры метода

Параметр	Описание
TASK_ID	Идентификатор задачи.
NAME	Имя файла.
CONTENT	Содержимое файла в **base64**.

Пример

```
$curl = curl_init();
curl_setopt($curl, CURLOPT_URL,
'http://test-
domain.ru/rest/task.item.addfile.xml');
curl_setopt($curl,
CURLOPT_RETURNTRANSFER,true);
curl_setopt($curl, CURLOPT_POST, true);
curl_setopt($curl, CURLOPT_POSTFIELDS,
array(
    'auth' =>
```

```
'z3eamwwkpgl7u18kx14q1s4c0ffckqsn',
    'TASK_ID' => "140",
    'FILE[NAME]' => 'desc.txt',
    'FILE[CONTENT]' =>
base64_encode(file_get_contents($_SERVER['DO
CUMENT_ROOT'] .'/desc.txt'))
));
$out = curl_exec($curl);
print($out);
curl_close($curl);
```

task.item.deletefile

Метод удаляет привязку файла к задаче.

Параметры метода

Параметр	Описание
auth	Токен авторизации.
TASK_ID	Идентификатор задачи.
ATTACHMENT_ID	Идентификатор прикрепленного файла.

Пример вызова

```
$request = 'http://your-domain.ru/rest/task.item.deletefile.xml?auth=1iqeuq94vzfxu01bouws3voja2lsezfq&TASK_ID=3&ATTACHMENT_ID=28'
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Комментарии::task.commentit

Возвращает список методов вида **task.commentitem.*** и их описание.

Возвращаемое значение этого метода не предназначено для автоматической обработки, т.к. его формат может быть изменен без предупреждения.

Метод может быть полезен в качестве справочной информации, т.к. всегда содержит актуальную информацию.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
    'task.commentitem.getmanifest',
    [],
    function(result)
    {
        console.info(result.data());
        console.log(result);
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Комментарии::task.commentit

Описание и пример

Возвращает список комментариев к задаче.

Пример

```
// Получить все комментарии для задачи ID=1
с восходящей сортировкой по ID и фильтрацией
по AUTHOR_ID

BX24.callMethod(
        'task.commentitem.getlist',
        [1, {'ID': 'asc'}, {'>AUTHOR_ID':
2}],
        function(result){
                console.info(result.data());
                console.log(result);
        }
);
```

Параметры

Параметр	Описание

TASKID	Идентификатор задачи. Обязательный параметр.
ORDER	Массив для сортировки результата. Поле для сортировки может принимать значения: ▪ **ID** - идентификатор комментария; ▪ **AUTHOR_ID** - идентификатор автора комментария; ▪ **AUTHOR_NAME** - имя автора; ▪ **AUTHOR_EMAIL** - почтовый адрес автора; ▪ **POST_DATE** - дата публикации комментария. Направление сортировки может принимать значения: ▪ **asc** - по возрастанию; ▪ **desc** - по убыванию; Необязательный. По умолчанию фильтруется по убыванию идентификатора комментария.
FILTER	Массив вида *{"фильтруемое_поле": "значение фильтра" [, ...]}*. Фильтруемое поле может принимать значения: ▪ **ID** - идентификатор комментария; ▪ **AUTHOR_ID** - идентификатор автора комментария; ▪ **AUTHOR_NAME** - имя автора; ▪ **POST_DATE** - дата публикации комментария. Перед названием фильтруемого поля может указать тип фильтрации: ▪ "!" - не равно; ▪ "<" - меньше; ▪ "<=" - меньше либо равно; ▪ ">" - больше; ▪ ">=" - больше либо равно. *"значения фильтра"* - одиночное значение или массив.

	Необязательный. По умолчанию записи не фильтруются.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Комментарии::task.commentit

Возвращает комментарий к задаче.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
ITEMID	Идентификатор комментария. Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.commentitem.get',
   [13, 1205],
   function(result){
      console.info(result.data());
      console.log(result);
```

```
    }
);
```

Комментарии::task.commentit

Создает новый комментарий к задаче. Возвращает идентификатор добавленного комментария.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
FIELDS	Массив полей данных по задаче (**POST_MESSAGE**). Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Поля	Описание	Версия
AUTHOR_ID	Идентификатор пользователя, от имени которого создается комментарий.	
AUTHOR_NAME	Имя пользователя (опционально).	
AUTHOR_EMAIL	E-mail пользователя (опционально).	

USE_SMILES	(Y|N) - парсить или нет комментарии на наличие смайлов.	
POST_MESSAGE	Текст сообщения.	
UF_FORUM_MESSAGE_DOC	Массив файлов с диска для прикрепления вида `['n123', ...]`	

Пример

```
// Добавляем новый комментарий с текстом
"HELLO" для задачи с ID=13

BX24.callMethod(
    'task.commentitem.add',
    [13, {'POST_MESSAGE': 'HELLO'}],
    function(result){
      console.info(result.data());
      console.log(result);
    }
);
```

Комментарии::task.commentit

Обновляет данные комментария. Требуется обязательная авторизация через oauth и получение auth кода.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
ITEMID	Идентификатор комментария. Обязательный параметр.
FIELDS	Массив полей данных по комментарию (**POST_MESSAGE**). Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
// Обновить комментарий с ID=1205, задав
текст "HI"

BX24.callMethod(
```

```
        'task.commentitem.update',
        [13, 1205, {'POST_MESSAGE': 'HI'}],
        function(result){
                console.info(result.data());
                console.log(result);
        }
);
```

Комментарии::task.commentit

Удаляет комментарий. Требуется обязательная авторизация через oauth и получение auth кода.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
ITEMID	Идентификатор комментария. Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.commentitem.delete',
   [13, 1205],
   function(result){
      console.info(result.data());
      console.log(result);
```

```
    }
);
```

Комментарии::task.commentit

Проверяет, разрешено ли действие.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
ITEMID	Идентификатор комментария. Обязательный параметр.
ACTIONID	Идентификатор проверяемого действия: ▪ **1** - ACTION_COMMENT_ADD; ▪ **2** - ACTION_COMMENT_MODIFY; ▪ **3** - ACTION_COMMENT_REMOVE. Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
// Для комментария с ID=1205 проверяем,
разрешено ли действие удаления

BX24.callMethod(
   'task.commentitem.isactionallowed',
   [13, 1205, 3],
   function(result){
      console.info(result.data());
      console.log(result);
   }
);
```

task.dependence.add

```
task.dependence.add(taskIdFrom, taskIdTo, linkType)
```

Метод создаёт зависимость одной задачи от другой.

Параметры

Параметр	Описание	С версии
taskIdFrom	Идентификатор задачи от которой создаётся зависимость	
taskIdTo	Идентификатор задачи для которой создаётся зависимость	

linkType	Тип зависимости:
	`const LINK_TYPE_START_START = 00; // Связь старт-старт`
	`const LINK_TYPE_START_FINISH = 01; // Связь старт-финиш`
	`const LINK_TYPE_FINISH_START = 02; // Связь финиш-старт`
	`const`	

	`LINK_TYPE_FINISH_FINISH = 03; // Связь финиш-финиш`	

Тип зависимости может передаваться просто цифрами. Но в случае обращения со стороны PHP рекомендуется `ProjectDependenceTable::LINK_TYPE_START_START`, то есть использовать значение константы.

task.dependence.delete

```
task.dependence.delete(taskIdFrom, taskIdTo, linkType)
```

Метод удаляет зависимость одной задачи от другой.

Параметры

Параметр	Описание	С версии
taskIdFrom	Идентификатор задачи от которой удаляется зависимость	
taskIdTo	Идентификатор задачи для которой удаляется зависимость	

linkType	Тип зависимости:
	`const LINK_TYPE_START_START = 0x00; // Связь старт-старт`
	`const LINK_TYPE_START_FINISH = 0x01; // Связь старт-финиш`
	`const LINK_TYPE_FINISH_START = 0x02; // Связь финиш-старт`
	`const`	

	LINK_TYPE_FINISH_FINISH = 0x03; // Связь финиш-финиш	

task.stages.add

Метод добавления стадии Канбана / Моего плана. Принимает на вход массив fields.

Поля массива

Метод	Описание	С версии
TITLE	Заголовок стадии	
COLOR	Цвет стадии	
AFTER_ID	Идентификатор стадии, после которой надо добавить. Если не указано или равно 0, добавится в начало.	
ENTITY_ID	Идентификатор сущности. Может равняться ID группы, тогда стадия добавится в Канбан группы При недостаточном уровне прав выводится ошибка доступа. Если равняется 0 или отсутствует, то стадия добавляется в Мой план текущего пользователя.	
isAdmin	Если установлено *true*, то проверки прав происходить не будет. При условии, что	

	запрашивающий является admином портала.	

Возвращает ID добавленной стадии.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

task.stages.canmovetask

Определяет, может ли текущий пользователь перемещать задачи в указанной сущности.

Параметры

Метод	Описание	С версии
entityId	ID сущности	
entityType	Тип сущности (U - пользователь, G - группа). В случае U (Мой план) *true* вернется только в одном случае, если в entityId передаётся идентификатор текущего пользователя.	

Возвращает true/false

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

task.stages.delete

Метод удаления стадии Канбана / Моего плана. Принимает на вход id стадии.

Стадия проверяется на достаточный уровень прав, а также на то, что в ней нет задач.

Параметры:

Параметр	Описание	С версии
isAdmin	Если установлено *true*, то проверки прав происходить не будет. При условии, что запрашивающий является админом портала.	

Возвращает true в случае успеха.

task.stages.get

Метод получения стадий Канбана / Моего плана.

Параметры:

Параметр	Описание	С версии
entityId	Идентификатор сущности. Если равняется ID группы, то возвращаются стадии Канбана группы. При недостаточном уровне прав выводится ошибка доступа. Если параметр равен 0, то возвращаются стадии Моего плана текущего пользователя.	
isAdmin	Если установлено *true*, то проверки прав происходить не будет. При условии, что запрашивающий является админом портала.	

Возвращает массив стадий, поля описаны в таблице стадий.

task.stages.movetask

Метод перемещения задачи из одной стадии в другую.

Параметры

Метод	Описание	С версии
id	Идентификатор задачи	
stageId	ID стадии, в которую надо переместить задачу	
before	ID задачи, перед которой надо поставить задачу в стадии.	
after	ID задачи, после которой надо поставить задачу в стадии.	

Примечание. Если параметры before и after не переданы одновременно, то задача добавляется в колонке согласно настройкам проекта/моего плана. В ином случае before и after взаимоисключающие. Указывается по необходимости или тот или другой параметр.

Метод работает следующим образом. Если передана стадия группы, перемещение происходит в рамках Канбана группы. Если передана стадия Моего плана, перемещение происходит в нем. Перед перемещением происходит проверка прав.

task.stages.update

Метод обновления стадии Канбана / Моего плана. Принимает на вход id стадии и массив fields.

Параметры

Метод	Описание	С версии
id	Идентификатор стадии	
fields	Массив для обновления, аналогичный массиву task.stages.add, за исключением поля ENTITY_ID - его менять нельзя. Проверяется проверка прав доступа аналогичная task.stages.add.	
isAdmin	Если установлено *true*, то проверки прав происходить не будет. При условии, что запрашивающий является админом портала.	

Метод также применяется для перемещения стадии с одной позиции на другую - для этого достаточно передать нужный AFTER_ID.

Возвращает true в случае успеха.

© «Битрикс», 2001-2008, «1С-

Чек-листы::task.checklistitem.getm

Возвращает список методов вида **task.checklistitem.*** и их описание.

Возвращаемое значение этого метода не предназначено для автоматической обработки, т.к. его формат может быть изменен без предупреждения.

Метод может быть полезен в качестве справочной информации, т.к. всегда содержит актуальную информацию.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
    'task.checklistitem.getmanifest',
    [],
    function(result)
    {
        console.info(result.data());
        console.log(result);
    }
);
```

Чек-листы::task.checklistitem.getli

Возвращает список элементов чек-листа в задаче.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
ORDER	Массив для сортировки результата. Поле для сортировки может принимать значения: ▪ **ID** - идентификатор элемента чек-листа; ▪ **CREATED_BY** - идентификатор пользователя, создавшего элемент; ▪ **TOGGLED_BY** - идентификатор пользователя, изменившего состояние элемента чек-листа; ▪ **TOGGLED_DATE** - время, когда было изменено состояние элемента чек-листа; ▪ **TITLE** - заголовок элемента чек-листа; ▪ **SORT_INDEX** - индекс сортировки элемента; ▪ **IS_COMPLETE** - элемент отмечен как выполненный; Направление сортировки может принимать значения: ▪ **asc** - по возрастанию;

	▪ **desc** - по убыванию; Необязательный. По умолчанию фильтруется по убыванию идентификатора элемента чек-листа.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.checklistitem.getlist',
   [13, {'TOGGLED_DATE': 'desc'}],
   function(result){
      console.info(result.data());
      console.log(result);
   }
);
```

Чек-листы::task.checklistitem.get

Возвращает элемент чек-листа по его идентификатору.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
ITEMID	Идентификатор элемента. Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.checklistitem.get',
   [13, 20],
   function(result){
      console.info(result.data());
      console.log(result);
```

```
    }
);
```

Чек-листы::task.checklistitem.add

Добавление нового элемента чек-листа к задаче. Возвращает идентификатор добавленного элемента

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
FIELDS	Массив полей элемента чек-листа (**TITLE**, **SORT_INDEX**, **IS_COMPLETE**). Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
// Добавляем к задаче с ID=13 новый "выполненный" элемент чек-листа с текстом "Пункт выполнен"
```

```
BX24.callMethod(
   'task.checklistitem.add',
   [13, {'TITLE': 'Пункт выполнен',
'IS_COMPLETE': 'Y'}],
   function(result){
      console.info(result.data());
      console.log(result);
   }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Чек-листы::task.checklistitem.upda

Обновляет данные элемента чек-листа.

Рекомендуется перед обновлением данных проверять, допустимо ли данное действие (task.checklistitem.isactionallowed).

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
ITEMID	Идентификатор элемента чек-листа. Обязательный параметр.
FIELDS	Массив полей полей элемента чек-листа (**TITLE**, **SORT_INDEX**, **IS_COMPLETE**). Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
// Обновляем для элемента с ID=25 состояние
на "невыполнен", а текст на "Пункт не
выполнен"

BX24.callMethod(
   'task.checklistitem.update',
   [13, 25, {TITLE: 'Пункт не выполнен',
'IS_COMPLETE': 'N'}],
   function(result){
      console.info(result.data());
      console.log(result);
   }
);
```

Чек-листы::task.checklistitem.delet

Удаляет элемент чек-листа.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
ITEMID	Идентификатор элемента чек-листа. Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.checklistitem.delete',
   [13, 20],
   function(result){
      console.info(result.data());
      console.log(result);
```

```
    }
);
```

Чек-листы::task.checklistitem.comp

Отмечает элемент чек-листа как выполненный.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
ITEMID	Идентификатор элемента чек-листа. Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.checklistitem.complete',
   [13, 21],
   function(result){
      console.info(result.data());
      console.log(result);
```

```
    }
);
```

Смотрите также

- task.checklistitem.renew

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Чек-листы::task.checklistitem.rene

Отмечает выполненный элемент чек-листа как вновь активный.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
ITEMID	Идентификатор элемента чек-листа. Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.checklistitem.renew',
   [13, 21],
   function(result){
      console.info(result.data());
      console.log(result);
```

```
    }
);
```

Смотрите также

- task.checklistitem.complete

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Чек-листы::task.checklistitem.mov

Помещает элемент чек-листа в списке после указанного.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
ITEMID	Идентификатор элемента чек-листа. Обязательный параметр.
AFTERITEMID	Идентификатор элемента чек-листа, после которого будет помещен заданный элемент. Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.checklistitem.moveafteritem',
```

```
    [13, 21, 9],
    function(result){
       console.info(result.data());
       console.log(result);
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Чек-листы::task.checklistitem.isact

Проверяет, разрешено ли действие.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
ITEMID	Идентификатор элемента чек-листа. Обязательный параметр.
ACTIONID	Идентификатор проверяемого действия: ▪ **1** - ACTION_ADD; ▪ **2** - ACTION_MODIFY; ▪ **3** - ACTION_REMOVE; ▪ **4** - ACTION_TOGGLE. Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
// Для элемента с ID=21 проверяем, разрешено
ли действие его изменения

BX24.callMethod(
   'task.checklistitem.isactionallowed',
   [13, 21, 2],
   function(result){
      console.info(result.data());
      console.log(result);
   }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

task.elapseditem.getmanifest

Возвращает список методов вида **task.elapseditem.*** и их описание.

Возвращаемое значение этого метода не предназначено для автоматической обработки, т.к. его формат может быть изменен без предупреждения.

Метод может быть полезен в качестве справочной информации, т.к. всегда содержит актуальную информацию.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

```
BX24.callMethod(
        'task.elapseditem.getmanifest',
        [],
        function(result)
        {
                console.info(result.data());
                console.log(result);
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

task.elapseditem.getlist

Описание

Возвращает список записей о затраченном времени по задаче.

Параметры

Параметр	Описание
TASKID	Идентификатор задачи. Необязательный параметр.
ORDER	Массив для сортировки результата. Поле для сортировки может принимать значения: ▪ **ID** - идентификатор записи о затраченном времени; ▪ **USER_ID** - идентификатор пользователя, от имени которого была сделана запись о затраченном времени; ▪ **MINUTES** - затраченное время, минуты; ▪ **SECONDS** - затраченное время, секунды; ▪ **CREATED_DATE** - дата создания записи; ▪ **DATE_START** - дата начала; ▪ **DATE_STOP** - дата конца. Направление сортировки может принимать значения: ▪ **asc** - по возрастанию; ▪ **desc** - по убыванию;

	Необязательный. По умолчанию фильтруется по убыванию идентификатора записи о затраченном времени.
FILTER	Массив вида *{"фильтруемое_поле": "значение фильтра" [, ...]}*. Фильтруемое поле может принимать значения: ▪ **ID** - идентификатор комментария; ▪ **USER_ID** - идентификатор пользователя, от имени которого была сделана запись о затраченном времени; ▪ **CREATED_DATE** - дата создания записи; Перед названием фильтруемого поля может указать тип фильтрации: ▪ "!" - не равно; ▪ "<" - меньше; ▪ "<=" - меньше либо равно; ▪ ">" - больше; ▪ ">=" - больше либо равно. *"значения фильтра"* - одиночное значение или массив. Необязательный. По умолчанию записи не фильтруются.
SELECT	Массив полей записей, которые будут возвращены методом. Можно указать только те поля, которые необходимы. Если в массиве присутствует значение `"*"`, то будут возвращены все доступные поля. Значение по умолчанию - пустой массив `array()` - означает, что будут возвращены все поля основной таблицы запроса.
PARAMS	Массив для опций вызова. Элементом является массив NAV_PARAMS вида *{"опция вызова": 'значение' [, ...]}*, хранящий следующие опции: ▪ **nPageSize** - количество элементов на странице. В целях ограничения нагрузки

	на постраничную навигацию наложено ограничение в 50 записей. ▪ **iNumPage** - номер страницы при постраничной навигации.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Примеры

```
// Получить все записи о затраченном времени
с сортировкой по ID в нисходящем порядке.
// Будут отфильтрованы только те записи, ID
которых имеет значение меньше 50.

BX24.callMethod(
   'task.elapseditem.getlist',
   [1, {'ID': 'desc'}, {'<ID': 50}],
   function(result){
      console.info(result.data());
      console.log(result);
   }
);
```

Получение выборки по затраченному времени на основании общий условий фильтрации. Например, выбрать данные о трудозатратах с указанной даты:

```
BX24.callMethod(
  'task.elapseditem.getlist',
  [{'ID': 'desc'}, {'>=CREATED_DATE': '2018-
02-16'}],
  function(result){
     console.info(result.data());
```

```
      console.log(result);
   }
);
```

Пример для работы с php:

```
// Пример для работы с php
// Получение GET-запроса на выборку данных.
$appParams = array(
    "auth" =>
'92006f4ae0c55d400f1e6e09428af64a',
    "ORDER" => array("ID" => "DESC"),
// Сортировка по ID - по убыванию
    "FILTER" => array(">ID" => 1),
// Фильтр
    "SELECT" => array('ID', 'TASK_ID'),
// Выборка - только ID записи и задачи
    "PARAMS" => array('NAV_PARAMS' => array(
// Постраничка
        "nPageSize" => 2,
// по 2 элемента на странице
        'iNumPage' => 2
// страница номер 2
    )),
);

$appRequestUrl = 'http://test-
domain.ru/rest/task.elapseditem.getlist.xml?
'.http_build_query($appParams);

print(urldecode($appRequestUrl));;
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

task.elapseditem.get

Возвращает запись о затраченном времени по ее идентификатору.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи. Обязательный параметр.
ITEMID	Идентификатор записи. Обязательный параметр.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.elapseditem.get',
   [13, 217],
   function(result){
      console.info(result.data());
      console.log(result);
```

```
    }
);
```

task.elapseditem.add

```
task.elapseditem.add(taskId, arFields)
```

Добавляет затраченное время к задаче. Возвращает идентификатор добавленной записи.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.
ARFIELDS	Массив, содержащий записи о пользователе, времени и комментарии (**SECONDS** , **COMMENT_TEXT**, **USER_ID** и **CREATED_DATE**).

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.elapseditem.add',
   [1, {SECONDS: 113, COMMENT_TEXT: 'текст
комментария', CREATED_DATE: '2016-01-20
17:26:37'}],
   function(result)
        {
      console.info(result.data());
      console.log(result);
        }
);
```

Пример добавления времени

```
BX24.callMethod(
  'task.elapseditem.add',
  [315, {SECONDS: 113, COMMENT_TEXT:
'добавили из реста', CREATED_DATE: '2018-02-
16 17:26:37', USER_ID: 6}],
  function(result)
{
     console.info(result.data());
     console.log(result);
}
);
```

task.elapseditem.update

```
task.elapseditem.update(taskId, itemId,
arFields)
```

Изменяет параметры указанной записи о затраченном времени.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.
ITEMID	Идентификатор записи о затраченном времени.
ARFIELDS	Массив, содержащий записи о времени и комментарии (**SECONDS** , **COMMENT_TEXT** и **CREATED_DATE**). Допустимо использовать **MINUTES** вместо **SECONDS**, но нельзя их использовать одновременно.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.elapseditem.update',
   [1, 204, {SECONDS: 666, COMMENT_TEXT:
'текст комментария', CREATED_DATE: '2016-01-
20 17:26:37'}],
   function(result)
        {
      console.info(result.data());
      console.log(result);
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

task.elapseditem.delete

```
task.elapseditem.delete(taskId, itemId)
```

Удаляет запись о затраченном времени.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.
ITEMID	Идентификатор записи о затраченном времени.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.elapseditem.delete',
   [1, 203],
   function(result)
```

```
        {
      console.info(result.data());
      console.log(result);
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

task.elapseditem.isactionallow

```
task.elapseditem.isactionallowed(taskId,
itemId, actionId)
```

Проверяет, разрешено ли действие.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.
ITEMID	Идентификатор записи о затраченном времени.
ACTIONID	Идентификатор действия: ▪ **1** - ACTION_ELAPSED_TIME_ADD; ▪ **2** - ACTION_ELAPSED_TIME_MODIFY; ▪ **3** - ACTION_ELAPSED_TIME_REMOVE.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.elapseditem.isActionAllowed',
   [1, 204, 2],
   function(reply)
        {
      var isAllowed = reply.data();      //
или так (аналогично): var isAllowed =
reply.answer.result;
        }
);
```

task.planner.getlist

Возвращает массив, содержащий идентификаторы задач в плане на день.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
    'task.planner.getlist',
    [],
    function(result)
        {
        console.info(result.data());
        console.log(result);
        }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

task.item.userfield.getfields

```
$appParams = array(
     'auth' =>
'q21g8vhcqmxdrbhqlbd2wh6ev1debppa',
);
```

Метод возвращает все доступные поля свойства.

Параметры метода

Параметр	Описание
auth	Токен авторизации.

Примеры вызова

```
$request = 'http://your-
domain.ru/rest/task.item.userfield.getfields
.xml?'.http_build_query($appParams);'
```

```
BX24.callMethod(
        'task.item.userfield.getfields',

{'q21g8vhcqmxdrbhqlbd2wh6ev1debppa'},

        function(result)
        {
                console.info(result.data());
                console.log(result);
        }
);
```

task.item.userfield.gettypes

```
$appParams = array(
     'auth' =>
'q21g8vhcqmxdrbhqlbd2wh6ev1debppa',
);
```

Метод возвращает все доступные типы данных.

Параметры метода

Параметр	Описание
auth	Токен авторизации.

Примеры вызова

```
$request = 'http://your-
domain.ru/rest/task.item.userfield.gettypes.
xml?'.http_build_query($appParams);'
```

```
BX24.callMethod(
        'task.item.userfield.gettypes',

{'q21g8vhcqmxdrbhqlbd2wh6ev1debppa'},

        function(result)
        {
                console.info(result.data());
                console.log(result);
        }
);
```

task.item.userfield.add

Метод создает новое свойство.

Системное ограничение на название поля - 20 знаков. К названию пользовательского поля всегда добавляется префикс UF_TASK_, то есть реальная длина названия - 12 знаков.

Параметры метода

Параметр	Описание
auth	Токен авторизации.
PARAMS	Массив с параметрами свойства вида array("параметр": 'значение' [, ...]), содержащий следующие параметры: ▪ **USER_TYPE_ID** - тип данных пользовательского поля. Допустимые значения: ▪ `string` (**Строка**); ▪ `double` (**Число**); ▪ `date` (**Дата**); ▪ `boolean` (**Да/Нет**); ▪ **FIELD_NAME** - код поля; ▪ **XML_ID** - внешний код; ▪ **EDIT_FORM_LABEL** - подпись в форме форматирования (указывается на английском ('en') и русском ('ru") языках; ▪ **LABEL** - заголовок поля.

Примеры вызова.

```
$appParams = array(
          'auth' =>
'q21g8vhcqmxdrbhqlbd2wh6ev1debppa',
          'PARAMS' => array(
                    'USER_TYPE_ID' =>
'string',
                    'FIELD_NAME' =>
'NEW_TASKS_FIELD',
                    'XML_ID' =>
'MY_TASK_FIELD',
                    'EDIT_FORM_LABEL' =>
array(
                        'en' => 'New task
field',
                        'ru' => 'Новое поле
задач'
                    ),
                    'LABEL' => 'New task
field'
           ),
);

$request = 'http://your-
domain.ru/rest/task.item.userfield.add.xml?'
.http_build_query($appParams);'
```

```
BX24.callMethod(
    'task.item.userfield.add',
    {
        PARAMS:
        {
            'USER_TYPE_ID' : 'string',
            'FIELD_NAME' :
'NEW_TASKS_FIELD',
            'XML_ID' : 'MY_TASK_FIELD',
            'EDIT_FORM_LABEL' : {'en':'New
task field', 'ru':'Новое поле задач'},
            'LABEL' : 'New task field'
        }
    },
    function(result)
    {
        console.info(result.data());
        console.log(result);
    }
);
```

task.item.userfield.get

Метод возвращает свойство по идентификатору.

Параметры метода

Параметр	Описание
auth	Токен авторизации.
ID	Идентификатор пользовательского поля.

Пример

```
$appParams = array(
     'auth' =>
'q21g8vhcqmxdrbhqlbd2wh6ev1debppa',
     'ID' => 77
);
```

Примеры вызова

```
$request = 'http://your-
domain.ru/rest/task.item.userfield.get.xml?'
```

```
.http_build_query($appParams);
```

```
BX24.callMethod(
        'task.item.userfield.get',
        {'q21g8vhcqmxdrbhqlbd2wh6ev1debppa',
77},

        function(result)
        {
                console.info(result.data());
                console.log(result);
        }
);
```

task.item.userfield.getlist

Метод возвращает список свойств.

Параметры метода

Параметр	Описание
auth	Токен авторизации.
ORDER	Массив для сортировки результата. Массив вида array("поле сортировки"=>"направление сортировки" [, ...]).
FILTER	Массив фильтрации результата вида array("фильтруемое поле"=>"значение фильтра" [, ...]). Обязательный параметр.

Пример

```
$appParams = array(
     'auth' =>
'q21g8vhcqmxdrbhqlbd2wh6ev1debppa',
     'ORDER' => array('ID' => 'asc'),
     'FILTER' => array('USER_TYPE_ID' =>
'string')
);
```

Примеры вызова

```
$request = 'http://your-
domain.ru/rest/task.item.userfield.getlist.x
ml?'.http_build_query($appParams);
```

```
BX24.callMethod(
    "task.item.userfield.getlist",
    {
        order:
        {
            "ID": "ASC"
        },
        filter:
        {
            "EDIT_IN_LIST": "Y"
        }
    },
    function(result)
    {
    }
);
```

task.item.userfield.update

Метод используется для редактирования параметров свойства.

Параметры метода

Параметр	Описание
auth	Токен авторизации.
ID	Идентификатор пользовательского поля.
DATA	Массив array("поле"=>"значение", ...). Содержит значения редактируемых параметров.

Пример

```
$appParams = array(
    'auth' =>
'q21g8vhcqmxdrbhqlbd2wh6ev1debppa',
    'ID' => 77
    'DATA' => array('XML_ID' =>
'new_external_id')
);
```

Примеры вызова

```
$request = 'http://your-
domain.ru/rest/task.item.userfield.update.xm
l?'.http_build_query($appParams);
```

```
BX24.callMethod(
        'task.item.userfield.update',
        {'q21g8vhcqmxdrbhqlbd2wh6ev1debppa',
77, ['XML_ID': 'new_external_id']},

        function(result)
        {
                console.info(result.data());
                console.log(result);
        }
);
```

task.item.userfield.delete

Метод удаляет свойство.

Параметры метода

Параметр	Описание
auth	Токен авторизации.
ID	Идентификатор пользовательского поля.

Пример

```
$appParams = array(
    'auth' =>
'q21g8vhcqmxdrbhqlbd2wh6ev1debppa',
    'ID' => 77
);
```

Примеры вызова

```
$request = 'http://your-
domain.ru/rest/task.item.userfield.delete.xm
l?'.http_build_query($appParams);
```

```
BX24.callMethod(
        'task.item.userfield.delete',
        {'q21g8vhcqmxdrbhqlbd2wh6ev1debppa',
77},

        function(result)
        {
                console.info(result.data());
                console.log(result);
        }
);
```

tasks.api.scrum.backlog.add

```
tasks.api.scrum.backlog.add(fields)
```

Метод добавляет бэклог в Скрам.

Может понадобиться, чтобы явно создать бэклог при импорте после создания Скрама.

Параметры

Параметр	Описание
fields	Поля, соответствующие доступному списку полей tasks.api.scrum.backlog.getFields. Поле **groupId** обязательное.

Примеры

```
const groupId = 1;
const createdBy = 1;

BX24.callMethod(
```

```
    'tasks.api.scrum.backlog.add',
    {
        fields:{
           groupId: groupId,
           createdBy: createdBy,
        }
    },
    function(res)
    {
       console.log(res);
    }
);
```

tasks.api.scrum.backlog.delete

```
tasks.api.scrum.backlog.delete(id)
```

Метод удаляет бэклог.

В обычной ситуации удалять бэклог не нужно. При удалении бэклога *Битрикс24* автоматически создаст его заново при открытии страницы планирования в задачах Скрама.

Метод используется, если бэклог добавлен в какую то группу/проект ошибочно.

Параметры

Параметр	Описание
id	Идентификатор бэклога. Обязательный параметр.

Примеры

```
const backlogId = 1;
```

```
BX24.callMethod(
   'tasks.api.scrum.backlog.delete',
   {
      id: backlogId
   },
   function(res)
   {
      console.log(res);
   }
);
```

tasks.api.scrum.backlog.get

```
tasks.api.scrum.backlog.get(groupId)
```

Метод возвращает значения полей бэклога по id Скрама.

Может понадобиться, чтобы получить id бэклога для добавления или переноса в бэклог задачи.

Параметры

Параметр	Описание
groupId	Идентификатор Скрама.

Примеры

```
const groupId = 1;

BX24.callMethod(
   'tasks.api.scrum.backlog.get',
   {
       id: groupId,
```

```
    },
    function(res)
    {
       console.log(res);
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

tasks.api.scrum.backlog.getFie

```
tasks.api.scrum.backlog.getFields()
```

Метод возвращает доступные поля бэклога.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
   'tasks.api.scrum.backlog.getFields',
   {},
   function(res)
   {
      console.log(res);
   }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

tasks.api.scrum.backlog.update

```
tasks.api.scrum.backlog.update(id, fields)
```

Метод изменяет бэклог.

Параметры

Параметр	Описание
id	Идентификатор бэклога. Обязательный параметр.
fields	Поля, соответствующие доступному списку полей tasks.api.scrum.backlog.getFields.

Примеры

```
const backlogId = 1;
const groupId = 1;
const createdBy = 1;
const modifiedBy = 1;
```

```
BX24.callMethod(
   'tasks.api.scrum.backlog.update',
   {
      id: backlogId,
      fields:{
         groupId: groupId,
         createdBy: createdBy,
         modifiedBy: modifiedBy,
      }
   },
   function(res)
   {
      console.log(res);
   }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

tasks.api.scrum.sprint.add

```
tasks.api.scrum.sprint.add(fields)
```

Метод добавляет спринт в Скрам.

Параметры

Параметр	Описание
fields	Поля, соответствующие доступному списку полей tasks.api.scrum.sprint.getFields. Поле **groupId** обязательное. Доступные значения для полей дат **dateStart** и **dateEnd**: ('ISO 8601', timestamp). Доступные значения для поля **status**: ('active', 'planned', 'completed').

Примеры

```
const groupId = 1;
const name = 'Sprint 1';
const createdBy = 1;
const sort = 1;
const status = 'planned';
const dateStart = '2021-11-
22T00:00:00+02:00';
const dateEnd = '2021-11-29T00:00:00+02:00';

BX24.callMethod(
   'tasks.api.scrum.sprint.add',
   {
       fields:{
         name: name,
         groupId: groupId,
         createdBy: createdBy,
         sort: sort,
         status: status,
         dateStart: dateStart,
         dateEnd: dateEnd,
       }
   },
   function(res)
   {
      console.log(res);
   }
);
```

tasks.api.scrum.sprint.complet

```
tasks.api.scrum.sprint.complete(id)
```

Метод завершает активный спринт выбранного Скрама.

При завершении спринта незавершенные задачи переносятся в бэклог.

Параметры

Параметр	Описание
id	Идентификатор Скрама. Обязательный параметр.

Примеры

```
const groupId = 1;

BX24.callMethod(
   'tasks.api.scrum.sprint.complete',
   {
```

```
        id: groupId
    },
    function(res)
    {
        console.log(res);
    }
);
```

tasks.api.scrum.sprint.delete

```
tasks.api.scrum.sprint.delete(id)
```

Метод удаляет спринт.

При удалении спринта с задачами задачи будут перемещены в бэклог.

Параметры

Параметр	Описание
id	Идентификатор спринта. Обязательный параметр.

Примеры

```
const sprintId = 1;

BX24.callMethod(
   'tasks.api.scrum.sprint.delete',
   {
```

```
        id: sprintId
    },
    function(res)
    {
        console.log(res);
    }
);
```

tasks.api.scrum.sprint.get

```
tasks.api.scrum.sprint.get(sprintId)
```

Метод возвращает значения полей спринта по его id.

Параметры

Параметр	Описание
sprintId	Идентификатор спринта.

Примеры

```
const sprintId = 2;

BX24.callMethod(
   'tasks.api.scrum.sprint.get',
   {
       id: sprintId,
   },
   function(res)
```

```
    {
        console.log(res);
    }
);
```

tasks.api.scrum.sprint.getField

```
tasks.api.scrum.sprint.getFields()
```

Метод возвращает доступные поля спринта.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
   'tasks.api.scrum.sprint.getFields',
   {},
   function(res)
   {
      console.log(res);
   }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

tasks.api.scrum.sprint.list

Описание и пример

```
tasks.api.scrum.sprint.list()
```

Метод возвращает список спринтов.

Метод аналогичен другим методам с фильтрацией по списку.

Можно передавать значения полей в **filter**, **select**, **order**.

Параметры

Без параметров.

Примеры

```
const groupId = 1;

BX24.callMethod(
   'tasks.api.scrum.sprint.list',
   {
      filter: {
```

```
        GROUP_ID: groupId,
        '>=DATE_END': new Date()
        }
    },
    function(res)
    {
        console.log(res);
    }
);
```

Доступные поля

Поле	Описание
ID	Идентификатор спринта
GROUP_ID	Идентификатор Скрама
ENTITY_TYPE	Тип элемента
NAME	Имя
SORT	Сортировка
CREATED_BY	Кем создан
MODIFIED_BY	Кем изменён
DATE_START	Дата запуска
DATE_END	Дата окончания
STATUS	Статус
INFO	Информация

© «Битрикс», 2001-2008, «1С-Битрикс»

tasks.api.scrum.sprint.start

```
tasks.api.scrum.sprint.start(id)
```

Метод запускает спринт.

Запустить можно только планируемый спринт.

При запуске спринта колонки активного спринта и роботы будут перенесены из предыдущего завершенного спринта, если он есть.

Задачи будут добавлены в канбан активного спринта. Если в спринте в этот момент были завершенные задачи, они будут перенесены в бэклог.

Параметры

Параметр	Описание
id	Идентификатор спринта. Обязательный параметр.

Примеры

```
const sprintId = 2;

BX24.callMethod(
   'tasks.api.scrum.sprint.start',
   {
      id: sprintId
   },
   function(res)
   {
      console.log(res);
   }
);
```

tasks.api.scrum.sprint.update

```
tasks.api.scrum.sprint.update(fields)
```

Метод изменяет спринт.

Все поля спринта доступны для обновления. Необновляемые поля можно не передавать.

Параметры

Параметр	Описание
fields	Поля, соответствующие доступному списку полей tasks.api.scrum.sprint.getFields. Доступные значения для полей дат **dateStart** и **dateEnd**: ('ISO 8601', timestamp). Доступные значения для поля **status**: ('active', 'planned', 'completed').

Примеры

```
const sprintId = 2;
const groupId = 1;
const name = 'Sprint 2';
const dateStart = '2021-11-
22T00:00:00+02:00';
const dateEnd = '2021-11-29T00:00:00+02:00';

BX24.callMethod(
   'tasks.api.scrum.sprint.update',
   {
       id: sprintId,
       fields:{
         name: name,
         groupId: groupId,
         dateStart: dateStart,
         dateEnd: dateEnd,
       }
   },
   function(res)
   {
      console.log(res);
   }
);
```

tasks.api.scrum.epic.add

```
tasks.api.scrum.epic.add(fields)
```

Метод добавляет эпик в Скрам.

Параметры

Параметр	Описание
fields	Поля, соответствующие доступному списку полей tasks.api.scrum.epic.getFields. Поля **name** и **groupId** обязательные. В поле **files** можно передать массив значений с идентификаторами файлов, указав префикс `n`.

Примеры

```
const groupId = 1;
const name = 'Epic 1';
```

```
const description = 'Description text';
const color = '#69dafc';
const files = ['n428'];

BX24.callMethod(
   'tasks.api.scrum.epic.add',
   {
       fields:{
         name: name,
         groupId: groupId,
         color: color,
         files: files
       }
   },
   function(res)
   {
      console.log(res);
   }
);
```

tasks.api.scrum.epic.delete

```
tasks.api.scrum.epic.delete(id)
```

Метод удаляет эпик.

Параметры

Параметр	Описание
id	Идентификатор эпика. Обязательный параметр.

Примеры

```
const epicId = 1;

BX24.callMethod(
   'tasks.api.scrum.epic.delete',
   {
      id: epicId
   },
   function(res)
```

```
    {
        console.log(res);
    }
);
```

tasks.api.scrum.epic.get

```
tasks.api.scrum.epic.get(epicId)
```

Метод возвращает значения полей эпика по его id.

Параметры

Параметр	Описание
epicId	Идентификатор эпика.

Примеры

```
const epicId = 1;

BX24.callMethod(
   'tasks.api.scrum.epic.get',
   {
       id: epicId,
   },
   function(res)
```

```
    {
        console.log(res);
    }
);
```

tasks.api.scrum.epic.getFields

```
tasks.api.scrum.epic.getFields()
```

Метод возвращает доступные поля эпика.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
   'tasks.api.scrum.epic.getFields',
   {},
   function(res)
   {
      console.log(res);
   }
);
```

tasks.api.scrum.epic.list

Описание и пример

```
tasks.api.scrum.epic.list()
```

Метод возвращает список эпиков.

Метод аналогичен другим методам с фильтрацией по списку.

Можно передавать значения полей в **filter**, **select**, **order**.

Параметры

Без параметров.

Примеры

```
const groupId = 1;

BX24.callMethod(
   'tasks.api.scrum.epic.list',
   {
      filter: {
```

```
        GROUP_ID: groupId
        }
    },
    function(res)
    {
        console.log(res);
    }
);
```

Доступные поля

Поле	Описание
ID	Идентификатор эпика
GROUP_ID	Идентификатор Скрама
NAME	Имя
DESCRIPTION	Описание
CREATED_BY	Кем создан
MODIFIED_BY	Кем изменён
COLOR	Цвет

tasks.api.scrum.epic.update

```
tasks.api.scrum.epic.update(id, fields)
```

Метод изменяет эпик в Скраме.

Все поля эпика доступны для обновления. Необновляемые поля можно не передавать.

Параметры

Параметр	Описание
id	Идентификатор эпика.
fields	Поля, соответствующие доступному списку полей tasks.api.scrum.epic.getFields. В поле **files** можно передать массив значений с идентификаторами файлов, указав префикс `n`. Если в **files** передать пустой массив, файлы удалятся.

Примеры

```
const epicId = 1;
const name = 'Updated epic name';
const description = 'Updated description
text';
const color = '#bbecf1';
const files = ['n429'];

BX24.callMethod(
   'tasks.api.scrum.epic.update',
   {
      id: epicId,
       fields:{
         name: name,
         description: description,
         color: color,
         files: files
       }
   },
   function(res)
   {
      console.log(res);
   }
);
```

tasks.api.scrum.task.get

```
tasks.api.scrum.task.get(id)
```

Метод возвращает значения полей задачи Скрама по её id.

Параметры

Параметр	Описание
id	Идентификатор задачи.

Примеры

```
const taskId = 1;

BX24.callMethod(
   'tasks.api.scrum.task.get',
   {
      id: taskId
   },
   function(res)
   {
      console.log(res);
```

```
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

tasks.api.scrum.task.getFields

Описание и пример

```
tasks.api.scrum.task.getFields()
```

Метод возвращает доступные поля задачи Скрама.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
   'tasks.api.scrum.task.getFields',
   {},
   function(res)
   {
      console.log(res);
   }
);
```

Доступные поля

Поле	Описание
entityId	Идентификатор бэклога или спринта
storyPoints	Стори Поинты (относительная оценка сложности задачи)
epicId	Идентификатор эпика
sort	Сортировка
createdBy	Кем создана
modifiedBy	Кем изменена

tasks.api.scrum.task.update

```
tasks.api.scrum.task.update(id, fields)
```

Метод создает или изменяет задачу Скрама.

Задачу нужно предварительно создать, либо получить уже существующую. Задачу нужно предварительно привязать к Скраму с помощью методов tasks.task.add / tasks.task.update, обновив поле `GROUP_ID`. *Битрикс24* автоматически добавляет ее в бэклог Скрама, если он существует.

Метод используется, если нужно создать задачу в Скраме, перенести задачу из другого проекта, перенести её между бэклогом и спринтами, изменить стори поинты, привязать эпик.

Параметры

Параметр	Описание
id	Идентификатор задачи.
fields	Поля, соответствующие доступному списку полей tasks.api.scrum.task.getFields.

Примеры

```
const taskId = 1;
const epicId = 1;
const storyPoints = '8';
const sprintId = 2;

BX24.callMethod(
   'tasks.api.scrum.task.update',
   {
      id: taskId,
       fields:{
         epicId: epicId,
         storyPoints: storyPoints,
         entityId: sprintId
       }
   },
   function(res)
   {
      console.log(res);
   }
);
```

tasks.api.scrum.kanban.addSta

```
tasks.api.scrum.kanban.addStage(fields)
```

Метод создаёт стадию канбана Скрама.

Параметры

Параметр	Описание
fields	Поля, соответствующие доступному списку полей tasks.api.scrum.kanban.getFields. Поля **name** и **sprintId** обязательные. Поле **sort** кратно 100. Доступные значения для поля **type**: (NEW, WORK, FINISH).

Примеры

```
const sprintId = 2;

let name = 'firstColumn';
let sort = 100;
let type = 'NEW';
let color = '00C4FB';

BX24.callMethod(
   'tasks.api.scrum.kanban.addStage',
{
   fields: {
      sprintId: sprintId,
      name: name,
      sort: sort,
      type: type,
      color: color,
   }
},
   function(res)
   {
      console.log(res);
   }
);

name = 'secondColumn';
sort = 200;
type = 'FINISH';
color = '75D900';

BX24.callMethod(
   'tasks.api.scrum.kanban.addStage',
{
   sprintId: sprintId,
   fields: {
      name: name,
      sort: sort,
      type: type,
```

```
        color: color,
    }
},
    function(res)
    {
        console.log(res);
    }
);
```

tasks.api.scrum.kanban.addTas

```
tasks.api.scrum.kanban.addTask(sprintId, taskId, stageId)
```

Метод добавляет задачу в канбан Скрама.

Параметры

Параметр	Описание
sprintId	Идентификатор спринта. Обязательный параметр.
taskId	Идентификатор задачи. Обязательный параметр.
stageId	Идентификатор стадии. Обязательный параметр.

Примеры

```
const sprintId = 1;
const taskId = 1;
const stageId = 1;

BX24.callMethod(
   'tasks.api.scrum.kanban.addTask',
   {
      sprintId: sprintId,
      taskId: taskId,
      stageId: stageId
   },
   function(res)
   {
      console.log(res);
   }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

tasks.api.scrum.kanban.delete

```
tasks.api.scrum.kanban.deleteStage(stageId)
```

Метод удаляет стадию.

Стадия не будет удалена, если в ней есть задачи.

Параметры

Параметр	Описание
stageId	Идентификатор стадии.

Примеры

```
const stageId = 1;

BX24.callMethod(
   'tasks.api.scrum.kanban.deleteStage',
   {
      stageId: stageId
   },
```

```
    function(res)
    {
        console.log(res);
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

tasks.api.scrum.kanban.delete

```
tasks.api.scrum.kanban.deleteTask(sprintId, taskId)
```

Метод удаляет задачу из канбана Скрама.

Параметры

Параметр	Описание
sprintId	Идентификатор спринта. Обязательный параметр.
taskId	Идентификатор задачи. Обязательный параметр.

Примеры

```
const sprintId = 1;
const taskId = 1;

BX24.callMethod(
```

```
    'tasks.api.scrum.kanban.deleteTask',
    {
       sprintId: sprintId,
       taskId: taskId
    },
    function(res)
    {
       console.log(res);
    }
);
```

tasks.api.scrum.kanban.getFiel

```
tasks.api.scrum.kanban.getFields()
```

Метод возвращает доступные поля стадии канбана.

Параметры

Без параметров

Примеры

```
BX24.callMethod(
   'tasks.api.scrum.kanban.getFields',
   {},
   function(res)
   {
      console.log(res);
   }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

tasks.api.scrum.kanban.getSta

```
tasks.api.scrum.kanban.getStages(sprintId)
```

Метод возвращает стадии канбана по id спринта.

Параметры

Параметр	Описание
sprintId	Идентификатор спринта.

Примеры

```
const sprintId = 2;

BX24.callMethod(
   'tasks.api.scrum.kanban.getStages',
   {
      sprintId: sprintId
   },
   function(res)
```

```
    {
        console.log(res);
    }
);
```

tasks.api.scrum.kanban.update

```
tasks.api.scrum.kanban.updateStage(stageId, fields)
```

Метод изменяет стадию канбана Скрама.

Параметры

Параметр	Описание
stageId	Идентификатор стадии.
fields	Поля, соответствующие доступному списку полей tasks.api.scrum.kanban.getFields. Доступные для обновления поля: (name, sprintId, sort, type, color). Необновляемые поля можно не передавать. Поле **sort** кратно 100. Доступные значения поля для **type**: (NEW, WORK, FINISH).

Примеры

```
const stageId = 2;

const name = 'Updated name';
const sort = 200;
const type = 'WORK';
const color = '47D1E2';

BX24.callMethod(
   'tasks.api.scrum.kanban.updateStage',
{
   stageId: stageId,
   fields: {

      name: name,
      sort: sort,
      type: type,
      color: color,
   }
},
   function(res)
   {
      console.log(res);
   }
);
```

task.items.getlist

Описание

Возвращает массив задач, каждая из которых содержит массив полей (аналогичен массиву, возвращаемому task.item.getdata).

Параметры

Параметр	Описание
ORDER	Массив для сортировки результата. Массив вида *{"поле_сортировки": 'направление сортировки' [, ...]}*. Поле для сортировки может принимать значения: ▪ **TITLE** - название задачи; ▪ **DATE_START** - дата старта; ▪ **DEADLINE** - крайний срок; ▪ **STATUS** - статус; ▪ **PRIORITY** - приоритет; ▪ **MARK** - оценка; ▪ **CREATED_BY** - постановщик; ▪ **RESPONSIBLE_ID** - ответственный; ▪ **GROUP_ID** - рабочая группа. Направление сортировки может принимать значения: ▪ **asc** - по возрастанию; ▪ **desc** - по убыванию;

	Необязательный. По умолчанию фильтруется по убыванию идентификатора задачи. **Примечание**. Допускается сортировка по пользовательским полям.

FILTER	Массив вида *{"фильтруемое_поле": "значение фильтра" [, ...]}*. Фильтруемое поле может принимать значения:
	
	▪ **ID** - идентификатор задачи;
	▪ **PARENT_ID** - идентификатор родительской задачи;
	▪ **GROUP_ID** - идентификатор рабочей группы;
	▪ **CREATED_BY** - постановщик;
	▪ **STATUS_CHANGED_BY** - пользователь, последним изменивший статус задачи;
	▪ **PRIORITY** - приоритет;
	▪ **FORUM_TOPIC_ID** - идентификатор темы форума;
	▪ **RESPONSIBLE_ID** - ответственный;
	▪ **TITLE** - название задачи (можно искать по шаблону [%_]) ;
	▪ **TAG** - тэг;
	▪ **REAL_STATUS** - статус задачи. Константы отражающие статусы задач:
	  ▪ STATE_NEW = 1;
	  ▪ STATE_PENDING = 2;
	  ▪ STATE_IN_PROGRESS = 3;
	  ▪ STATE_SUPPOSEDLY_COMPLETED = 4;
	  ▪ STATE_COMPLETED = 5;
	  ▪ STATE_DEFERRED = 6;
	  ▪ STATE_DECLINED = 7;
	
	▪ **STATUS** - статус для сортировки. Аналогичен **REAL_STATUS**, но имеет дополнительно два мета статуса:
	  ▪ **-2** - не просмотренная задача;

    - **-1** - просроченная задача.

- **MARK** - оценка;
- **XML_ID** - внешний код;
- **SITE_ID** - идентификатор сайта;
- **ADD_IN_REPORT** - задача в отчете (Y|N);
- **DATE_START** - дата начала выполнения;
- **DEADLINE** - крайний срок;
- **CREATED_DATE** - дата создания;
- **CLOSED_DATE** - дата завершения;
- **CHANGED_DATE** - дата последнего изменения;
- **ACCOMPLICE** - идентификатор соисполнителя;
- **AUDITOR** - идентификатор аудитора;
- **DEPENDS_ON** - идентификатор предыдущей задачи;
- **ONLY_ROOT_TASKS** - только задачи, которые не являются подзадачами (корневые задачи), а также подзадачи родительской задачи, к которой текущий пользователь доступа не имеет (Y|N);
- **SUBORDINATE_TASKS** - задачи текущего пользователя и его подчиненных (Y|N);
- **OVERDUED** - были просрочены (Y|N);
- **DEPARTMENT_ID** - идентификатор отдела.

Перед названием фильтруемого поля может указать тип фильтрации:

- "!" - не равно
- "<" - меньше
- "<=" - меньше либо равно
- ">" - больше
- ">=" - больше либо равно

"*значения фильтра*" - одиночное значение или массив.

	Необязательный. По умолчанию записи не фильтруются.
TASKDATA	Массив возвращаемых полей задачи.
NAV_PARAMS	Постраничная навигация. Доступны следующие опции: ▪ **iNumPage** - номер страницы.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Примеры

```
// Получим список всех задач (по умолчанию
сработает ограничение — постраничка по 50
элементов)
BX24.callMethod(
        'task.items.getlist',
        [],
        function(result)
        {
                console.info(result.data());
                console.log(result);
        }
);
```

```
// Получим список задач с идентификаторами
1,2,3,4,5,6. Причем выберем только поля ID и
TITLE.
// Режим постранички — по 2 элемента на
странице, 2-ая страница.
// Сортировка по ID — по убыванию.
```

```
BX24.callMethod(
        'task.items.getlist',
        [
                {ID : 'desc'},          //
Сортировка по ID — по убыванию.
                {ID: [1,2,3,4,5,6]},    //
Фильтр
                ['ID', 'TITLE'],        //
Выбираемые поля
                {
                        NAV_PARAMS: {
// постраничка
                                iNumPage  :
2               // страница номер 2
                        }
                }
        ],
        function(result)
        {
                console.info(result.data());
                console.log(result);
        }
);
```

Примечание: В целях ограничения нагрузки на постраничную навигацию наложено ограничение в 50 задач.

comment::task.comment.add

Добавление комментария к задаче.

Параметры функции

Параметр	Описание
TASKID	Идентификатор задачи.
COMMENTTEXT	Комментарий.

Внимание! Соблюдение порядка следования параметров в запросе обязательно. При его нарушении запрос будет выполнен с ошибками.

Пример

```
BX24.callMethod(
   'task.comment.add',
   [1, 'текст комментария'],
   function(result)
        {
      console.info(result.data());
      console.log(result);
```

```
            }
);
```

OnTaskAdd

Событие, вызываемое после создания задачи.

Поля ответа:

Поле	Описание
FIELDS_BEFORE	поля задачи до события. В случае отсутствия доступных полей задачи данное поле будет содержать значение **undefined**.
FIELDS_AFTER	поля задачи после события. В случае отсутствия доступных полей задачи данное поле будет содержать значение **undefined**.
IS_ACCESSIBLE_BEFORE	была ли доступна задача до события (на чтение). Возможные значения: ▪ **Y** (Yes) - да; ▪ **N** (No) - нет; ▪ **undefined** - не определено или проверка не производилась.
IS_ACCESSIBLE_AFTER	стала ли доступна задача после события (на чтение). Возможные значения: ▪ **Y** (Yes) - да; ▪ **N** (No) - нет;

	▪ **undefined** - не определено или проверка не производилась.

Пример значений полей:

```
$arEventFields = array(
    'FIELDS_BEFORE' => 'undefined', // поля задачи до события отсутствовали (задача еще не была создана)
    'FIELDS_AFTER' => array('ID' => $taskId), // идентификатор созданной задачи
    'IS_ACCESSIBLE_BEFORE' => 'N', // задача до события не была доступна на чтение (задача еще не была создана)
    'IS_ACCESSIBLE_AFTER' => 'undefined' // проверка доступности задачи на чтение не производилась
);
```

OnTaskUpdate

Событие, вызываемое при обновлении задачи.

Поля ответа:

Поле	Описание
FIELDS_BEFORE	поля задачи до события. В случае отсутствия доступных полей задачи данное поле будет содержать значение **undefined**.
FIELDS_AFTER	поля задачи после события. В случае отсутствия доступных полей задачи данное поле будет содержать значение **undefined**.
IS_ACCESSIBLE_BEFORE	была ли доступна задача до события (на чтение). Возможные значения: ▪ **Y** (Yes) - да; ▪ **N** (No) - нет; ▪ **undefined** - не определено или проверка не производилась.
IS_ACCESSIBLE_AFTER	стала ли доступна задача после события (на чтение). Возможные значения: ▪ **Y** (Yes) - да; ▪ **N** (No) - нет;

	▪ **undefined** - не определено или проверка не производилась.

Пример значений полей:

```
$arEventFields = array(
    'FIELDS_BEFORE' => array('ID' =>
$taskId), // идентификатор изменяемой задачи
    'FIELDS_AFTER' => array('ID' =>
$taskId), // идентификатор изменяемой задачи
    'IS_ACCESSIBLE_BEFORE' => 'undefined',
// проверка доступности задачи на чтение не
производилась
    'IS_ACCESSIBLE_AFTER' => 'undefined' //
проверка доступности задачи на чтение не
производилась
);
```

OnTaskDelete

Событие, вызываемое при удалении задачи.

Поля ответа:

Поле	Описание
FIELDS_BEFORE	Поля задачи до события. В случае отсутствия доступных полей задачи данное поле будет содержать значение **undefined**.
FIELDS_AFTER	Поля задачи после события. В случае отсутствия доступных полей задачи данное поле будет содержать значение **undefined**.
IS_ACCESSIBLE_BEFORE	Была ли доступна задача до события (на чтение). Возможные значения: ▪ **Y** (Yes) - да; ▪ **N** (No) - нет; ▪ **undefined** - не определено или проверка не производилась.
IS_ACCESSIBLE_AFTER	Стала ли доступна задача после события (на чтение). Возможные значения: ▪ **Y** (Yes) - да; ▪ **N** (No) - нет;

	▪ **undefined** - не определено или проверка не производилась.

Пример значений полей:

```
$arEventFields = array(
    'FIELDS_BEFORE' => array('ID' =>
$taskId), // идентификатор удаляемой задачи
    'FIELDS_AFTER' => 'undefined', // поля
задачи после события отсутствуют (задача
была удалена)
    'IS_ACCESSIBLE_BEFORE' => 'undefined',
// проверка доступности задачи на чтение не
производилась
    'IS_ACCESSIBLE_AFTER' => 'N' // задача
после события не была доступна на чтение
(задача была удалена)
);
```

OnTaskCommentAdd

Событие срабатывает при добавлении комментария к задаче.

Поля ответа:

Поле	Описание
FIELDS_BEFORE	поля задачи до события. В случае отсутствия доступных полей задачи данное поле будет содержать значение **undefined**.
FIELDS_AFTER	поля задачи после события. В случае отсутствия доступных полей задачи данное поле будет содержать значение **undefined**.
IS_ACCESSIBLE_BEFORE	была ли доступна задача до события (на чтение). Возможные значения: ▪ **Y** (Yes) - да; ▪ **N** (No) - нет; ▪ **undefined** - не определено или проверка не производилась.
IS_ACCESSIBLE_AFTER	стала ли доступна задача после события (на чтение). Возможные значения: ▪ **Y** (Yes) - да;

	▪ **N** (No) - нет; ▪ **undefined** - не определено или проверка не производилась.

Пример значений полей:

```
$arEventFields = array(
    'FIELDS_BEFORE' => array('ID' =>
$taskId), // идентификатор изменяемого
комментария
    'FIELDS_AFTER' => array('ID' =>
$taskId), // идентификатор изменяемого
комментария
    'IS_ACCESSIBLE_BEFORE' => 'undefined',
// проверка доступности задачи на чтение не
производилась
    'IS_ACCESSIBLE_AFTER' => 'undefined' //
проверка доступности задачи на чтение не
производилась
);
```

OnTaskCommentUpdate

Событие срабатывает при проведении операций над комментарием к задаче, в частности, при обновлении или удалении.

Поля ответа:

Поле	Описание
FIELDS_BEFORE	поля задачи до события. В случае отсутствия доступных полей задачи данное поле будет содержать значение **undefined**.
FIELDS_AFTER	поля задачи после события. В случае отсутствия доступных полей задачи данное поле будет содержать значение **undefined**.
IS_ACCESSIBLE_BEFORE	была ли доступна задача до события (на чтение). Возможные значения: ▪ **Y** (Yes) - да; ▪ **N** (No) - нет; ▪ **undefined** - не определено или проверка не производилась.
IS_ACCESSIBLE_AFTER	стала ли доступна задача после события (на чтение). Возможные значения: ▪ **Y** (Yes) - да;

	▪ **N** (No) - нет; ▪ **undefined** - не определено или проверка не производилась.

Пример значений полей:

```
$arEventFields = array(
    'FIELDS_BEFORE' => array('ID' =>
$taskId), // идентификатор изменяемой задачи
    'FIELDS_AFTER' => array('ID' =>
$taskId), // идентификатор изменяемой задачи
    'IS_ACCESSIBLE_BEFORE' => 'undefined',
// проверка доступности задачи на чтение не
производилась
    'IS_ACCESSIBLE_AFTER' => 'undefined' //
проверка доступности задачи на чтение не
производилась
);
```

configuration.import.register

```
configuration.import.register(
   data
)
```

Регистрация импорта.

Параметры

Параметр	Описание	С версии
data	Массив ссылок на файлы из экспортированного архива.	

Пример

```
$rest = CRest::call(
   'configuration.import.register',
   [
       'data' => [
          'STRUCTURE' => [],
          'FILES' => [],
          'MANIFEST' => [],
       ],
```

```
    ]
);
```

Результат выполнения:

```
Array
        (
            [result] => Array
                (
                    [processId] => 42
                )

            [time] => Array
                (
                    [start] =>
1629302119.8213
                    [finish] =>
1629302119.9701
                    [duration] =>
0.14881992340088
                    [processing] =>
0.085903882980347
                    [date_start] => 2021-08-
18T17:55:19+02:00
                    [date_finish] => 2021-
08-18T17:55:19+02:00
                )

        )
```

Где:

- **processId** - идентификатор процесса, на основе которого работает импорт.

configuration.import.unregiste

```
configuration.import.unregister(
   processId
)
```

Отмена зарегистрированного импорта.

Параметры

Параметр	Описание	С версии
processId	Идентификатор процесса.	

Пример

Отменить импорт можно просто прервав его текущим шагом:

```
$rest = CRest::call(
   'configuration.import.unregister',
   [
      'processId' => 42,
   ]
);
```

Возвращается:

- либо успешный результат выполнения запроса:

```
Array
(
    [result] => Array
        (
            [success] => Y
        )

    [time] => Array
        (
            [start] => 1629302449.9443
            [finish] => 1629302450.0617
            [duration] =>
0.11744093894958
            [processing] =>
0.035245895385742
            [date_start] => 2021-08-
18T18:00:49+02:00
            [date_finish] => 2021-08-
18T18:00:50+02:00
        )

)
```

- либо ошибка:

```
Array
(
    [error] => PROCESS_NOT_FOUND
    [error_description] => Process
doesn't found.
)
```

configuration.import.get

```
configuration.import.get(
   processId
)
```

Получение информации о текущем шаге импорта. Импорт происходит на агенте, выполняемом не чаще раза в минуту, поэтому нецелесообразно узнавать статус более одного раза в минуту (для каждой регистрации свой импорт).

Параметры

Параметр	Описание	С версии
processId	Идентификатор процесса.	

Пример

```
$res =
CRest::call('configuration.import.get',
   [
      'processId' => $id
   ]
);
```

Возможный результат:

```
Array
        (
            [result] => Array
                (
                    [status] => P
                    [progress] => Array
                        (
                            [action] =>
manifest
                            [step] => 0
                            [section] =>
REST_APPLICATION
                        )

                )

            [time] => Array
                (
                    [start] =>
1629303000.9341
                    [finish] =>
1629303001.0443
                    [duration] =>
0.11020803451538
                    [processing] =>
0.054312944412231
                    [date_start] => 2021-08-
18T18:10:00+02:00
                    [date_finish] => 2021-
08-18T18:10:01+02:00
                )

        )
```

Где:

- **status**:
  - S - старт (сразу после регистрации);
  - P - выполняется. Попутно возвращается **progress** с некоторыми подробностями о шаге;
  - F - импорт завершен. Также может возвращаться **additional** с дополнительными данными завершенного импорта, если они есть для данного импорта;
  - E - ошибка. Подробности в сообщениях **errors**;
  - U - неизвестный процесс (либо неверный ID, либо информация [dw]уже удалена[/dw][di]Информация очищается через 3 дня.[/di]).

- **progress** - некоторые подробности о текущем шаге. Данные используются для отладки.
- **notice** - уведомления о некритичных ошибках.
- **errors** - ошибки есть, но импорт продолжается.
- **exception** - импорт остановлен. На следующем шаге будет статус E (если он ещё не установлен).

События

Событие	Описание и параметры	Полн
OnSaleOrderSaved	Происходит в конце сохранения заказа, когда заказ и все связанные сущности уже сохранены. **Параметры** FIELDS — Массив содержит поле ID со значением идентификатора заказа.	OnSaleOr
OnSaleBeforeOrderDelete	Вызывается перед удалением заказа. **Параметры** FIELDS — Массив содержит поле ID со значением идентификатора заказа.	OnSaleBe
On[dw]<сущность>[/dw][di] где <сущность> - это: - свойство заказа PropertyValue;	Происходит непосредственно после сохранения сущности системы заказов. **Параметры**	OnProper OnPaymen OnShipme

- оплата Payment; - отгрузка Shipment; - заказ Order. [/di]EntitySaved	FIELDS	Массив содержит поле ID со значением идентификатора созданной сущности.	OnOrderE
On[dw]<сущность>[/dw][di] где <сущность> - это: - свойство заказа PropertyValue; - оплата Payment; - отгрузка Shipment; - заказ Order. [/di]Deleted	Вызывается при непосредственном удалении сущности из базы. **Параметры** FIELDS — Массив содержит поле ID со значением идентификатора созданной сущности.		OnPrope OnPayme OnShipme OnOrderI

Ресурс варианта свойства

Вариант свойства:

```
{
  description: null,
  id: 10,
  name: 'Размер',
  orderPropsId: 39,
  sort: 100,
  value: 'M',

}
```

Поля соответствуют доступному списку полей getFields.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.propertyVariant.add

```
sale.propertyVariant.add(fields)
```

Метод добавляет вариант свойства.

Если операция успешна, возвращается ресурс варианта свойства в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.propertyVariant.add',
    {
          fields: {
            orderPropsId: 39,
```

```
            name: 'Размер',
            value: 'M',
            }
          }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertyVariant.delete

```
sale.propertyVariant.delete(id)
```

Метод удаляет вариант свойства.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер варианта свойства.

Примеры

```
BX24.callMethod(
    'sale.propertyVariant.delete',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertyVariant.get

```
sale.propertyVariant.get(id)
```

Метод для доступа к полям варианта свойства.

Если операция успешна, возвращается ресурс варианта свойства в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер варианта свойств.

Примеры

```
BX24.callMethod(
    'sale.propertyVariant.get',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertyVariant.getFields

```
sale.propertyVariant.getFields()
```

Метод возвращает поля варианта свойства.

Возвращаемые поля:

Поле	Описание
id	ID варианта.
name	Название варианта/
orderPropsId	Привязка к свойству.
sort	Сортировка.
value	Значение варианта.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'sale.propertyVariant.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertyVariant.list

```
sale.propertyVariant.list(select, filter, order, navigation)
```

Метод получает список вариантов свойств.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.propertyVariant.list',
    { select:{
                id
          } ,
          filter:{
            orderPropsId: 1
          },
          order:{
                id: asc
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertyVariant.update

```
sale.propertyVariant.update(id, fields)
```

Метод для обновления полей варианта свойства.

Если операция успешна, возвращается ресурс варианта свойства в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер варианта свойства.
fields	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.propertyVariant.update',
    {
          id: 75,
```

```
            fields: {
              name: 'Размер',
              value: 'S',
              }
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс группы свойств

Группа свойств:

```
{
  "id":"1",
  "personTypeId":"1",
  "name":"Личные данные",
  "sort":"200"
}
```

Поля соответствуют доступному списку полей getFields.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.propertygroup.add

```
sale.propertygroup.add(fields)
```

Метод добавляет группу свойств.

Если операция успешна, возвращается ресурс группы свойств в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.propertygroup.add',
    {
          fields: {
            name: 'Тест',
```

```
                }
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertygroup.delete

```
sale.propertygroup.delete(propertyGroupsId)
```

Метод удаляет группу свойств.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
propertyGroupsId	string	Номер группы свойств.

Примеры

```
BX24.callMethod(
    'sale.propertygroup.delete',
    { propertyGroupsId: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertygroup.get

```
sale.propertygroup.get(propertyGroupsId)
```

Метод для доступа к полям группы свойств.

Если операция успешна, возвращается ресурс группы свойств в теле ответа.

Параметры

Параметр	Тип	Описание
propertyGroupsId	string	Номер группы свойств.

Примеры

```
BX24.callMethod(
    'sale.propertygroup.get',
    { propertyGroupsId: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertygroup.getFields

```
sale.propertygroup.getFields()
```

Метод возвращает поля группы свойств.

Возвращаемые поля:

Поле	Описание
id	Идентификатор группы свойств.
personTypeId	Идентификатор типа плательщика.
name	Название.
sort	Сортировка.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'sale.propertygroup.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertygroup.list

```
sale.propertygroup.list(select, filter,
order, navigation)
```

Метод получает список групп свойств.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.propertygroup.list',
    { select:{
                id
          } ,
          filter:{
            personTypeId: 1
          },
          order:{
                id: ASC
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertygroup.update

```
sale.propertygroup.update(propertyGroupsId,
fields)
```

Метод для обновления полей группы свойств.

Если операция успешна, возвращается ресурс группы свойств в теле ответа.

Параметры

Параметр	Тип	Описание
propertyGroupsId	string	Номер группы свойств.
fields	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.propertygroup.update',
    {
```

```
            propertyGroupsId: 75,
            fields: {
              name: 'Тест',
              }
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс заказа

Заказ и связанные сущности:


```
{
  "order":{
    "id":"81",
    "lid":"s1",
    "dateInsert":"2018-08-
17T15:21:02+03:00",
    "dateUpdate":"2018-08-
17T18:18:21+03:00",
    "personTypeId":"1",
    "statusId":"N",
    "dateStatus":"2018-08-
17T15:20:59+03:00",
    "empStatusId":null,
    "marked":"N",
    "dateMarked":null,
    "empMarkedId":null,
    "reasonMarked":null,
    "price":"4627.3000",
    "discountValue":"0.0000",
    "taxValue":"0.00",
    "userDescription":"",
    "additionalInfo":null,
    "comments":"",
    "companyId":"0",
    "responsibleId":null,
    "statGid":null,
    "datePayBefore":null,
    "dateBill":null,
```

```
    "recurringId":null,
    "lockedBy":"1",
    "dateLock":"2018-08-17T18:16:32+03:00",
    "recountFlag":"Y",
    "affiliateId":null,
    "deliveryDocNum":null,
    "deliveryDocDate":null,
    "updated1c":"N",
    "orderTopic":null,
    "xmlId":null,
    "id1c":null,
    "version1c":null,
    "version":"53",
    "externalOrder":"N",
    "storeId":null,
    "canceled":"N",
    "empCanceledId":null,
    "dateCanceled":null,
    "reasonCanceled":null,
    "basketItems"[
        ресурс корзины
    ]
    ,
    "properties":[
      ресурс значения свойств
    ],
    "payments":[
      ресурс оплат
    ],
    "shipments":[
      ресурс отгрузок
    ]
  }
}
```


Поле	Тип	Описание

lid	string	Код сайта.
personTypeId	string	ID типа плательщика.
statusId	string	ID статуса.
basketItems[]	list	Ресурс корзины.
properties[]	list	Ресурс значения свойств.
payments[]	list	Ресурс оплат.
shipments[]	list	Ресурс отгрузки.
discounts	nested object	Примененные скидки (временно не поддерживается).
taxes	nested object	Налоги (временно не поддерживается).
прочие поля		Поля, соответствующие доступному списку полей getFields.

sale.order.add

```
sale.order.add(fields)
```

Метод добавляет заказ.

Если операция успешна, возвращается ресурс заказа в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.order.add',
    {
          fields: {
                lid: 's1',
        userId: 1,
        orderTopic: '',
```

```
        responsibleId: 1,
        userDescription: '',
        currency: 'rb',
        personTypeId: 1
          }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.order.delete

```
sale.order.delete(id)
```

Метод удаляет заказ и связанные сущности.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер заказа.

Примеры

```
BX24.callMethod(
    'sale.order.delete',
    { id: id },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
```

```
        else
            console.log(result.data());
});
```

sale.order.get

```
sale.order.get(id)
```

Метод для доступа к полям заказа и полям связанных сущностей.

Если операция успешна, возвращается ресурс заказа в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер заказа.

Примеры

```
BX24.callMethod(
    'sale.order.get',
    { id: id },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
```

```
        else
            console.log(result.data());
});
```

sale.order.getFields

Описание и пример

```
sale.order.getFields()
```

Метод возвращает поля заказа.

Параметры

Без параметров.

Пример

```
BX24.callMethod(
    'sale.order.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание	Тип	Обязательнос
id	Идентификатор заказа. Только для чтения.	int	
lid	Идентификатор сайта.	string	*
dateInsert	Дата создания заказа.	datetime	
dateUpdate	Дата обновления заказа. Только для чтения.	datetime	
personTypeId	Идентификатор типа плательщика.	int	*
personxmlId	Внешний идентификатор типа плательщика. Только для чтения.	string	
statusId	Идентификатор статуса.	char	
dateStatus	Дата изменения статуса. Только для чтения.	datetime	
marked	Промаркирован ли заказ.	char	
dateMarked	Дата маркировки заказа. Только	datetime	

	для чтения.		
empMarkedId	ID пользователя, поставившего маркировку.	int	
reasonMarked	Причина, по которой заказ был промаркирован.	string	
price	Цена.	float	
discountValue	Значение скидки.	float	
taxValue	Ставка налога на заказ.. Только для чтения.	float	
userDescription	Пользовательское описание.	string	
additionalInfo	Дополнительная информация.	string	
comments	Комментарии.	string	
companyId	ID компании, привязанной к заказу.	int	
responsibleId	ID ответственного.	int	
recurringId	Код продления подписки (если он есть).	char	
lockedBy	Кем заблокирован заказ.	char	
dateLock	Дата блокировки. Только для	datetime	

	чтения.		
recountFlag	Флаг пересчета (Y/N).	char	
affiliateId	Идентификатор аффилиата.	int	
updated1c	Обновлен ли через 1С.	char	
orderTopic	Тема заказа.	string	
xmlId	Внешний идентификатор.	string	
statusxmlId	Внешний идентификатор статуса. Только для чтения.	string	
id1c	Идентификатор в 1С.	string	
version	Версия документа. Только для чтения.	int	
version1c	Версия в 1С.	string	
externalOrder	Флаг Y/N: заказ из внешней системы или нет.	char	
canceled	Была ли отмена.	char	
dateCanceled	Дата отмена. Только для чтения.	datetime	
empCanceledId	Дата отмены.	int	

reasonCanceled	Причина отмены.	string	
userId	Идентификатор клиента.	int	*
currency	Валюта.	string	*
accountnumber	Номер аккаунта. Только для чтения.	string	
payed	Заказ оплачен. Только для чтения.	Int	
deducted	Применена скидка. Только для чтения.	char	

sale.order.list

```
sale.order.list(select, filter, order,
navigation)
```

Метод получает список заказов.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.order.list',
    { select:{
                id,
                name
          } ,
          filter:{
                userId: 2
          },
          order:{
                id: ASC
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.order.tryadd

```
sale.order.tryadd(fields)
```

Метод добавляет заказ без сохранения заказа.

Если операция успешна, возвращается ресурс заказа в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей fields.

Дополнительные поля в ответе

В ответ добавляются два свойства для получения списка платежных систем и служб доставок из заказа.

Поле	Тип	Описание
order.deliveryIdListAction[]	list	Список служб доставок.
order.paySystemIdList[]	list	Список платежных систем.

Примеры

```
BX24.callMethod(
    'sale.order.tryadd',
    {
          fields: {
                siteId: 's1',
        userId: 1,
        orderTopic: '',
        responsibleId: 1,
        userDescription: '',
        personTypeId: 1
          }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.order.tryupdate

```
sale.order.tryupdate(id, fields)
```

Метод для обновления полей заказа без сохранения заказа.

Если операция успешна, возвращается ресурс заказа в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер заказа.
fields	object	Поля, соответствующие доступному списку полей fields .

Дополнительные поля в ответе

В ответ добавляются два свойства для получения списка платежных систем и служб доставок из заказа.

Поле	Тип	Описание
order.deliveryIdListAction[]	list	Список служб доставок.
order.paySystemIdList[]	list	Список платежных систем.

Примеры

```
BX24.callMethod(
    'sale.order.tryupdate',
    {
          id: 91,
          fields: {
                comments: 'test'
          }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.order.update

```
sale.order.update(id, fields)
```

Метод для обновления полей заказа.

Если операция успешна, возвращается ресурс заказа в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер заказа.
fields	object	Поля, соответствующие доступному списку полей fields .

Примеры

```
BX24.callMethod(
    'sale.order.update',
    {
          id: 91,
          fields: {
```

```
                    comments: 'test'
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс элемента списка заказов из внешних источников

Элемент списка заказов из внешних источников:


```
{
        "externalOrderId": "1",
        "id": "3",
        "orderId": "1",
        "params": false,
        "tradingPlatformId": "2",
        "tradingPlatformXmlId": "2",
        "xmlId":
"bx_e02f1e9aa89ad3a66dbe11daa4b7436f"
}
```


Поля соответствуют доступному списку полей getFields.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.tradeBinding.getFields

```
sale.tradeBinding.getFields()
```

Возвращает поля заказов из внешних источников.

Возвращаемые поля:

Поле	Описание
id	Идентификатор записи списка.
xmlId	Внешний идентификатор записи.
externalOrderId	Номер заказа во внешней системе (задается внешней системой).
orderId	Номер заказа в системе.
tradingPlatformId	Номер торговой платформы.
tradingPlatformXmlId	Внешний идентификатор торговой платформы.
params	Поле, в котором может храниться дополнительная информация.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'sale.tradeBinding.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.tradeBinding.list

```
sale.tradeBinding.list(select, filter,
order, navigation)
```

Метод для получения списка заказов из внешних источников.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей getFields.
filter	object	Поля, соответствующие доступному списку полей getFields.
order	object	Поля, соответствующие доступному списку полей getFields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.tradeBinding.list',
    { select:{
                tradingPlatformId
          } ,
          filter:{
            orderId: '2'
          },
          order:{
                tradingPlatformId: asc
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс значения свойства

Значение свойства:

```
{
  "id":"609",
  "name":"test ENUM",
  "value":[
    "code1",
    "code2"
  ],
  "code":"",
  "orderPropsId":"20"
}
```

Поля соответствуют доступному списку полей getFields.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.propertyvalue.delete

```
sale.propertyvalue.delete(propertyValueId)
```

Метод удаляет значения свойства.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
propertyValueId	string	Номер значения свойства.

Примеры

```
BX24.callMethod(
    'sale.propertyvalue.delete',
    { propertyValueId: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertyvalue.get

```
sale.propertyvalue.get(propertyValueId)
```

Метод для доступа к полям значений свойств заказа.

Если операция успешна, возвращается ресурс значения свойства в теле ответа.

Параметры

Параметр	Тип	Описание
propertyValueId	string	Номер значения свойства.

Примеры

```
BX24.callMethod(
    'sale.propertyvalue.get',
    { propertyValueId: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertyvalue.getFields

```
sale.propertyvalue.getFields()
```

Метод возвращает поля значения свойства.

Возвращаемые поля:

Поле	Описание
id	ID свойства.
orderId	ID заказа.
orderPropsId	ID свойства заказа
name	Название свойства.
value	Значение свойства.
code	Код в системе.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'sale.propertyvalue.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertyvalue.list

```
sale.propertyvalue.list(select, filter,
order, navigation)
```

Метод получает список значений свойств.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.propertyvalue.list',
    { select:{
                id
          } ,
          filter:{
            orderId: 81
          },
          order:{
                id: ASC
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertyvalue.modify

```
sale.propertyvalue.modify(fields.order.id,
fields.order.propertyValues[])
```

Метод для изменения значения свойства.

Если операция успешна, возвращается ресурс значения свойства в теле ответа.

Параметры

Параметр	Тип	Описание
fields.order.id	int	Номер заказа.
fields.order.propertyValues[]	list	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.propertyvalue.modify',
```

```
    {
                fields: {
                        order: {
                                id: 33,

propertyValues: [
                                {

orderPropsId: 2,

value: 'bx@bx.bx2'
                                },
                                {

orderPropsId: 1,

value: 'Test'
                                },
                                {

orderPropsId: 3,

value: '77777777777'
                                },
                                {

orderPropsId: 4,

value: '101000'
                                },
                                {

orderPropsId: 6,

value: '0000073738'
                                },
                                {
```

```
orderPropsId: 7,

value: 'test'
                                    },
                                    {

orderPropsId: 20,

value: [

'code1',

'code2'
                                                ]
                                    },
                                    {

orderPropsId: 21,

value: 'code1'
                                    }
                            ],
                    },
            }
        }
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Кассы::sale.cashbox.handler.a

Описание и пример

Метод добавляет REST-обработчик кассы.

Пример

```
BX.rest.callMethod(
    "sale.cashbox.handler.add",
    {
        "CODE": "restcashbox01",
        "NAME": "REST-касса 01",
        "SORT": 100,
        "SETTINGS":
        {
            "PRINT_URL":
"http:\/\/example.com\/rest_print.php",
            "CHECK_URL":
"http:\/\/example.com\/rest_check.php",
            "HTTP_VERSION": "1.1",
            "CONFIG":
            {
                "AUTH": {
                    "LABEL": "Авторизация",
                    "ITEMS": {
                        "KEYWORD": {
                            "TYPE":
```

```
"STRING",
                            "LABEL":
"Кодовое слово"
                        },
                        "PREFERENCE": {
                            "TYPE": "ENUM",
                            "LABEL":
"Множественный выбор",
                            "REQUIRED": "Y",
                            "OPTIONS": {
                                "FIRST":
"Первый",
                                "SECOND":
"Второй",
                                "THIRD":
"Третий",
                            }
                        }
                    }
                },
                "INTERACTION": {
                    "LABEL": "Настройки
взаимодействия с кассой",
                    "ITEMS": {
                        "MODE": {
                            "TYPE": "ENUM",
                            "LABEL": "Режим
работы с кассой",
                            "OPTIONS": {
                                "ACTIVE":
"боевой",
                                "TEST":
"тестовый"
                            }
                        }
                    }
                }
```

```
            }
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
```

Параметры

Параметр	Описание	С версии
CODE	Обязательный. Код REST-обработчика. Должен быть уникальным среди всех обработчиков.	
NAME	Обязательный. Название REST-обработчика.	
SORT	Сортировка (по умолчанию `100`).	
SUPPORTS_FFD105	Поддерживает ли касса формат фискальных данных версии 1.05 (по умолчанию `N`).	
SETTINGS	Обязательный. Настройки обработчика (все параметры обязательные): ▪ **PRINT_URL** - адрес, на который отправляются	

данные для печати чека;

- **CHECK_URL** - адрес, по которому происходит проверка статуса чека;
- **HTTP_VERSION** - версия протокола HTTP, используемая для запросов. Возможные значения: `1.0`, `1.1`. Если параметр не заполнен, то для запросов используется HTTP 1.0. Параметр доступен с версии **sale 22.0.100**;
- **CONFIG** - структура настроек, которые пользователь сможет устанавливать и изменять на странице редактирования кассы. Каждый ключ в этом параметре задаёт один раздел на странице настроек:
  - **LABEL** - заголовок раздела;
  - **ITEMS** - список настроек раздела:
    - **TYPE** - тип настройки;
    - **LABEL** - название настройки;
    - **REQUIRED** - является ли настройка обязательной.

Кассы::sale.cashbox.handler.d

```
sale.cashbox.handler.delete(id)
```

Метод удаляет REST-обработчик кассы.

Если к обработчику привязаны кассы, то обработчик не будет удалён (сначала необходимо удалить кассы).

Параметры

Параметр	Описание	С версии
ID	Обязательный. ID обработчика.	

Примеры

```
BX.rest.callMethod(
    "sale.cashbox.handler.delete",
    {
        "ID": 1,
    },
```

```
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
```

Кассы::sale.cashbox.handler.u

```
sale.cashbox.handler.update(id, fields)
```

Метод обновляет данные REST-обработчика кассы.

Параметры

Параметр	Описание	С версии
ID	Обязательный. ID обновляемого обработчика.	
FIELDS	Обязательный. Значения обновляемых полей.	

Примеры

```
BX.rest.callMethod(
    "sale.cashbox.handler.update",
    {
        "ID": 1,
```

```
        "FIELDS":
        {
            "NAME": "Новое имя"
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Кассы::sale.cashbox.handler.li

```
sale.cashbox.handler.list()
```

Метод для получения списка доступных REST-обработчиков касс.

Параметры

Без параметров

Пример

```
BX.rest.callMethod(
    "sale.cashbox.handler.list",
    {

    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
```

```
);
```

Кассы::sale.cashbox.add

Описание и пример

Метод добавляет кассу.

Пример

```
BX.rest.callMethod(
    "sale.cashbox.add",
    {
        "NAME": 'Rest-касса',
        "REST_CODE": 'restcashbox01',
        "EMAIL": "user@example.com",
        "NUMBER_KKM": "123",
        "ACTIVE": "Y",
        "OFD": "bx_ofdruofd",
        "OFD_SETTINGS":
        {
            "OFD_MODE":
            {
                "IS_TEST": "N"
            }
        },
        "SETTINGS":
        {
            "AUTH":
            {
                "KEYWORD": "top_secret!",
```

```
                    "PREFERENCE": "SECOND"
                },
                "INTERACTION":
                {
                    "MODE": "ACTIVE"
                }
            }
        },
        function(result)
        {
            if(result.error())
                console.error(result.error());
            else
                console.dir(result.data());
        }
);
```

Параметры

Параметр	Описание	С версии
NAME	Обязательный. Название кассы.	
REST_CODE	Обязательный. Код REST-обработчика кассы.	
EMAIL	Обязательный. Указывается email, на который будут отправляться уведомления в случае возникновения ошибок при печати чеков.	
OFD	Код обработчика ОФД (по умолчанию - без ОФД). Доступные обработчики ОФД: ▪ **bx_firstofd**	

	▪ **bx_platformaofd** ▪ **bx_yarusofd** ▪ **bx_taxcomofd** ▪ **bx_ofdruofd** ▪ **bx_tenzorofd** ▪ **bx_conturofd**	
OFD_SETTINGS	Настройки ОФД (по умолчанию - пустой массив).	
NUMBER_KKM	Внешний идентификатор кассы (по умолчанию пуст).	
KKM_ID	Марка ККМ (по умолчанию пуста).	
OFD_SETTINGS	Настройки ОФД (по умолчанию - пустой массив).	
ACTIVE	Активность кассы (по умолчанию `N`).	
SORT	Сортировка (по умолчанию `100`).	
USE_OFFLINE	Используется ли касса офлайн (по умолчанию `N`).	
SETTINGS	Настройки кассы в соответствии со структурой настроек (по умолчанию пусты).	

Кассы::sale.cashbox.update

```
sale.cashbox.update(id, fields)
```

Метод обновляет существующую кассу.

Параметры

Параметр	Описание	С версии
ID	Обязательный. ID обновляемой кассы.	
FIELDS	Обязательный. Значения обновляемых полей.	

Примеры

```
BX.rest.callMethod(
    "sale.cashbox.update",
    {
        "ID": 1,
        "FIELDS": {
```

```
            "NAME": "Новое имя",
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
```

Кассы::sale.cashbox.delete

```
sale.cashbox.delete(id)
```

Метод удаляет кассу.

Параметры

Параметр	Описание	С версии
ID	Обязательный. ID удаляемой кассы.	

Примеры

```
BX.rest.callMethod(
    "sale.cashbox.delete",
    {
        "ID": 1,
    },
    function(result)
    {
        if(result.error())
```

```
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Кассы::sale.cashbox.list

Описание и пример

Метод возвращает список настроенных касс.

Пример

```
// выбор всех полей всех касс
BX.rest.callMethod(
    "sale.cashbox.list",
    {

    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);

// выбор определённых полей
BX.rest.callMethod(
    "sale.cashbox.list",
    {
        "SELECT": ["ID", "NAME"]
    },
```

```
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);

// фильтрация элементов
BX.rest.callMethod(
    "sale.cashbox.list",
    {
        "SELECT": ["ID", "NAME"],
        "FILTER": {">ID": 9}
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);

BX.rest.callMethod(
    "sale.cashbox.list",
    {
        "SELECT": ["ID", "NAME"],
        "FILTER": {"=NAME": "Моя Rest-
касса"}
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
```

```
            console.dir(result.data());
    }
);

// упорядочивание элементов
BX.rest.callMethod(
    "sale.cashbox.list",
    {
        "SELECT": ["ID", "NAME"],
        "ORDER": {"ID": "DESC"},
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
    }
);
```

Параметры

Параметр	Описание	С версии
SELECT	Список выбираемых полей. Доступные для выбора поля: ▪ **ID** - ID кассы; ▪ **NAME** - название кассы; ▪ **HANDLER** - обработчик; ▪ **OFD** - код обработчика ОФД; ▪ **EMAIL** - email, на который будут отправляться уведомления в случае возникновения ошибок при печати чеков;	

	▪ **DATE_CREATE** - дата создания; ▪ **DATE_LAST_CHECK** - дата последней проверки; ▪ **NUMBER_KKM** - внешний идентификатор кассы; ▪ **KKM_ID** - марка ККМ; ▪ **ACTIVE** - активность кассы; ▪ **SORT** - сортировка; ▪ **USE_OFFLINE** - используется ли касса офлайн; ▪ **ENABLED** - доступность.	
FILTER	Список фильтров. Формат такой же, как у фильтров ORM (к примеру, в [ds]getList[/ds][di] Чтобы новое API выглядело для разработчика менее пугающим и более знакомым, сохранено имя самого популярного метода: *getList*. Но если раньше каждый getList имел свой набор параметров и зашитое непрозрачное поведение, то теперь этот метод един для всех сущностей и подчиняется одним законам. Даже при желании у разработчика сущности сделать "костыль" в getList ничего не выйдет. Подробнее...[/di]).	
ORDER	Параметры сортировки. Формат: `{поле: направление (ASC, DESC)}`.	

© 1С-Бухгалтер, 2001-2008, 1С 1С-Бухгалтер

Кассы::sale.cashbox.check.app

Описание и пример

Метод сохраняет результат печати чека, напечатанного на REST-кассе.

Пример

```
BX.rest.callMethod(
    "sale.cashbox.check.apply",
    {
        'UUID':'check|example.com|1',
        'PRINT_END_TIME':'1609459200',
        'REG_NUMBER_KKT':'1234567891011121',
        'FISCAL_DOC_ATTR':'1234567890',
        'FISCAL_DOC_NUMBER':'12345',
        'FISCAL_RECEIPT_NUMBER':'123',
        'FN_NUMBER':'1234567891011121',
        'SHIFT_NUMBER':'1'
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.dir(result.data());
```

```
        }
);
```

Параметры

Параметр	Описание	С версии
UUID	Обязательный. UUID чека.	
PRINT_END_TIME	Время окончания печати чека.	
REG_NUMBER_KKT	Регистрационный номер ККТ.	
FISCAL_DOC_ATTR	Фискальный признак документа.	
FISCAL_DOC_NUMBER	Номер фискального документа.	
FISCAL_RECEIPT_NUMBER	Фискальный номер чека.	
FN_NUMBER	Номер фискального накопителя.	
SHIFT_NUMBER	Номер смены.	

Кассы::Пример реализации простой кассы на REST API

Добавление обработчика кассы

Пример добавления обработчика кассы с возможностью настройки данных для авторизации, информации о компании и режиме работы кассы:

```
        BX.rest.callMethod(
    "sale.cashbox.handler.add",
    {
        "CODE": "my_rest_cashbox",
        "NAME": "Моя REST-касса",
        "SORT": 100,
        "SETTINGS":
        {
            "PRINT_URL":
"http:\/\/example.com\/rest_print.php",
            "CHECK_URL":
"http:\/\/example.com\/rest_check.php",
            "CONFIG":
            {
                "AUTH": {
                    "LABEL": "Авторизация",
                    "ITEMS": {
                        "LOGIN": {
                            "TYPE":
"STRING",
                            "REQUIRED": "Y",
```

```
                            "LABEL": "Логин"
                        },
                        "PASSWORD": {
                            "TYPE":
"STRING",
                            "REQUIRED": "Y",
                            "LABEL":
"Пароль"
                        },
                    }
                },
                "COMPANY": {
                    "LABEL": "Данные об
организации",
                    "ITEMS": {
                        "INN": {
                            "TYPE":
"STRING",
                            "REQUIRED": "Y",
                            "LABEL": "ИНН
организации"
                        }
                    }
                },
                "INTERACTION": {
                    "LABEL": "Настройки
взаимодействия с кассой",
                    "ITEMS": {
                        "MODE": {
                            "TYPE": "ENUM",
                            "LABEL": "Режим
работы с кассой",
                            "OPTIONS": {
                                "ACTIVE":
"боевой",
                                "TEST":
"тестовый"
```

```
                                    }
                                }
                            }
                        }
                    }
                }
            },
            function(result)
            {
                if(result.error())
                    console.error(result.error());
                else
                    console.dir(result.data());
            }
);
```

Добавленная касса настраивается пользователем через настройки магазина или через Центр продаж. Пример настройки:

- Вкладка **Касса**:
- Вкладка **Настройки**:

Страница PRINT_URL

Страница **PRINT_URL** - адрес, на который отправляются данные для печати чека.

Структура POST-параметров, отправляемых на адрес **PRINT_URL**:

Параметр	Описание
type	Тип чека. Значения: ▪ **sell** - полная оплата;

	▪ **sellreturncash** - полный возврат наличными; ▪ **sellreturn** - полный возврат безналичными; ▪ **advancepayment** - аванс; ▪ **advancereturn** - возврат аванса безналичными; ▪ **advancereturncash** - возврат аванса наличными; ▪ **creditpayment** - оплата кредита; ▪ **creditpaymentreturn** - возврат оплаты кредита безналичными; ▪ **creditpaymentreturncash** - возврат оплаты кредита наличными; ▪ **credit** - покупка в кредит; ▪ **creditreturn** - возврат покупки в кредит; ▪ **prepayment** - частичная предоплата; ▪ **prepaymentreturn** - возврат частичной предоплаты безналичными; ▪ **prepaymentreturncash** - возврат частичной предоплаты наличными; ▪ **fullprepayment** - 100% предоплата; ▪ **fullprepaymentreturn** - возврат 100% предоплаты безналичными; ▪ **fullprepaymentreturncash** - возврат 100% предоплаты наличными.
calculated_sign	Признак прихода/расхода. Значения: ▪ **income** - приход; ▪ **consumption** - расход.
unique_id	ID чека в базе данных портала.
items	Массив товаров в чеке. Структура элементов массива: ▪ **name** - название товара; ▪ **base_price** - базовая цена товара; ▪ **price** - цена товара;

- **sum** - сумма позиции;
- **quantity** - количество товара;
- **vat** - id налога;
- **vat_sum** - сумма налога;
- **payment_object** - признак предмета расчёта. Значения:
  - **commodity** - товар;
  - **excise** - подакцизный товар;
  - **job** - работа;
  - **service** - услуга;
  - **payment** - платёж;
  - **gambling_bet** - ставка азартной игры;
  - **gambling_prize** - выигрыш азартной игры;
  - **lottery** - лотерейный билет;
  - **lottery_prize** - выигрыш лотереи;
  - **intellectual_activity** - предоставление результатов интеллектуальной деятельности;
  - **agent_commission** - агентское вознаграждение;
  - **composite** - составной предмет расчета;
  - **another** - иной предмет расчета;
  - **property_right** - имущественное право;
  - **non-operating_gain** - внереализационный доход;
  - **sales_tax** - торговый сбор;
  - **resort_fee** - курортный сбор.
- **payment_method** - признак способа расчёта. Значения:
  - **full_payment** - полный расчёт;
  - **advance** - аванс;
  - **prepayment** - предоплата;
  - **full_prepayment** - 100% предоплата;
  - **credit** - покупка в кредит;

	▪ **credit_payment** - оплата кредита.
date_create	Дата создания чека (timestamp).
payments	Массив оплат. Структура элементов массива: ▪ **type** - тип оплаты. Значения: ▪ **cash** - оплата наличными; ▪ **cashless** - безналичный расчёт. ▪ **is_cash** - производится ли оплата наличными. Значения: Y/N ▪ **sum** - сумма оплаты.
client_email	E-mail клиента.
total_sum	Общая сумма по чеку.
uuid	Идентификатор документа во внешней системе (портал *Битрикс24*).
number_kkm	Внешний идентификатор кассы (из настроек кассы).
service_email	Email (из настроек кассы).
cashbox_params	Массив настроек кассы. Структура задаётся параметром **CONFIG** массива **SETTINGS**, указанного в качестве параметра метода sale.cashbox.handler.add.

По адресу **PRINT_URL** происходит обработка входных данных, формирование документа и возвращение результата печати. Ответом в случае ошибки служит JSON-массив вида:

```
{
"ERRORS": [
      "Сообщение об ошибке",
```

```
        "Сообщение об ошибке",
        ...
]
}
```

При успешной печати массив имеет вид:

```
{
"UUID": "00112233-4455-6677-8899-
aabbccddeeff"
}
```

Пример реализации простого обработчика:

```
<?php

$login = $_REQUEST['cashbox_params']['AUTH']
['LOGIN'];
$password = $_REQUEST['cashbox_params']
['AUTH']['PASSWORD'];

if (empty($login) || empty($password))
{
  echo json_encode([
     'ERRORS' => [
        'Authorization data is missing',
     ]
  ]);
  die();
}

// произведение необходимых действий:
авторизация пользователя, обработка данных,
регистрация чека в системе, ...
```

```
echo json_encode([
  'UUID' => $resultUUID,
]);
```

Страница CHECK_URL

Страница **CHECK_URL** - адрес, по которому происходит проверка успешности печати чека.

Запрос по адресу **CHECK_URL** производится по запросу менеджера либо спустя некоторое время после успешной печати чека.

Структура POST-параметров, отправляемых на адрес **PRINT_URL**:

Параметр	Описание
uuid	Идентификатор чека.

Запрос по адресу **CHECK_URL** должен возвращать данные о чеке, данные об ошибке, возникшей при печати, либо статус “в ожидании печати”.

Пример реализации простого обработчика:

```
<?php

$uuid = $_REQUEST['uuid'];

// произведение необходимых действий:
проверка статуса чека, ...

// при печати чека произошла ошибка
if ($result['ERROR'])
{
  echo json_encode([
     'STATUS' => 'ERROR',
```

```
      'ERROR' => 'Сообщение об ошибке'
  ]);
  die();
}

// чек ещё не напечатан
if ($result['WAIT'])
{
  echo json_encode([
      'STATUS' => 'WAIT',
  ]);
  die();
}

// отправка результата

echo json_encode([
  'STATUS' => 'DONE',
  'UUID' => $uuid,
  'REG_NUMBER_KKT' => '000111222333',
  'FISCAL_DOC_ATTR' => '33445500',
  'FISCAL_DOC_NUMBER' => 123,
  'FISCAL_RECEIPT_NUMBER' => 10,
  'FN_NUMBER' => '0011223344556677',
  'SHIFT_NUMBER' => 12,
  'PRINT_END_TIME' => 1609452000,
]);
```

Данные, возвращаемые **CHECK_URL**, сохраняются в базу данных и используются для генерации ссылки на чек.

Ресурс корзины

Элемент коллекции корзины и связанные сущности:


```
{
  "module":"catalog",
  "productId":"62",
  "id":"171",
  "lid":"s1",
  "quantity":"2.0000",
  "weight":"0.00",
  "price":"1614.1500",
  "customPrice":"N",
  "basePrice":"1899.0000",
  "productPriceId":"21",
  "priceTypeId":"1",
  "currency":"RUB",
  "barcodeMulti":"Y",
  "name":"Штаны Жизнь в Абстракции",
  "catalogXmlId":"clothes_offers",
  "vatRate":"0.0000",
  "notes":"Розничная цена",
  "discountPrice":"284.8500",

"productProviderClass":"\\Bitrix\\Catalog\\P
roduct\\CatalogProvider",
  "dimensions":"a:3:
{s:5:\u0022WIDTH\u0022;N;s:6:\u0022HEIGHT\u0
022;N;s:6:\u0022LENGTH\u0022;N;}",
  "type":null,
  "setParentId":null,
  "detailPageUrl":"\/catalog\/pants\/pants-
```

```
life-in-abstraction\/",
  "measureCode":"796",
  "measureName":"Штука",
  "orderId":"81",
  "productXmlId":"207#209",
  "subscribe":"N",
  "recommendation":null,
  "vatIncluded":"Y",
  "sort":"100",
  "discountName":null,
  "discountValue":null,
  "discountCoupon":null,
  "properties":[
    {
      "id":"680",
      "basketId":"171",
      "name":"Артикул",
      "value":"177-78-02",
      "code":"ARTNUMBER",
      "sort":"100"
    },
    {
      "id":"681",
      "basketId":"171",
      "name":"Цвет",
      "value":"Красный с Синим",
      "code":"COLOR_REF",
      "sort":"100"
    },
    {
      "id":"682",
      "basketId":"171",
      "name":"Размеры одежды",
      "value":"L",
      "code":"SIZES_CLOTHES",
      "sort":"100"
    },
```

```
      {
        "id":"683",
        "basketId":"171",
        "name":"Catalog XML_ID",
        "value":"clothes_offers",
        "code":"CATALOG.XML_ID",
        "sort":"100"
      },
      {
        "id":"684",
        "basketId":"171",
        "name":"Product XML_ID",
        "value":"207#209",
        "code":"PRODUCT.XML_ID",
        "sort":"100"
      }
    ]
  }
```


Поле	Тип	Описание
basketItems[]	list	Ресурс корзины.
properties[]	list	Свойства корзины.
прочие поля		Поля, соответствующие доступному списку полей getFields.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.basketitem.add

```
sale.basketitem.add(fields)
```

Метод добавляет в коллекцию элемент корзины.

Если операция успешна, возвращается ресурс корзины в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.basketitem.add',
        {
        fields: {
                            orderId: 85
```

```
                                productId: 227,
                                quantity: 2,
                                name: 'Платье'
            }
        },
        function(result) {
            if (result.error())

console.error(result.error().ex);
            else
                console.log(result.data());
        });
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.basketitem.getFields

Описание и пример

```
sale.basketitem.getFields()
```

Метод возвращает поля элемента корзины.

Параметры

Без параметров.

Пример

```
BX24.callMethod(
    'sale.basketitem.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
```

```
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание
id	Идентификатор товара.
lid	Сайт, на котором сделана покупка.
orderId	Идентификатор заказа.
productId	Код товара.
productPriceId	Идентификатор цены товара.
priceTypeId	Идентификатор типа цены товара.
name	Название товара.
price	Цена товара с учетом скидок и наценок.
currency	Валюта.
basePrice	Цена товара без учета скидок и наценок.
vatIncluded	Флаг(Y/N): включен налог или нет.
weight	Вес.
quantity	Количество.
module	Модуль.
productProviderClass	Имя класса провайдера.

notes	Особые заметки.
detailPageUrl	URL страницы детального просмотра.
discountPrice	Скидка.
catalogXmlId	Внешний идентификатор каталога.
productXmlId	Внешний идентификатор товара.
discountName	Название скидки.
discountValue	Значение скидки.
discountCoupon	Купон на скидку.
vatRate	Ставка НДС.
subscribe	Флаг (Y/N): подписка на товар.
barcodeMulti	Флаг (Y/N): штрихкод уникальный или нет.
customPrice	Флаг (Y/N): кастомная цена или нет.
dimensions	Размеры.
type	Тип товара.
setParentId	Идентификатор родительского раздела.
measureCode	Код единицы измерения.
measureName	Название единицы измерения.
recommendation	Рекомендация.
sort	Сортировка.

© «Битрикс», 2001-2008, «1С-

sale.basketitem.addCatalogPro

```
sale.basketitem.addCatalogProduct(id, fields)
```

Метод добавляет в коллекцию элемент корзины.

Если операция успешна, возвращается ресурс корзины в теле ответа.

В отличие от sale.basketitem.add, метод **sale.basketitem.addCatalogProduct** принимает минимальный набор полей для добавления записи в корзину. А при его вызове поля товара заполняются значениями из системного провайдера автоматически, тем самым упрощая работу с корзиной.

Параметры

Параметр	Тип	Описание
id	string	Номер элемента коллекции корзины.
fields	object	Поля, соответствующие доступному списку полей sale.basketItem.getFieldsCatalogProduct.

Примеры

```
BX24.callMethod(
    'sale.basketitem.addCatalogProduct',
    {
        fields: {
              productId: 53,
              orderId: 53,
              quantity: 2,
              currency: 'RUB'
        }
    },
    function(result) {
        if (result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
    });
```

sale.basketItem.getFieldsCatal

Описание

```
sale.basketItem.getCatalogProductFields()
```

Метод возвращает поля элемента корзины.

Возвращаемые поля

Поле	Описание
id	Идентификатор товара.
lid	Сайт, на котором сделана покупка.
orderId	Идентификатор заказа.
productId	Код товара.
productPriceId	Идентификатор цены товара.
priceTypeId	Идентификатор типа цены товара.
name	Название товара.
price	Цена товара с учетом скидок и наценок.

currency	Валюта.
basePrice	Цена товара без учета скидок и наценок.
vatIncluded	Флаг(Y/N): включен налог или нет.
weight	Вес.
quantity	Количество.
module	Модуль.
productProviderClass	Имя класса провайдера.
notes	Особые заметки.
detailPageUrl	URL страницы детального просмотра.
discountPrice	Скидка.
catalogXmlId	Внешний идентификатор каталога.
productXmlId	Внешний идентификатор товара.
discountName	Название скидки.
discountValue	Значение скидки.
discountCoupon	Купон на скидку.
vatRate	Ставка НДС.
subscribe	Флаг (Y/N): подписка на товар.
barcodeMulti	Флаг (Y/N): штрихкод уникальный или нет.
customPrice	Флаг (Y/N): кастомная цена или нет.
dimensions	Размеры.

type	Тип товара.
setParentId	Идентификатор родительского раздела.
measureCode	Код единицы измерения.
measureName	Название единицы измерения.
recommendation	Рекомендация.
sort	Сортировка.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(

'sale.basketItem.getCatalogProductFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.basketitem.delete

```
sale.basketitem.delete(basketId)
```

Метод удаляет элемент коллекции корзины.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
basketId	string	Номер элемента коллекции корзины.

Примеры

```
BX24.callMethod(
    'sale.basketitem.delete',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.basketitem.get

```
sale.basketitem.get(basketId)
```

Метод для доступа к полям элемент коллекции корзины и полям связанных сущностей.

Если операция успешна, возвращается ресурс корзины в теле ответа.

Параметры

Параметр	Тип	Описание
basketId	string	Номер элемента коллекции корзины.

Примеры

```
BX24.callMethod(
    'sale.basketitem.get',
    { id: id },
    function(result)
    {
```

```
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.basketItem.list

```
sale.basketItem.list(select, filter, order,
navigation)
```

Метод получает список элементов коллекции корзины.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.basketItem.list',
    { select:{
                id,
                name
          } ,
          filter:{
                quantity: 2
          },
          order:{
                id: ASC
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.basketitem.update

```
sale.basketitem.update(id, fields)
```

Метод для обновления полей элемента коллекции корзины.

Если операция успешна, возвращается ресурс корзины в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер элемента коллекции корзины.
fields	object	Поля, соответствующие доступному списку полей getFields .

Примеры

```
BX24.callMethod(
    'sale.basketitem.update',
    {
          id: 101,
```

```
            fields: {
                  quantity: 1,
                  productId: 61
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс локализации статусов

Локализация статусов:

```
{
        description: null,
        lid: 'ru',
        name: 'D',
        statusId: 'XX'
}
```

Поля соответствуют доступному списку полей getFields.

© «Битрикс», 2001-2008, «1С-Битрикс»: 2008-2022

sale.statusLang.add

```
sale.statusLang.add(fields)
```

Метод для добавления локализации.

Если операция успешна, возвращается ресурс локализации статусов в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей getFields.

Примеры

```
BX24.callMethod(
    'sale.statusLang.add',
    {
          fields: {
                statusId: 'XX',
```

```
                    lid: 'ru',
                    name: 'Финальный'
                }
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.statusLang.deleteByFilter

```
sale.statusLang.deleteByFilter(fields)
```

Метод для удаления локализации.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей getFields.

Примеры

```
BX24.callMethod(
    'sale.statusLang.deleteByFilter',
     {
           fields: {
                statusId: 'XX',
                lid: 'ru'
```

```
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.statusLang.getFields

```
sale.statusLang.getFields()
```

Метод возвращает поля локализации статусов.

Возвращаемые поля:

Поле	Описание
description	Описание.
lid	Привязка к языку.
name	Название.
statusId	Код статуса.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'sale.statusLang.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.statusLang.getListLangs

```
sale.statusLang.getListLangs()
```

Метод для получения списка языков для локализации.

Если операция успешна, возвращается список языков в теле ответа.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'sale.statusLang.getListLangs',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
```

```
                console.log(result.data());
});
```

sale.statusLang.list

```
sale.statusLang.list(select, filter, order,
navigation)
```

Метод для получения списка локализаций статусов.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.statusLang.list',
    { select:{
                statusId
          } ,
          filter:{
            lid: 'en'
          },
          order:{
                name: asc
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс оплат

Элемент коллекции оплат:


```
{
  "id":"150",
  "orderId":"81",
  "paid":"N",
  "datePaid":null,
  "empPaidId":null,
  "paySystemId":"3",
  "psStatus":null,
  "psStatusCode":null,
  "psInvoiceId":null,
  "psStatusDescription":null,
  "psStatusMessage":null,
  "psSum":null,
  "psCurrency":null,
  "psResponseDate":null,
  "payVoucherNum":"",
  "payVoucherDate":null,
  "datePayBefore":null,
  "dateBill":"2018-08-17T15:21:00+03:00",
  "xmlId":null,
  "sum":"4627.3000",
  "priceCod":"0.0000",
  "currency":"RUB",
  "paySystemName":"Яндекс.Деньги",
  "responsibleId":null,
  "empResponsibleId":null,
  "dateResponsibleId":null,
  "comments":null,
```

```
    "companyId":"0",
    "payReturnNum":"",
    "payReturnDate":null,
    "empReturnId":null,
    "payReturnComment":"",
    "isReturn":"N",
    "marked":"N",
    "dateMarked":null,
    "empMarkedId":null,
    "reasonMarked":null,
    "updated1c":"N",
    "id1c":null,
    "version1c":null,
    "externalPayment":"N"
  }
```


Поля соответствуют доступному списку полей getFields.

sale.payment.add

```
sale.payment.add(fields.orderId,
fields.payment)
```

Метод добавляет элемент коллекции оплат.

Если операция успешна, возвращается ресурс оплат в теле ответа.

Параметры

Параметр	Тип	Описание
fields.orderId	object	Номер заказа.
fields.payment	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.payment.add',
    {
          fields: {
```

```
                    orderId: 33,
                    sum: 200,
                    paid: N,
                    comments: 'Комментарий',
                    paySystemId: 3,
                    responsibleId: 1,
                    payVoucherNum: '000000001'
                }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.payment.delete

```
sale.payment.delete(id)
```

Метод удаляет элемент коллекции оплат.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер элемента коллекции оплат.

Примеры

```
BX24.callMethod(
    'sale.payment.delete',
    { id: id },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
```

```
        else
            console.log(result.data());
});
```

sale.payment.get

```
sale.payment.get(id)
```

Метод для доступа к полям элемента коллекции оплат.

Если операция успешна, возвращается ресурс элемента коллекции оплат в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер элемента коллекции оплат.

Примеры

```
BX24.callMethod(
    'sale.payment.get',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.payment.getFields

Описание и пример

```
sale.payment.getFields()
```

Метод возвращает поля оплаты.

Параметры

Без параметров.

Пример

```
BX24.callMethod(
    'sale.payment.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
```

```
                console.log(result.data());
 });
```

Возвращаемые поля

Поле	Описание
id	Идентификатор оплаты.
orderId	Идентификатор заказа.
paid	Флаг оплаченности: Y или N.
datePaid	Дата оплаты.
empPaidId	Кто оплатил.
paySystemId	ID платежной системы.
psStatus	Данные, пришедшие от платежной системы: статус транзакции.
psStatusCode	Данные, пришедшие от платежной системы: код транзакции.
psInvoiceId	Идентификатор оплаты в платежной системе.
psStatusDescription	Данные, пришедшие от платежной системы: описание транзакции.
psStatusMessage	Данные, пришедшие от платежной системы: сообщение.
psSum	Данные, пришедшие от платежной системы: сумма транзакции.
psCurrency	Данные, пришедшие от платежной системы: валюта.

psResponseDate	Данные, пришедшие от платежной системы: дата ответа от платежной системы.
payVoucherNum	Номер платежного документа.
payVoucherDate	Дата платежного документа.
datePayBefore	Дата, до которой необходимо оплатить счет (в магазине не используется).
dateBill	Дата выставления счета.
xmlId	Внешний идентификатор.
sum	Сумма оплаты.
priceCod	Стоимость оплаты при доставке (используется, например, для наложенного платежа).
currency	Валюта.
paySystemName	Название платежной системы.
responsibleId	ID пользователя, ответственного за оплату.
empResponsibleId	ID пользователя, который назначил ответственного.
dateResponsibleId	Дата назначения ответственного.
comments	Комментарий к оплате.
companyId	ID компании, которая будет принимать оплату.
payReturnNum	Номер документа возврата.
payReturnDate	Дата документа возврата.

empReturnId	ID пользователя, который выполнял возврат.
payReturnComment	Комментарий к возврату.
isReturn	Выполнялся ли возврат.
marked	Промаркирована ли оплата.
dateMarked	Дата маркировки.
empMarkedId	Кем промаркирована оплата.
reasonMarked	Причина маркировки.
updated1c	Обновлена ли через 1С.
id1c	Идентификатор в 1С.
version1c	Версия документа оплаты от 1С.
externalPayment	Флаг Y/N: внешняя оплата или нет.

sale.payment.list

```
sale.payment.list(select, filter, order,
navigation)
```

Метод получает список элементов коллекции оплат.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.payment.list',
    { select:{
                id
          } ,
          filter:{
                orderId: 2
          },
          order:{
                id: ASC
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.payment.update

```
sale.payment.update(id, fields)
```

Метод для обновления полей элемента коллекции оплат.

Если операция успешна, возвращается ресурс оплат в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер элемента коллекции оплат.
fields	object	Поля, соответствующие доступному списку полей fields .

Примеры

```
BX24.callMethod(
    'sale.payment.update',
    {
          id: 101,
```

```
            fields: {
                  payd: Y
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс отгрузки

Элемент коллекции отгрузки:


```
{
  "id":"182",
  "orderId":"81",
  "statusId":"DN",
  "basePriceDelivery":"500.0000",
  "priceDelivery":"500.0000",
  "customPriceDelivery":"N",
  "currency":"RUB",
  "discountPrice":null,
  "allowDelivery":"Y",
  "dateAllowDelivery":"2018-08-
17T17:33:17+03:00",
  "empAllowDeliveryId":"1",
  "deducted":"N",
  "dateDeducted":null,
  "empDeductedId":null,
  "reasonUndoDeducted":null,
  "deliveryId":"1",
  "deliveryDocNum":"000000001",
  "deliveryDocDate":null,
  "trackingNumber":"0000000002",
  "trackingStatus":null,
  "trackingDescription":null,
  "trackingLastCheck":null,
  "xmlId":null,
  "deliveryName":"Доставка курьером",
  "canceled":"N",
  "dateCanceled":null,
```

```
  "empCanceledId":null,
  "marked":"N",
  "dateMarked":null,
  "empMarkedId":null,
  "reasonMarked":null,
  "system":"N",
  "responsibleId":"1",
  "empResponsibleId":"1",
  "dateResponsibleId":"2018-08-
17T17:33:17+03:00",
  "comments":"Комментарий",
  "companyId":"0",
  "updated1c":"N",
  "id1c":null,
  "version1c":null,
  "externalDelivery":"N",
  "shipmentItems":[
    {
      "id":"278",
      "orderDeliveryId":"182",
      "basketId":"171",
      "dateInsert":"2018-08-
17T16:21:18+03:00",
      "quantity":"1.0000",
      "reservedQuantity":"0.0000",
    },
    {
      "isChanged":false,
      "id":"287",
      "orderDeliveryId":"182",
      "basketId":"173",
      "dateInsert":"2018-08-
17T18:20:27+03:00",
      "quantity":"1.0000",
      "reservedQuantity":"0.0000",
    }
```

```
    ]
  }
```

Поле	Тип	Описание
shipmentItems[]	list	Позиции корзины для отгрузки.
прочие поля		Поля, соответствующие доступному списку полей getFields.

sale.shipment.add

```
sale.shipment.add(fields)
```

Метод добавляет элемент коллекции отгрузок.

Если операция успешна, возвращается ресурс отгрузки в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.shipment.add',
    {
          fields: {
                        orderId: 81,
                        deducted: 'N',
```

```
                            deliveryDocNum:
'000000001',
                            trackingNumber:
'0000000002',
                            comments:
'Comments',
                            statusId: 'DN',
                            responsibleId: '1',
                            deliveryId: '1',
                            customPriceDelivery:
'N',
                            priceDelivery:
'500',
                            allowDelivery: 'Y',
                            basePriceDelivery:
'500'
                }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.shipment.delete

```
sale.shipment.delete(shipmentId)
```

Метод удаляет элемент коллекции отгрузок.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
shipmentId	string	Номер элемента коллекции отгрузок.

Примеры

```
BX24.callMethod(
    'sale.shipment.delete',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.shipment.get

```
sale.shipment.get(shipmentId)
```

Метод для доступа к полям отгрузки заказа и полям связанных сущностей.

Если операция успешна, возвращается ресурс отгрузки в теле ответа.

Параметры

Параметр	Тип	Описание
shipmentId	string	Номер элемента коллекции отгрузок.

Примеры

```
BX24.callMethod(
    'sale.shipment.get',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.shipment.getFields

Описание и пример

```
sale.shipment.getFields()
```

Метод возвращает поля отгрузки.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'sale.shipment.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
```

```
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание
id	ID отгрузки.
orderId	ID заказа.
statusId	Статус отгрузки.
basePriceDelivery	Базовая стоимость доставки.
priceDelivery	Стоимость доставки.
customPriceDelivery	Флаг, установлена ли цена вручную.
currency	Валюта.
discountPrice	Сумма итоговой скидки/наценки для отгрузки.
allowDelivery	Флаг разрешения доставки: Y/N.
dateAllowDelivery	Дата разрешения доставки.
empAllowDeliveryId	ID пользователя, который разрешал доставку.
deducted	Флаг отгрузки: Y/N.
dateDeducted	Дата отгрузки.
empDeductedId	ID пользователя, который отгружал отгрузку.
reasonUndoDeducted	Причина отмены отгрузки.

deliveryId	ID службы доставки.
deliveryDocNum	Номер документа отгрузки.
deliveryDocDate	Дата документа отгрузки.
trackingNumber	Трэк-номер.
trackingStatus	Статус отправления.
trackingDescription	Описание статуса отправления.
trackingLastCheck	Дата последней проверки статуса отправления.
xmlId	Внешний идентификатор.
deliveryName	Название службы доставки.
canceled	Отменена ли отгрузка.
dateCanceled	Дата отмены отгрузки.
empCanceledId	ID пользователя, который отменил отгрузку.
marked	Промаркирована ли отгрузка.
dateMarked	Дата маркировки.
empMarkedId	ID пользователя, который промаркировал отгрузку.
reasonMarked	Причина маркировки.
system	Флаг, устанавливающий, системная это отгрузка или нет.
responsibleId	ID пользователя, ответственного за отгрузку.

empResponsibleId	ID пользователя, который назначил ответственного.
dateResponsibleId	Дата назначения ответственного.
comments	Комментарий.
companyId	ID компании, занимающейся отгрузкой.
updated1c	Обновлена ли отгрузка через 1С
id1c	ID отгрузки в 1С.
version1c	Версия документа отгрузки от 1С.
externalDelivery	Флаг Y/N: доставка из внешней системы или нет.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.shipment.list

```
sale.shipment.list(select, filter, order,
navigation)
```

Метод получает список элементов коллекции отгрузок.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.shipment.list',
    { select:{
                id
          } ,
          filter:{
                orderId: 2
          },
          order:{
                id: ASC
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.shipment.update

```
sale.shipment.update(Id, fields)
```

Метод для обновления полей элемента коллекции отгрузок.

Если операция успешна, возвращается ресурс отгрузки в теле ответа.

Параметры

Параметр	Тип	Описание
Id	string	Номер элемента коллекции отгрузок.
fields	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.shipment.update',
    {
        id: 34,
```

```
            fields: {
                  deducted: 'N',
                  deliveryDocNum: '000000001',
                  trackingNumber:
'0000000002',
                  comments: 'Comments',
                  statusId: 'DN',
                  responsibleId: '1',
                  deliveryId: '1',
                  customPriceDelivery: 'N',
                  priceDelivery: '500',
                  allowDelivery: 'Y',
                  basePriceDelivery: '500'
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
        });
```

Ресурс привязки свойства

Привязка свойств:

```
{
  entityId: 6,
  entityType: 'D',
  propertyId: 40
}
```

Поля соответствуют доступному списку полей getFields.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.propertyRelation.add

```
sale.propertyRelation.add(fields)
```

Метод добавляет привязку свойства.

Если операция успешна, возвращается ресурс привязки свойства в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей getFields.

Примеры

```
BX24.callMethod(
    'sale.propertyRelation.add',
    {
        fields: {
            entityId: 6,
```

```
                entityType: 'D',
                propertyId: 40
            }
        }
    },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertyRelation.deleteBy

```
sale.propertyRelation.deleteByFilter(fields)
```

Метод удаляет группу свойств.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей getFields.

Примеры

```
BX24.callMethod(
    'sale.propertyRelation.deleteByFilter',
     {
          fields: {
                entityId: 6,
                entityType: 'D',
```

```
                propertyId: 40
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertyRelation.getFields

```
sale.propertyRelation.getFields()
```

Метод возвращает поля привязки свойства.

Возвращаемые поля:

Поле	Описание
entityId	Идентификатор сущности.
entityType	Тип сущности.
propertyId	Идентификатор свойства.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'sale.propertyRelation.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.propertyRelation.list

```
sale.propertyRelation.list(select, filter,
order, navigation)
```

Метод получает список привязок свойства.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.propertyRelation.list',
    { select:{
                entityId
          } ,
          filter:{
            propertyId: 1
          },
          order:{
                entityType: asc
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс свойства заказа

Свойство заказа:


```
{
  id: 20,
  personTypeId: 1,
  name: 'test ENUM',
  type: 'ENUM',
  required: 'N',
  defaultValue: null,
  sort: 100,
  userProps: 'N',
  isLocation: 'N',
  propsGroupId: 2,
  description: null,
  isEmail: 'N',
  isProfileName: 'N',
  isPayer: 'N',
  isLocation4tax: 'N',
  isFiltered: 'N',
  code: null,
  isZip: 'N',
  isPhone: 'N',
  isAddress: 'N',
  active: 'Y',
  util: 'N',
  inputFieldLocation: '0',
  multiple: 'Y'
  settings: []
}
```

Поля соответствуют доступному списку полей getFieldsByType.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.property.add

```
sale.property.add(fields)
```

Метод добавляет свойство заказа.

Если операция успешна, возвращается ресурс свойства заказа в теле ответа.

Параметры

Параметр	Тип	Описание
fields	nested object	Поля, соответствующие доступному списку полей getFieldsByType.

Примеры

```
BX24.callMethod(
    'sale.property.add',
    {
        fields: {
            personTypeid: '1',
```

```
                propsGroupId: '8',
                name: 'Должность',
                code: 'Position',
                active: 'Y',
                util: 'Y',
                userProps: 'Y',
                isFiltered: 'Y',
                sort: '100',
                description: '',
                type: 'ENUM',
                required: 'N',
                settings[
                        multiline: 'Y',
                        maxlength: 100
                ]
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.property.delete

```
sale.property.delete(propertyId)
```

Метод удаляет свойство заказа.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
propertyId	string	Номер свойства заказа.

Примеры

```
BX24.callMethod(
    'sale.property.delete',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.property.get

```
sale.property.get(id)
```

Метод для доступа к полям и настройкам свойства заказа.

Если операция успешна, возвращается ресурс свойства заказа в теле ответа.

Параметры

Параметр	Тип	Описание
id	int	Номер свойства заказа.

Примеры

```
BX24.callMethod(
    'sale.property.get',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.property.getFieldsByType

Описание и пример

```
sale.property.getFieldsByType()
```

Метод возвращает поля и настройки свойства заказа для определенного типа свойства.

Параметры

Без параметров.

Пример

```
BX24.callMethod(
    'sale.property.getFieldsByType',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
```

```
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание
id	ID свойства.
personTypeId	ID типа плательщика, к которому привязано свойство.
name	Название свойства.
type	Тип.
required	Флаг обязательности.
defaultValue	Значение по умолчанию.
sort	Сортировка.
userProps	Входит ли в профиль.
isLocation	Является ли местоположением.
propsGroupId	ID группы.
description	Описание.
isemail	Является ли почтой.
isProfileName	Является ли названием профиля пользователя.
isPayer	Используется ли как имя плательщика.
isLocation4Tax	Используется ли как местоположение для налогов.

isFiltered	Доступно ли в фильтре по заказам.
code	Код в системе.
isZip	Используется ли как почтовый индекс.
isPhone	Является ли телефоном.
isAddress	Является ли адресом.
active	Флаг активности.
util	Флаг служебности.
inputFieldLocation	Положение поля ввода.
multiple	Флаг множественности.
settings	Настройки.

sale.property.list

```
sale.property.list(select, filter, order,
navigation)
```

Метод получает список свойств заказа.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.property.list',
    { select:{
                id
          } ,
          filter:{
            code: 'test'
          },
          order:{
                id: ASC
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.property.update

```
sale.property.update(fields)
```

Метод для обновления полей свойства заказа.

Если операция успешна, возвращается ресурс свойства заказа в теле ответа.

Параметры

Параметр	Тип	Описание
fields	nested object	Поля, соответствующие доступному списку полей getFieldsByType.

Примеры

```
BX24.callMethod(
    'sale.property.update',
    {
        id: 34,
        fields: {
```

```
                personTypeid: '1',
                propsGroupId: '8',
                name: 'Должность',
                code: 'Position',
                active: 'Y',
                util: 'Y',
                userProps: 'Y',
                isFiltered: 'Y',
                sort: '100',
                description: '',
                type: 'ENUM',
                required: 'N',
                multiple: 'N',
                defaultValue: [
                        'code1',
                        'code2'
                ],
                settings[
                        multiline: 'Y',
                        maxlength: 100
                ]
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс свойства корзины

Элемент коллекции свойства корзины:

```
{
        "basketId": "112",
        "code": "SIZE",
        "id": "211",
        "name": "Размер",
        "sort": "200",
        "value": "M",
        "xmlId": "bx_5c9267b9eacaf"
}
```

Поля соответствуют доступному списку полей getFields.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.basketproperties.add

```
sale.basketproperties.add(fields.basketprope
rties)
```

Метод добавляет элемент коллекции свойств табличной части корзины.

Если операция успешна, возвращается ресурс свойства табличной части корзины в теле ответа.

Параметры

Параметр	Тип	Описание
fields.basketproperties	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.basketproperties.add',
    {
```

```
            fields: {
                    value: красный,
                    basketId: 18,
                    name: Цвет
                    code: COLOR
                }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.basketproperties.delete

```
sale.basketproperties.delete(id)
```

Метод удаляет элемент коллекции свойств табличной части корзины.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер элемента коллекции свойств табличной части корзины.

Примеры

```
BX24.callMethod(
    'sale.basketproperties.delete',
    { id: id },
    function(result)
    {
```

```
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.basketProperties.get

```
sale.basketProperties.get(id)
```

Метод для доступа к полям элемента коллекции свойств табличной части корзины.

Если операция успешна, возвращается ресурс свойства табличной части корзины в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер элемента коллекции свойств табличной части корзины.

Примеры

```
BX24.callMethod(
    'sale.basketProperties.get',
    { id: id },
    function(result)
    {
```

```
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.basketproperties.getFields

```
sale.basketproperties.getFields()
```

Метод возвращает поля элемента свойств корзины.

Возвращаемые поля:

Поле	Описание
id	Идентификатор товара.
basketId	Идентификатор корзины.
name	Название товара.
value	Значение свойства.
code	Код свойства.
sort	Сортировка.
xmlId	Внешний идентификатор.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'sale.basketproperties.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.basketproperties.list

```
sale.basketproperties.list(select, filter,
order, navigation)
```

Метод получает список элементов свойств коллекции корзины.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.basketproperties.list',
    { select:{
                id,
                name
          } ,
          filter:{
                quantity: 2
          },
          order:{
                id: ASC
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.basketproperties.update

```
sale.basketproperties.update(id, fields)
```

Метод для обновления элемента коллекции свойств табличной части корзины.

Если операция успешна, возвращается ресурс свойства табличной части корзины в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер элемента коллекции свойств табличной части корзины.
fields	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.basketproperties.update',
```

```
    {
          id: 101,
          fields: {
                  value: красный,
                  name: Цвет
                  code: COLOR
                }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс свойства отгрузки

Свойство отгрузки:


```
{
  id: 20,
  personTypeId: 1,
  name: 'test ENUM',
  type: 'ENUM',
  required: 'N',
  defaultValue: null,
  sort: 100,
  userProps: 'N',
  isLocation: 'N',
  propsGroupId: 2,
  description: null,
  isEmail: 'N',
  isProfileName: 'N',
  isPayer: 'N',
  isLocation4tax: 'N',
  isFiltered: 'N',
  code: null,
  isZip: 'N',
  isPhone: 'N',
  isAddress: 'N',
  active: 'Y',
  util: 'N',
  inputFieldLocation: '0',
  multiple: 'Y'
  settings: []
}
```

Поля соответствуют доступному списку полей getFieldsByType.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.shipmentproperty.add

```
sale.shipmentproperty.add(fields)
```

Метод для добавления свойства отгрузки.

Если операция успешна, возвращается ресурс свойства отгрузки в теле ответа.

Параметры

Параметр	Тип	Описание
fields	nested object	Поля, соответствующие доступному списку полей getFieldsByType.

Примеры

```
BX24.callMethod(
    'sale.shipmentproperty.add',
    {
        fields: {
            personTypeid: '1',
```

```
                propsGroupId: '8',
                name: 'Должность',
                code: 'Position',
                active: 'Y',
                util: 'Y',
                userProps: 'Y',
                isFiltered: 'Y',
                sort: '100',
                description: '',
                type: 'ENUM',
                required: 'N',
                settings[
                        multiline: 'Y',
                        maxlength: 100
                ]
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.shipmentproperty.delete

```
sale.shipmentproperty.delete(propertyId)
```

Метод для удаления свойства отгрузки.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
propertyId	string	Номер свойства отгрузки.

Примеры

```
BX24.callMethod(
    'sale.shipmentproperty.delete',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.shipmentproperty.get

```
sale.shipmentproperty.get(id)
```

Метод для доступа к полям и настройкам свойства отгрузки.

Если операция успешна, возвращается ресурс свойства отгрузки в теле ответа.

Параметры

Параметр	Тип	Описание
id	int	Номер свойства отгрузки.

Примеры

```
BX24.callMethod(
    'sale.shipmentproperty.get',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.shipmentproperty.getField

Описание и пример

```
sale.shipmentproperty.getFieldsByType()
```

Метод, возвращающий поля и настройки свойства отгрузки для определённого типа свойства.

Параметры

Без параметров.

Пример

```
BX24.callMethod(
    'sale.shipmentproperty.getFieldsByType',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
```

```
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание
id	ID свойства.
personTypeId	ID типа плательщика, к которому привязано свойство.
name	Название свойства.
type	Тип.
required	Флаг обязательности.
defaultValue	Значение по умолчанию.
sort	Сортировка.
userProps	Входит ли в профиль.
isLocation	Является ли местоположением.
propsGroupId	ID группы.
description	Описание.
isemail	Является ли почтой.
isProfileName	Является ли названием профиля пользователя.
isPayer	Используется ли как имя плательщика.
isLocation4Tax	Используется ли как местоположение для

	налогов.
isFiltered	Доступно ли в фильтре по заказам.
code	Код в системе.
isZip	Используется ли как почтовый индекс.
isPhone	Является ли телефоном.
isAddress	Является ли адресом.
active	Флаг активности.
util	Флаг служебности.
inputFieldLocation	Положение поля ввода.
multiple	Флаг множественности.
settings	Настройки.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.shipmentproperty.list

```
sale.shipmentproperty.list(select, filter,
order, navigation)
```

Метод для получения списка свойств отгрузки.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.shipmentproperty.list',
    { select:{
                id
          } ,
          filter:{
            code: 'test'
          },
          order:{
                id: ASC
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.shipmentproperty.update

```
sale.shipmentproperty.update(fields)
```

Метод для обновления полей свойства отгрузки.

Если операция успешна, возвращается ресурс свойства отгрузки в теле ответа.

Параметры

Параметр	Тип	Описание
fields	nested object	Поля, соответствующие доступному списку полей getFieldsByType.

Примеры

```
BX24.callMethod(
    'sale.shipmentproperty.update',
    {
        id: 34,
        fields: {
```

```
                personTypeid: '1',
                propsGroupId: '8',
                name: 'Должность',
                code: 'Position',
                active: 'Y',
                util: 'Y',
                userProps: 'Y',
                isFiltered: 'Y',
                sort: '100',
                description: '',
                type: 'ENUM',
                required: 'N',
                multiple: 'N',
                defaultValue: [
                        'code1',
                        'code2'
                ],
                settings[
                        multiline: 'Y',
                        maxlength: 100
                ]
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.delivery.handler.add

Метод добавляет обработчик служб доставки.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.handler.add
```

Вызов JSON:

```
{
    CODE: "uber",
    NAME: "Uber",
    DESCRIPTION: "Uber Description",
    SETTINGS: {
        CALCULATE_URL:
"http://gateway.bx/calculate.php",
        CREATE_DELIVERY_REQUEST_URL:
"http://gateway.bx/create_delivery_request.p
hp",
        CANCEL_DELIVERY_REQUEST_URL:
"http://gateway.bx/cancel_delivery_request.p
hp",
        HAS_CALLBACK_TRACKING_SUPPORT: "Y",
        CONFIG: [
```

```
                {
                    TYPE: "STRING",
                    CODE: "SETTING_1",
                    NAME: "Setting 1",
                },
                {
                    TYPE: "STRING",
                    CODE: "SETTING_2",
                    NAME: "Setting 2",
                },
            ],
        },
        PROFILES: [
            {
                NAME: "Taxi",
                CODE: "TAXI",
                DESCRIPTION: "Taxi Delivery",
            },
            {
                NAME: "Cargo",
                CODE: "CARGO",
                DESCRIPTION: "Cargo Delivery",
            },
        ],
    }
```

Ответ JSON:


```
{
    result: 45,
    time: {
        start: 1638524653.148473,
        finish: 1638524653.535473,
        duration: 0.38700008392333984,
        processing: 0.023000001907348633,
        date_start: "2021-12-03T11:44:13+02:00",
        date_finish: "2021-12-03T11:44:13+02:00",
    },
}
```

sale.delivery.handler.delete

Метод удаляет обработчик служб доставки.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.handler.delete
```

Вызов JSON:

```
{
    ID: 33
}
```

Ответ JSON:

```
{
    result: true,
    time: {
        start: 1638371477.76651,
        finish: 1638371478.19051,
        duration: 0.4240000247955322,
        processing: 0.026999950408935547,
        date_start: "2021-12-01T17:11:17+02:00",
        date_finish: "2021-12-01T17:11:18+02:00",
    },
}
```

sale.delivery.handler.list

Метод для получения списка обработчиков служб доставки.

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.handler.list
```

Ответ JSON:


```
{
    result: [
    {
        ID: "50",
        NAME: "Uber",
        CODE: "uber",
        SORT: "100",
        DESCRIPTION: "Uber Description",
        SETTINGS: {
            CALCULATE_URL:
"http://gateway.bx/calculate.php",
            CREATE_DELIVERY_REQUEST_URL:
"http://gateway.bx/create_delivery_request.php",
            CANCEL_DELIVERY_REQUEST_URL:
"http://gateway.bx/cancel_delivery_request.php",
            HAS_CALLBACK_TRACKING_SUPPORT: "Y",
            CONFIG: [
                {
                    TYPE: "STRING",
                    NAME: "Setting 1",
                    CODE: "SETTING_1",
                },
                {
                    TYPE: "STRING",
```

```
                    NAME: "Setting 2",
                    CODE: "SETTING_2",
                },
            ],
        },
        PROFILES: [
                {
            NAME: "Taxi",
            DESCRIPTION: "Taxi Delivery",
            CODE: "TAXI",
                }
            ],
        }
    ],
    time: {
        start: 1638538190.926672,
        finish: 1638538191.315672,
        duration: 0.38899993896484375,
        processing: 0.01699995994567871,
        date_start: "2021-12-03T15:29:50+02:00",
        date_finish: "2021-12-03T15:29:51+02:00",
    },
}
```

sale.delivery.handler.update

Метод выполняет обновление обработчика служб доставки.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.handler.update
```

Вызов JSON:

```
{
    ID: 43,
    CODE: "uber",
    NAME: "Uber",
    DESCRIPTION: "Uber Description",
    SETTINGS: {
        CALCULATE_URL:
"http://gateway.bx/calculate.php",
        CREATE_DELIVERY_REQUEST_URL:
"http://gateway.bx/create_delivery_request.p
hp",
        CANCEL_DELIVERY_REQUEST_URL:
"http://gateway.bx/cancel_delivery_request.p
hp",
        HAS_CALLBACK_TRACKING_SUPPORT: "Y",
```

```
        CONFIG: [
            {
                TYPE: "STRING",
                CODE: "SETTING_1",
                NAME: "Setting 1",
            },
            {
                TYPE: "STRING",
                CODE: "SETTING_2",
                NAME: "Setting 2",
            },
        ],
    },
    PROFILES: [
        {
            NAME: "Taxi",
            CODE: "TAXI",
            DESCRIPTION: "Taxi Delivery",
        },
        {
            NAME: "Cargo",
            CODE: "CARGO",
            DESCRIPTION: "Cargo Delivery",
        },
    ],
}
```

Ответ JSON:


```
{
    result: true,
    time: {
        start: 1638371182.61351,
        finish: 1638371182.99051,
        duration: 0.377000093460083,
        processing: 0.02500009536743164,
        date_start: "2021-12-01T17:06:22+02:00",
        date_finish: "2021-12-01T17:06:22+02:00",
```

```
    },
}
```

sale.delivery.add

Метод добавляет службу доставки.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.add
```

Вызов JSON:


```
{
    REST_CODE: "uber",
    NAME: "Uber Taxi",
    DESCRIPTION: "Uber Taxi Description",
    LOGOTYPE:
"/9j/4AAQSkZJRgABAQEBLAEsAAD/2wBDAAgGBgcGBQg
HBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0aHBwgJC4nICI
sIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/2wBDAQkJCQw
LDBgNDRgyIRwhMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjI
yMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjIyMjL/wgARCAI
VAyADASIAAhEBAxEB/8QAGwABAAIDAQEAAAAAAAAAAAA
AAAYHAwQFAQL/xAAaAQEAAwEBAQAAAAAAAAAAAAAAAwQ
FBgEC/9oADAMBAAIQAxAAAAGfgAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAHz4+tPUw4lrP8Y2dPkY3
jIxjIxjIxjIxjIxjIxjIxjIxjIxjIxjIxjIxjIxjIxjIxjI
xjIxjIxjIxjIxjIxjIxjIxjIxjIxjIxjIxjIxjIxjIxjIxj
J9YXre3eJ93oe0xZdyqH34AAAAAAAAAAAA5W9yceyGL
ZAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA++zw+hqQbo
3qgAAAAAAAAAAAGho7mnzN4KcgAAADT2qsvxWSrZeis
naqyyKcm6M6ZrbMFtfEuVs04LJzVhLK/1KRlWGPJwZfn
pK2bFeyfqtN/49sYYtkfPrVVs3Ktkq2Fp/WjvYtkPj3W
+IhxditZKtkvln5opK8mwEH1j1+fBtOCyVbLXxZe1XFj
50wU5AAAAGxr5ZvnsDrM8AAAAAAAAAAAADnae5p8xfCp
9gAAAfNWWnVm1WDYrLIreyMqxujCtILOoLow8QdDTSyJ
yylLKRzV1we9wbPxCB1NBv6G/F7Yw5LRfP18+qsHY5oF
j72jvcjohH7BOL2uL1WeFj5lUrikr5q8FKThwacwboaY
aMO9Y9cWPhWwypwAAAGXFll87A63OAAAAAAAAAAAAA52
nuafMXwqfYABz67vxWoqtajtKrPu0CrFpvVWWRt1z4sh
VYtSC8WcEGWm9VZLJPFPEtVWLU4MJ7hwlpvVWb9iaB0V
VvFqfNW+nytN6qxaY096uNLxaiqx2eLOe16qxaYi0rik
W8WoqsTiDdyblWLTeq6sfn134tRVYtRVlpU5fRSkAAZc
WWXzsDrc4AAAAAAAAAAAADnae5p8xfCp9gAc6u7Erveq
BqQe2pVdqY1kMayraya21q+mNyqnMGnOdN3Bz1xEpbEr
sUXHS0nd4Xdrfc4HLX3P6HPl+a6HW573z3xag4/SDxW+
lu6XXZwSeTrt8Tt8rfCv9xSKyqK9NRC5H251BZ1z1wM6
bQrmxq53agasC1KrtTGs+jGsgAMuLLL52B1ucAAAAAAA
AAAAABztPc0+YvhU+wAOdXdpaOlBXaxFyOvLU53RoTBQ
lVtZOhdjrlYjRgruc7e3VkzDLnRKW61j4rNYjWr133ZP
mi+tsY1lz+h8fflWrEbdWu/bDHRGDbArfSsb53KtdrEf
Xmr28ObGshF9RSK2Zr69eu1iJ/iLTrT3MuwFX70K5tLR
0oa7WIuRV3anP6NCYKEoADLiyy+dgdbnAAAAAAAAAAAA
Ac7T3NPmL4VPsAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ABlxZZfOwOtzgAAAAAAAAAAAOdp7mnzF8Kn2ABx4FPI
FvVPXjRh9eD14PXg9eD2z6utHJscOFTWF2Pjx6vRePR4
9Hj0ePR7atVWrj2YvEZdEbcYXYgDNkNVsa4BsWdWFn4t
mJRaUxa/F7YVfXZbjiiVxY+WpIDlRyzsZSrLiN+x63sj
EtBlzgAMuLLL52B1ucAAAAAAAAAAAABztPc0+YvhU+wA

OPAp7AugputybY0oYJo3HCCCA6W3LJSU/zbLrkw2hV9o
ZFjiQuaQux8OnzLYvxQBbIqbBcAo7y5q5I+D21KrtTGs
xeJS2JXI2zrWNdig65BUy1vCqtS4OeVCDPZ1Y2diWolF
5RF78P1eFH3hcjVTa1UnD3NPZLpBUnI6nLN6x64sfCth
lTgAMuLLL52B1ucAAAAAAAAAAAABztPc0+YvhU+wAOPA
p7AugprYqe2NKHtQibwgggLJk8VlBHa4sauTDaFaWVlW
OJDZnE5vj4tSrbPvxdzW2IMSrdpboFuYMvyU3h7vEPm0
azsvGsxiKSyM3I8dgQGfXYpW8jBJfKW2C4udUuY+H0Mt
lVvZGJaiUck8bvw/F20tdFyP2FzQV3IpEGplrI4XyG9Y
9cWPhWwypwAGXFll87A63OAAAAAAAAAAAAA52nuafMXw
qfYAHHgU9gXQU1sVPbGlD2oRN4QQQFjSmLSgj1b2HXgs
qtbKybHHh8tic/wAfVn1XaV+LsQec8cqjdmXYOs+uMQb
je+H1Z1Y2bkWY1GJPF7UWCxK7sW7HKoxJtIqHYsP0rVZ
MENEGxY9cWPjWYtHpFGb0X1dVK3Pbj++F16vLB6tQ28c
mo7yqg4gN6x64sfCthlTgAMuLLL52B1ucAAAAAAAAAAA
ABztPc0+YvhU+wAOPAp7AugprYqe2NKHtQibwgggLJlE
XlJFK4sitz6s2r7QyLHEhkzhdj4y2nU9sX4u157DiXqk
3C0YRM8hR3vc4Jls+rLUxrMaiEsiVyNY9cWPdilPvvBO
981gLQqXa4hi9+RtWXWNnYlqJRyRxe/DmuukLwuR/FWW
tVJw7WqmTFmRqS+FHN7RN6x64sfCthlTgAMuLLL52B1u
cAAAAAAAAAAAABztPc0+YvhU+wAOPAp7AugprUquf6UM
zhc055Trcyk7k2tskSrqYQ88tCr7QyLHEhc0hdj4WxU9
sX4u1A55CyAt/fJ32cWUgMLlkTPbUqu1MazF4lLYlcjW
PXFj3YpXFZVFitn2Ph9+HyDPZ1Y2diWolF5RF78PtiV0
uR2LC+cH18iw/quh3+AG9Y9cWPhWwypwAGXFll87A63O
AAAAAAAAAAAAA52nuafMXwqfYAHHgU9gXQU3Q57Shu7J
U9gHYeeH1q8uvjU1w8tCr7QyLHEhc0hdj4WxU9o34pC1
hstb4NzDw4KaeqHtqVXamNZi8SlsSuRrHrix7sUrAA5f
U5ZUIM9nVjZ2JaiUXlEXvwhcjAAAA3rHrix8K2GVOAAy
4ssvnYHW5wAAAAAAAAAAAAHO09zT5i+FT7AA48CnsC6C
mGlCB9eeAADy0Kws/IscSFzSF2Ph56vxePR49AAHtqVZ
aeLZi8SlsSuxvr5XYvt8D7fA+/PkAZ7OrKzcS1EovKYt
fhC5GAAABvWPXNjYVsMqcABlxZZfOwOtzgAAAAAAAAAA
AAOdp7mnzF8Kn2AB49evHo8ejx6PHo8ejz0PHo8ejx6P
Ho8ejx6PPQ8ejx6PHo8ejx6PHo89Dx6PHo8ejx6PHo8e
jz0A8AAMuLLL52B1ucAAAAAAAAAAAABztPV4mBbkqNK/
3JUaElRoSVGhJUaElRoSVGhJUaElRoSVGhJUaElRoSVG

hJUaElRoSVGhJUaElRoSVGhJUaElRoSVGhJUaElRoSVG
hJUaElRoSVGhJUaElRoSVGhJUaElRoSVGhJcsV3ZPJqO
jpAAAAAAAAAAAAAQSEWrVQAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAA7/AnpNQAAAAAAAAAAADEZWjgOrVM60it
m9ogAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAzGW4In1j
uuTnN9gzgAAAAAAADn9DhHxhAAD54EhEE1rE9K5WMK5
WMK5WMK5WMK5WMK5WMK5WMK5WMK5WMK5WMK5WMK5WMK5
WMK5WMK5WMK5WMK5WMK5WMK5WMK5WMK5WMK5WMK5WMK5
WMK5WMK5WMK5WMK5WMK5WMK5WMK5zz/wi8izAABkxjpd
OP90yAAAAAAAAAwanSHEwSIRlI8RwXYxnLb+M1Gf4Mb3
wAAAAAAAAAAAAAAAAAAAAAAAAAAAHp4+8hgbeQ0HTyHId
3KcDP3By9raAAAAAAAAAAAAAAAAD5+hi+NganzujQ+ei
OZ89Ucn57A4vnbHD+e8OB5IBHkhEd8kYjaSCNpII2kgj
aSCNpII2kgjnsiEeSER/3vjg+90cT3tDj+9ccr3qDm/X
QGl9bY1/rMPj7AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAD//xAAyEAAABQEGBQMDBAMBAAAAAAAAAQI
DBAUQERITFDQVIDIzQDAxNQYhYBYkQUMiI1Cg/9oACAE
BAAEFAv8AxgqfIgbyzGNQxqGNQxqGNQxqGNQxqGNQxqG
NQxqGNQxqGNQxqGNQxqGNQxqGNQxqGNQxqGNQxqGNQxq
GNQxqGNQxqGNQxqGNQxqGNQxqGNQxqGNQxqGNQxqGNQx
qGNQxqGNQxqGNQxqGNQxqGNQxqGNQxqGNQxqBOr
IJkAjJReeZkkluGv8DSo0mhZLLznV41fgiVGhRHeXmOq
wt/g0dX28yR6urjjVxxq441ccEZKK1b7TatXHGrjjVxw
h1t3kWtDZauONXHGrjhMllSuTVxxq441ccauOCMlFat9
ptWrjjVxxq44Q627yLWltOrjjVxxq44KSwpXqMn/s8yR
1emftyRNnbVd7bRum2rbO2FvbT9uSHs7arvraNyVbZWw
976jXc8yR1emftyRNnbVd7bRum2rbO2FvbT9uSHs7arv
raNyVbZWw976jXc8yR1emftyRNnbVd7bRum2rbO2FvbT
9uSHs7arvraNyVbZWw976jXc8yR1emftyRNnbVd7bRum
2rbO2FvbT9uSHs7arvraNyVbZWw976jXc8yR1c87Y3mL
zF5gjO+4hcQuIXEJR/u7zF5i8xS/vCuIXELiFY+yrzF5
i8xSvvMuIXELiE0v2V5i8xeYIzvuIXELiFxCWf7y8xeY

vMUv7wriFxC4hWPsLzF5i8xS/vMuIXELiEwv2d5i8xeY
vMF7c7Xc8yR1c87Y2l78kveW0rZW1nqtpO8tnbG0vfkl
7y2lbK2s8lK3ts3Zche3O13PMkdXPO2Npe/JL3ltK2Vt
Z6raTvLZ2xtL35Je8tpWytrPJSt7bN2XIXtztdzzJHVz
ztjaXvyS95bStlbWeq2k7y2dsbS9+SXvLaVsrazyUre2
zdlyF7c7Xc8yR1c6kpWnQxRoYo0MUaGLyqhx1q0MUaGK
NDFDbaGk2usNPDQxRoYo0MUNxmWVWqSladDFGhijQxRo
YvKqHHWrQxRoYo0MUNtoaTa6w08NDFGhijQxQ3FYaVap
JLToYo0MUaGKNDF9FrueZI6vwZrueZI6vwZrueZI6uep
rU3B1cgauQNXIGrkDVyBq5A1cgauQNXIGrkDVyBq5Aa+
7NYWtuLqpA1UgaqQNVIGqkDVSBqpA1UgaqQNVIGqkBMl
/EKy640jVyBq5A1cgauQNXIGrkDVyBq5A1cgauQNXIDE
p85ArLrjatVIBSpF+IxiMYjGIxiMYjE9L7Y1UgaqQIUh
5U30Gu55kjq56psLhcLhcLhcLhcLhcLhcGuzWNrcLhcL
hcLhcLhcLhcCL/IVnouFwuFwuCGsQuZBtJuuFwuDBfuB
Weq4JIsWWgZaBLqrUWTx1oQ5kWaMtAWw04241lO3CFvf
Qa7nmSOrnqexsapkt9rg84SIb8S2PAkym+DzhIgSYrdj
XZrG1sap0t5vhU4cKnDhU4Lp8tAMrjtLqFZ6LG47zpaK
UNFKC4km448hI0kgjciSEWsbgVnqCeqyr/KCI6bEuyqF
hqYhb30Gu55kjq56nsbKR8WPqL2soGwFdLFDMiNIa7NY
2tlJ+L5H4jElNRpioR2F1Cs9FlA2Nn8WVD42xjcCs9QT
1WVf5QR0G7JsqKsdRELe+g13PMkdXPU9jZSPix9Re1lA
2F4rW1/qDXZq+1IrxlqFJ+MDj7TI10YJWlZB1tLzTzC2
XCQoxhMjFZ6CSahgUKDsiBquGYgZiRjbE9aVQDQoEhRh
pJk+Kz1ZahhNKrJ1HelTP0/IFPpSIShJfTGjGZqUIW99
BrueZI6uep7GykfFj6i9rKDsBWdr/AFJ+xN9qr/eMMBY
aV8YYrxXqwkIkpcd6yrJwVL3Cf8FCsdJ/cywpFB2Qrl2
iUeJX9pFefUd1w6iZ/wAXxVzuPBebf2dL25HXW2W6lUT
mrshb30Gu55kjq56nsbKR8WPqL2soOwIVvaf1f0t9uqd
n+DK8Ur700x9QWIQa5HsQqTmZUT7Zdv8Amrj+RQdkQrp
lollcr+5vuf1oH9THUKt3PvdfhOyRVo0Z+JUGJpipRtV
Cthb30Gu55kjq56nsbKR8WPqL2soWwFcvOEof0t9ur7Y
XoFL+N/iZBRMHAGBHp0eK4J81MOOk7wfbLt/zVjIk9KB
QdkJMZqW2dEhmODxMRUWGQltJZfR7/wBTHUKuV54iMES
TCh7Cr/KU+TpZtlVjaadZC3voNdzzJHVz1PY2Uj4sfUX
tZQNhcK99oAJRpDXarG1IzIZhilfem2XC6yp0lagR3GZ
mYQq4xV1GSPeygbG4XWXWVMz4gR3GajUI53Pis9WYoYz

M/cYRV/lBSZOpgitRs6HZC3voNdzzJHVz1PY2Uj4sfUX
tZQNgK/sLGuzWNrZSfixW5L0dXEpoj1mU04hZONirxyj
zwXUKz0WUDYirSnYkXjc0cbmjjU0OuqedsY3ArPUE9Vl
X+UFEk5MwGRKKZHOLKELe+g13PMkdXPU9jZRlX0sfUKf
9VlCK6nD6gV+zsa7NY2tlJ+LH1F12Ur4wfUJf7AXUKz0
WUDYiv7DmY3ArPUC+x/qCMP1BGE59MqYCM0qT9QMYf1B
GFUmMTViFvfQa7nmSOrnqexs+n3r2RNilMivRH4640GR
KXHZTHYH1A9ifsa7NY2tlJ+LFdYdeVopQj0iU+tttLTY
r7mKYC6hWeiygbEV4jODhUMKhhUMKrWNwKz1enC3voNd
zzJHVz1PY2QpRw5TbiXW+SQ+iMw+8qQ/Y12axtbKT8Xy
uuoYakvqkyAXUKz0WUDY8tS+NsY3ArPV6cLe+g13PMkd
XPU9jbAqTsI49UiSCIyMGoklIq8Vgpk52a5a12axtbKW
80mm6hkahkahkKlx0h6tRGim1B6aqwuoVnosoGx5al8b
YxuBWer04W99BrueZI6uep7HlIzIGZnzNdmsbX0y6hWe
iy8yGJQxKGJQxKGJQxKtY3ArPV6cLe+g13PMkdXPU9j6
jXZq+29MuoVno9NjcCs9Xpw976DXc8yR1ehcLhcLhcLh
cLhcLhcLv+m13PMkdX4M13PMkdX4M13PMqU5MV7jDY4w
2OMNjjDY4w2OMNjjDY4w2OMNjjDY4w2OMNjjDY4w2OMN
jjDY4w2OMNjjDY4w2OMNjjDY4w2OMNjjDY4w2OMNjjDY
4w2OMNjjDY4w2OMNjjDY4w2OMNjjDY4w2OMNjjDY4w2O
MNjjDY4w2OMNjjDY4w2OMNjjDY4w2OMNjjDY4w2OMNjj
DY4w2OMNjjDY4w2OMNjjDY4w2OMNjjDY4w2OMNjjDY4w
2OMNjjDYiVRD0rzPqFH+X4NRkY6n5lXj6iB+DfT7FyPN
qkLRyfwSNHXKfZaSwz5JuISDlIByzByHDD6dQ3JirjK/
AmmlvLiRyiIJ9wgUpQKUkE82rxXJNwU4tfMaSUT1LSoL
gSUDIeIZLoyXRkujJdGS6Ml0ZLoyXRkujJdGS6Ml0ZLo
yXRkujJdGS6Ml0ZLoyXRkujJdGS6Ml0ZLoyXRkujJdGS
6Ml0ZLoyXRkujJdGS6Ml0ZLoyXRkujJdGS6Ml0ZLoyXR
kujJdGS6Ml0ZLoyXRkujJdGS6Ml0ZLoyXRkujJdGS6Ml
0ZLoTDkKDVKMNtIZTypWpIRKBHeXguMqb/EUNqWEJwI8
JTKFA4gOO4QNCi/CSSZgmHDBRDCY7afJNKTBsNmDioGk
GlUNM4MlwYFF/2cKhlODTuDSrBRAUVAKO2QJCS/42FIy
0GMhsadsaVsaVI0hDSDSGNKoaVY0zg07g07gyHBkuDKc
GWsZaxgUMKhhMXGLjFxi4xcYuMXGLjFxi4xhMYVDAoZa
xlrGU4MlwZLgyHBp3BpnBpVjSrGkUNININIkaVA0zYyG
xlNjCn/wAf//EACsRAAAEBAUFAQACAwAAAAAAAABAgM
REyAyBBAUMVESITBBYSJQgDNCUv/aAAgBAwEBPwH+miG

P+h0kIEIEIEIEIEIEIEIEIEIEIEIEIEIEIEIEIEIEIEIEIEIEIEIEIEIEI
EIEIEIEIEIEIEIEIEIEIEIEIEOkgtgj2BlDsfnYT/ALfwb6e
0fO1YVTrnQNQYbcNZ5OL6SiNQYQ8alQyUcCiNQYJ8456
gxqDBdyyU+ZHAagwhXUmOTjppOA1BhtfWVS7T87VhVYj
1lh98n7cmb8nLTyTvSnbJdx5M2ZP3ZYfapVp+dqwqHHe
g4DUfB/mGn+iEnuNR8HVN/I0/0S5f6Go+Cb1/kaf6JHT
3iNR8Go+DT/Rp/onw7QGo+CV1/oaf6Jkv8jUfB0Tf0NP
9EZPYaj4G3OuhVp+dqwqMRdlh/eWI2LJi7J6zJu4slWn
SrfJu0snrzyYtyxG+WH90KtPztWFQ40ajiNOoNNmjfJ1
BrLsNOoNtGk45OJ6kwGnUEsmRxyMokNOoadWZsHEadQS
UChktk1KiNOoNp6Shk62ajGnUGmzRvQq0/O1YX8Gq0/O
1YVD5/oRMRMRMRMRMMH3MPH+hExExExExExh/YeM+sRM
RMRMRMRMMHuHTPrMRMRMRMRMRMMbUKtPztWFQ/dVh9zD
19WH9h6+rD+w7edWH2oVafnasKh+6pj2HrqsP7D19WH9
h286sPtQq0/O1YVD91WH3D19WH9h6+rDh286sPtQq0/O
1YVD91WH3MPX1Yf2Hr6sP7Dt51YfahVp+dqwqH7qsPuY
evqw/sPX1Yf2Hbzqw+1CrT87VhUP3VYfcw9fVh/Yevqw
/sO3nVh9qFWn52rCofuqw+5h6+rD+w9fVh/YdvOrD7UK
tPztWF/BqtPztuJJMBNRyJqORNRyJqORNRyJqORNRyJq
ORNRyJqORNRyJqORNRyJqORNRyJqORNRyJqORNRyJqOR
NRyJqORNRyJqORNRyJqORNRyJqORNRyJqORNRyJqORNR
yJqOQp1MP7+f/xAA2EQAAAwQHBgYCAgIDAAAAAAAAAQI
DBAUzERMUFSBRUhASMDRxgSExMkJioSNBIlCA8DVDYf/
aAAgBAgEBPwH/AAzMyIqTDxElGdDIKbtFeahWLzFYvMV
i8xWLzFYvMVi8xWLzFYvMVi8xWLzFYvMVi8xWLzFYvMV
i8xWLzFYvMVi8xWLzFYvMVi8xWLzFYvMVi8xWLzFYvMV
i8xWLzFYvMVi8wTZoXkoMIktPg08SCFkst5PHibf/AKi
/o4a3NK6s/I+O/HS8KxObol4ppPyF1I1B8c0sEkZHsdG
BN2m6YupGoPMPSyZGsj2MWdY0JGYupGoLhiEpNVO26ka
hdSNQWW6oy2MYahbMlU+YupGoPLImTU0FsdHFLdnvmYu
pGoPjsTBRER4nc6GqevHfZ6sUJ9/bZFfQnZDJ3bZEOXV
/v72Ok9OxtLV0wtZh7HWSnpsf+YV/v62QyR32RWYXTEw
mp68d9nqwOrlaE71NAun5/Q/4/wCW92F7fD7G/eH8PTQ
Lp+f0Kiw/lpp/Qvb4fYtVs/BRRSLp+f0LDZ/zb1NAvb4
fYvKs/Hu+fgLp+f0Lp+f0L2+H2L2+H2LtrP573mLp+f0
LfUfi3aaPAXt8PsWS1/npopF0/P6FfYfxUUi9vh9irt/
8/TQLp+f0Ht0s9HjTTgYTU9eO+z1YIVLPrsi3s77IV61

bInI77IfzCdj3IVsYzE9cLKWWx6nq67HDl07InP7bIVL
Prsi3s74GE1PXjvs9WBzfEMEGlRC9WWRh9ekt6N39bHJ
5SwUZqF6ssjD2/IbM90i2OzUmTUlmL1ZZGG0RZtGZpIv
PYzVurJQvVlkYvVlke1ETZpSRUGL1ZZGGyyW0NRfvY7R
BDJkSDIXqyyMPbcmzTeLY5viGCTJRC9WWRh9ekt6N0vL
Awmp68d9nq/o2E1PXjvs9WCGoSbHxL9irRkKtGQq0ZCr
RkKtGQiiSIk0CHJSbDxIVachVpyFWnIVachVpyEVSRbt
H/ocEJNgVJCrRkKtGQq0ZCrRkKtGQiiSJSaA4oSbumkh
VoyFWjIVaMhVoyFWjIRRJE0KjAwmp68d9nqwQuT3xRb0
pENkYot7O4h/LliivqSHDl04orMLpgYTU9eO+z1YIXJ7
4ooXgkQ6Rii3t7iH8uWKK+aQ48unFFZhdMDCanrx32er
BC5PfFFfQkQ2RiivtDhy5Yor5kHKQnFFZhdMDCanrx32
erBC5PfFFvSkQ2Rii3s7iH8uWKK+pIcOXTiiswumBhNT
1477PVghcnvii3pSIbIxRb2dxD+XLFFfUkOHLpxRWYXT
Awmp68d9nqwQuT3xRb0pENkYot7O4h/LliivqSHDl04o
rMLpgYTU9eO+z1YIXJ74ot6UiGyMUW9ncQ/lyxRX1JDh
y6cUVmF0wMJqevHfZ6v6NhNT14706NltlKSkWF40iwvG
kWF40iwvGkWF40iwvGkWF40iwvGkWF40iwvGkWF40iwv
GkWF40iwvGkWF40iwvGkWF40iwvGkWF40iwvGkWF40iw
vGkWF40iwvGkWF40iwvGkWF40iwvGkWF40iwvGkWF40i
wvGkWF40iwvGkMnNuTQjNP+fn/8QAPxAAAQICBgcECQM
DBAMAAAAAAQACAzMQESAxMpISITBxcoGRQEFRcwQTImB
hgqGxwTRCUiNDUBRioKJTY9H/2gAIAQEABj8C/wCGD7O
tXrEViKxFYisRWIrEViKxFYisRWIrEViKxFYisRWIrEV
iKxFYisRWIrEViKxFYisRWIrEViKxFYisRWIrEViKxFY
isRWIrEViKxFYisRWIrEViKxFYisRWIrEViKxFYisRWI
rEViKvXtBav8AAVlfD3DrH+A+A9xawq/eQt7a0bWdD6q
fD6qfD6qfD6oEGsGxoviNafAlT4fVT4fVT4fVHQe11Xg
bFb3Bo+Knw+qnw+qnw+qDWxWEnursz4fVT4fVT4fVT4f
VAg1g2NF8RrT4EqfD6qfD6qfD6o6D2uq8DYre4NHxU+H
1U+H1U+H1QDYrCT8dqO2jaGzB4BY+UWI3Kx84sQeKwbM
HhFg8IsRuVj5hYg8W1b20btobMHgFj5RYjcrHzixB4rB
sweEWDwixG5WPmFiDxbVvbRu2hsweAWPlFiNysfOLEHi
sGzB4RYPCLEblY+YWIPFtW9tG7aGzB4BY+UWI3Kx84sQ
eKwbMHhFg8IsRuVj5hYg8W1b20bthG4VeVeVeUNZVwVw
VwVwUbjKvKvKvK1/yKuCuCuCg1fFXlXlXla/4lXBXBXB
RuFXlXlXlXlXBXBXBXBRuIq8q8q8rX/Iq4K4K4KDV8Ve

VeVeVr/iVcFcFcFG4VeVeVeVedi3to3bCNw2BZjcZsfM
bEHnY+Q2I3DsY3GbHzGxB52PlNiNw7VvbRu2EbhsCzG4
zY+Y2IPOx8hsRuHYxuM2PmNiDzsfKbEbh2re2jdsI3DY
FmNxmx8xsQedj5DYjcOxjcZsfMbEHnY+U2I3DtW9tG7Y
FrhW03hSGqQ1SGqS2yXOgtJN6kNUhqkNWjDaGjwsD1jA
6rxUhqkNUhq0ocMNNgtcKwe5SGqQ1SGqS2yXOgtJN6kN
UhqkNWjDbojwsD1jA6q6tSGqQ1SGrShww02C1wrBvUhq
kNUhqkN2Le2jd7jt7aN3uO3to3bB7mOLTWNYU+JmU+Jm
U+JmU+JmU+JmU+JmU+JmU+JmU+JmU+JmU+JmU+JmTOEJ
hY4tOn3FT4mZT4mZT4mZT4mZT4mZT4mZT4mZT4mZT4mZ
T4mZT4mZD+tEzUQtB7m6zcVPiZlPiZlPiZlPiZlPiZlP
iZlPiZlPiZlPiZlPiZlPiZlDBjRKtIfuog6D3NrruKnx
MyH9eJmV6vV6vV6vXrYUaJo940rlPiZlPiZlBDoryNK7
S2Le2jdsH7xtWcITOPaCiDvNqvuorZYh8Qog86BvWBvR
YG9E+D/pQ7R761+j+qOg0B4vaQsDeicwsbU4VXJ8M3tN
VEHi2Le2jdsH7xSIkOFW03HSCk/8AYJvrmaOldrp04MP
SbXVepP8A2C040PRbXVfSzhCZx0iJDgktNxrC/TnqF+n
PUL9OeoXtejxOQrVRvsCiDvNNcOE948WtX6aLkX6aLkT
WCBFyqpvosTKg4ejxdd40U4mBEDR36NMPiFEHnQN9Mbl
9qIURvc6mPvog8Wxb20btg/eKYG4/ej0fnS7zDQwf+wf
YqsUM4QmcdMDd+bNUWGHfHvWm32oJ7/CkUQd5pf5lkBe
kcBph8Qog86BvpjcvtRDYLy4Uxz/uqog8Wxb20btg/eK
YG4/ej0fnS7zDRD8z8FHfQzhCZxrVRA3fmgesiNbX4lT
4eZVscHD4UOhvFbXCop7HftNVArFEHeVqof5lGtYx1WM
dVib1UcBwJ0D30w6/5CiDzVyFY76XxmxIYDvFTYX1XrH
O04vj4UPiu/aESbzrog8Wxb20btg/eKYG4/ej0fnS7zD
RD8z8FHei7wTK/wCITONaDear0lA3fmiBrqFRXsu1psR
puPtjxFMVzT4V9FpOch3tNEHeVoDUAtRrKf5lDa//ACD
uRK5KpVXNC0mOuWk46lDINbS4UQj366l7R1oNGsV2i+I
4NaO8rRbWILbh4/GmDxbFvbRu2D94pgbj96PR+dLvMP2
oZ5n4KO9HemcIUPjUTxQ/gFBPw/K5KBzoLG3u1IUekkd
2romocVEDiUSh/mLkm1iv+oiFyQR3o7kN6Z5go9H5p1V
9etBo8dZpdCfp6TfBqcIWlW3xFD2fuHtN32IPFsW9tG7
YP3imBuP3o9H50u8w0NA/mEGjuR3pvCEzjWm3mrioFXh
+aG6byNFTYn0Re0Ev/k6gu/uHCE8m+pNQ4qIWqvWUa73
UP8ygMi16INeoqstfmVdT8yB0X5lHhswtdUEdyG9M8wU
Qh366l7Y1rVXqpjcvsmRP23O3UuqwP9oUweLYt7aN2wf

vFMDcfvR6Pzpd5hobr/uCjUmbgmca1LuUA/D82jHhPc8
97XfhalrXOiFvNL/MtRx/uVYWtQ+IUQea8UB8aY3L7UN
rPts9l1HrBiha+XfTB4ti3to3bB+8UwNx+9Ho/Ol3mGh
vmClnCEzjpgbvzRB9VEcyuuupfqXoGI/1jO8FNe3C4Vi
h2iKmv9oUCiDvNL/Moa+EQHadWsLG3Ksbcqxtyp0R+J2
s0w+IUQedA30xuX2o9WcMXVzoINxUSF3A6t1EHi2Le2j
dsH7xTC+FY+tEB3g4imvxeaIbfF/4pZwhM46YG780ej7
jTAr/AI0QD8DQKIO80v8AMob5gtw+IUQedAUuL9FLi/R
PjMBAd40BwvGsIaUKJX31VKXF+iY+Gx7XDUa6IPFsW9t
G7YP3imLB72nSFDoR1HuPgVoxIThy1INZDNXe4jUEyE2
5oqohwR+wVnnSzhCZx0wN35og+rhvfVXhC/TRcqAdDMN
ne5ybDbhaKhQxn8WUCiDvNL/MobUP7gWF3RYXdFhd0WE
9KYfEKIPPaQeLYt7aN2wfvFLYo1i5w+Ca9hra7WDZdFi
HUE+K+9xrpZwhM46YG782nRIhqa1PjO/caBRB3ml/mWv
SOA0w+IUQee0g8Wxb20btg/eLFWKEb2rVFDXfxdqWpVk
1I1P9Y7wYq36mi5o7rDOEJnHTBDojAau8qazMprMymsz
LXHhj5l7LjEPg1e37LBcwUiiDvNL/ADLXpHAaYfEKIPP
aQeLYt7aN2wfvFrUSFrNdpnCEzj2gog7zTqJWJ3VYndV
id1WJ3VYndViPWmHxCiDz2kHi2Le2jdsH7xtWcITOPaC
iDvO0h8Qog89pB4ti3to3e47e2jd7jt7aN3uO3trGuY4
1tr1KU9SnqU9SnqU9SnqU9SnqU9SnqU9SnqU9SnqU9Sn
qU9SnqU9SnqU9SnqU9SnqU9SnqU9SnqU9SnqU9SnqU9S
nqU9SnqU9SnqU9SnqU9SnqU9SnqU9SnqU9SnqU9SnqU9
SnqU9SnqU9SnqU9SnqU9SnqU9SnqU9SnqU9SnqU9SnqU
9SnqHDENwLj22A/ePceH/ALQT211WJntD3HiekH93sjt
3sj+k/W3/AOe4rYTLz3+CbCZhaKu1a3BagStTQr0WRCS
Fr1t7ne4egwVleyfbN7liWsBawQsXZama/itbrVThWFX
Bdo/A3KXXw61KflUp+VSn5VKflUp+VSn5VKflUp+VSn5
VKflUp+VSn5VKflUp+VSn5VKflUp+VSn5VKflUp+VSn5
VKflUp+VSn5VKflUp+VSn5VKflUp+VSn5VKflUp+VSn5
VKflUp+VSn5VKflUp+VSn5VKflUp+VSn5VKflUp+VSn5
VKflUp+VSn5VKflUp+VSn5VKflUp+VSn5VKflUp+VSn5
VKflUp+VSn5VKflUp+VaoTuepVxn8mrRY2oWtRqXt9VW
OxeI8fdHUEG9jwr2XdVdWtbT7k6gVhq3r2ndFdXv7TrA
WFd61PV4VwWArCen+ZwlYCrleFretZKwrU0f4a4LAFhV
y71eVjWP6LEFiC7l3K76rCsKwFYCsDuiwHosJ6LCeiuK
uKuKuKuKuVyuVyuKuKuKuKwnosJ6LCeiwO6LAVgKwrCrvq

u7qu5XhYgsf0WP6LEV3q76rCsAWEf8AD/xAAtEAABAgM
GBgIDAQEBAAAAAAABABExUfEQIUFhofAgMHGRscFQgUB
g0eGQoP/aAAgBAQABPyH/AMYBIAclgrsDqwUgdFX1XVX
VXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVX
VXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVX
VXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVX
VXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVXVi
HJLkbvuwD9Ddyn2I4j8+7BlfoowgQwgfzXhom4fo7hGF
4/NPyc2jVRKolUShlgHBGPBiw24FUSqJVErOEr5uC9dj
OTKiVRKolXGEgF88NEqiVRKolDLAOCMeDFhtwKolUSqJ
WeNXzcGDIXJlRKolUSjJSYADfzWs6783TczTcO4S4IvB
az34IeyPIzRc1QN/75BaTzdV8Tmm4dwlwReC1nvwQ9ke
Rmi5qgb/3yC0nm6r4nNNw7hLgi8FrPfgh7I8jNFzVA3/
vkFpPN1XxOabh3CXBH4LWe/BD2R5GaLmqBv8A3yC0nm6
r4bArCrCrCzETVIVIVIVIRgIEtMzVYVYVYQgBBziKkKk
KkKULoLpKsKsKsI3BeGN9KkKkKkIQMAAUrCrCrCYJk1S
FSFSFSEQABIE5VhVhVhCAEHOIqQqQqQr6Blukqwqwqwj
LJOMb6VIVIVIQgYAAvwVYVYVYVYUDkar4nI1Q4dgnwQ+
C0HrwR9kOQEDrycQ+Aj24cG66cGo8MDpyNV8TkaocOwT
4IfBaD14I+yHICB15OIfAR7cODddODUeGB05Gq+JyNUO
HYJ8EPgtB68EXZDkBA68nEPgI9uHBuunBqPDA5Gq+Fwb
YBiY8EzLXDloE5HE8EzN64XZwFSw9JwTM3dNZxLgG2CY
ljyZmbLQJyOPBMyy467ODwrRwTM3DtZxwD7BMBx5szM6
r9kzVfsmar4XBUCwRjFV0q6VdKulXSrpV0q6VdKulXSr
pESSXJIT9IoGAHawKqNVGqjVRqo1UaqNVGqjVRqo0XMp
7CGdLl08FVaqtVWqrVVqq1VaqtVWqrVVo3kQiCV99l2m
nvB4Ks0dOJ1n+6z/dZ/us/wB1n+6z/dCGh92f2GSrNVm
j5aLkTHk6r4XNgmmSCZIJkgmSCZIJkgmSCZIJkgmSCZI
IgxuC2SSB9iBTJBMkEyQTJBMkEyQTJBMkEyQTJBMkEKA
IiwQdhcEyQTJBMkEyQTJBABzBWbmfNGXYiSZIJkgmSCB
BH+9gguDD1TJBQGIFTCphHbdEAO4eSy/b/ENAaOHfrmq
YT4NCuYppIdndIpkggEIQ8nVfC5vE7RLsOCy+7NDijEA
u3S0oaF40X/ay+7NBDwYLTf8AVhgVsklsMjbeOHBearc
3tbm9rc3tA3ZT9BEIAgIghuDWCzYJC0nFpZwQ6qpVUrk
EY30XewM6YeC45UcSyRsBbs87NB62aNbuMlhYGIHzBuI
tGCMRPcCzQeTqvhc3idu0TWar1t2iQszWUMXIYtFGBWy
SWwyNus+XDlLzLnQp+pMYHE5H+26wWbBIWk3UeAgXUQi
CcwTl4F1DIvK7jxW7POzQetmjW7jJYGdw2trbw8V3qzQ

eTqvhc3idu0TWar1t2iQWQIn6RA0CMCnQaC3GRRCYHKy
x0dAwjcQXlYeAhBcIF/gWf5CcW2wALTk5EII7XZrDw4v
s2CQUKOiIEsLpFa54CIxi4QRdgGZCJcE9C6J0KlEZ9AX
Agg7OMkBcC6ZR/C6P92aD1QIHudULlly1mUzBz3ACzkG
CYMYdNkIZwCZwCOG5HHOzQeTqvhc3idu0TWar1tgtjCw
e2QNAgBbo6oXMUepBPmCPgou8KZXoIMWREkj/AKIlnAu
iBNIzI+QMAEM9E3BsVFXAUYdQTNuTwWIu1BxJC4t1yG8
MWifJGGgy3mQUfMEMfA8AoTBFNdu2Cb5kbwCncXjFcmh
OAXL/AFQJYkIErBoUIV+AvTiC5PxEVOhEPhKXkSmt0Hk
6r4XN4nbtE1mq9bdikUf0uxAoCGx0W9SWn+ChsZq/CWF
cmAgwew+y9ye8keiBYghAae4DMpmMghh19IEMeEAvNsM
xKDrerTeZBepAXIYGfIpnIBYNoWUaZR9dFDfZrH6CxcG
8gmIEfsIBIK0CHdHvBB0c5gCQ1dYywcf0ttwaDydV8Lm
8Tt2iazVetsdsYLcViJL2OBRgAhxEZlbHRb5JGx8ngou
8Z4E7hdCjEYA7yUXSRdpgRc16MWGjQ9kOORdgjgnUA3E
8tHoiPDkRJmvNsMxjojreXTlF7Atc8BbighFgJenBwOd
T6aeSOic60Loi8hfBa2ihvs1j9BZFD0IxCzEIy7RIG/q
iYKJbjInqLEfube1vi2pxHe3QeTqvhc3idu0TWar1t2i
QWQp95f4GwAXRUVifEthkURciFkl9J12C/AZkAwZEfSz
FdSIdCxevOc/b0ivRI65IogwIZcRY1pjgZIkk5LmzWPA
XUs19nVciHCKNNw9QiPDFNDoIoVxBuyM1iVoPVBi8jqC
FygDIDNEIE3GSx0vaFA9rH8O6/r6t0Hk6r4XN4nbtE1m
q9bdokLN0kbDArZJLYZG3WfKwdtnOQW5CNgeuYdsijZu
ATI2XZDaGcdbNYLNgkLdY8CwhYAd5cxVHqj1T6N2CZwB
rdnnZoPWzRrdxksvnuOzD2LBlOBiJojZrMoWaDydV8Lm
8TtYmwVhjhO4D/LTvBGjQerADEO9isMCtkkthkbdZ8rN
hytMnHf0bBTzeos1gs2CQt1jwLN0kePZ52aD1sJ1IuqH
/AEqH/SbMEwjuAFh0WM5IrAhwGP3VD/pBXGSiLxh7s0H
k6r4XN4naMidCH/RrYaZy+UQRoUzvF0KASugyJreTzsA
WRet/ga2GBWySWwyNus+VhrbD3S0FXayqSLDIITrAAyF
ggRndSf8ALNYLNgkLdY8CwRkMgZFV4q8VeIgDkI6rdnn
ZoPXmaDydV8Lm8TtFMx1BRQIojDHhbwDhiTILEizLKww
K2SS2GRt1ny4myqclXBheaQwFmsFmwSFuseByRbPOzQe
vM0Hk6r4XN4nwMOHjlOokgB4ZpAHIEZJqIJkshYEcL7W
CHj9Bv6NpgVsklsMjaJ6CcBBiVRyo5Ucgb9eFHsthu7l
BTgU8IMzM26wWbBIW6x4HJFs87NB68zQeTqvhc3ifFo6
FlETqL8JgVsklsMjawkmEgmEgmEhxawWbBIWgZguhZV4

q8VeKvFXiJgxKOq3Z52aD15mg8nVfC5vE+YYFbJJbzIp
jIpjIpjIpjIpjIpjIpjIpjIpjIpjIpjIpjIpjIoTAMRZsEgmTJ
kyZMmTJkyZbPOzQeqYyTGSYyTGSYyTGSYyTGSYyTGSYy
Q9jydV8NjJDsmSHZMkOyZIdkyQ7Jkh2TJDsmSHZMkOyZ
IdkyQ7Jkh2tYSTCSYSTCSYSTCSYSTCSYSTCSYSTCVjJh
IJhIJhIJhIJhIJhIJhIJhIJhIJhIJhIJhIJhIJhIWsJBMJBMJB
MJBMJBMJBMJBMJBMJBMJBMJBMJcnVfsmar9kzVfmnyN8
ITVeCrwVeCrwVeCrwVeCrwVeCrwVeCrwVeCrwVeCrwVe
CrwVeCrwVeCrwVeCrwVeCrwVeCrwVeCrwVeCrwVeCrwV
eCrwVeCrwVeCrwVeCrwVeCrwVeCrwVeCrwVeCrwVeCrw
VeCrwVeCrwVeCrwVeCrwVeCrwVeCrwVeCrwVeCrwVeCr
wTH2DkiX5sJ4H0P8Af0d8NcR7N7/NKzvpUdH/AEcorF9
Yjr4/OKQ3gSyn+ih9zcAYkoMjMn5UMCE/gLArqVgUdAn
NnnAzCYouwBcf9/QwssnYdU0j4Fc+XRTZ1CDitEdpl6h
Avnco/iDMiBTQUcmlhxG4hYgi4o8XHroQjGZISGPdKul
XSrpV0q6VdKulXSrpV0q6VdKulXSrpV0q6VdKulXSrpV
0q6VdKulXSrpV0q6VdKulXSrpV0q6VdKulXSrpV0q6Vd
KulXSrpV0q6VdKulXSrpV0q6VdKulXSrpV0q6VdKukXu
Wf9EcQABuvTV2VjxEb9DOwOJEABHB/BIcMixuyX6iTwp
4K/B2H4d7kAZi5fyQUMDoKgY+v0mJroF78LCA6FMBAAA
w/Iiq6hYFHS5EwI/aP+gI4Z0Q4nQoh9CMcNGIiD8wDQL
9IGQD4R1KBohRP4BBRamTqVBz9fDE0e0jHdlEuFE+Mfa
OR+0cNeY7IlgijWbm96yu5ZdNohZuxKTw4E6VGVOVIVI
WY7LMdlmOyzHZUhUhU5Ubhgo2KDYGfW8QsumSjN77LoS
BYgjJkGIkTT+0A4igHhQBDspqHYQAEB/3/wD/2gAMAwE
AAgADAAAAEPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPP

gBjXAEDXQAJDYAJDQgBjQAww1vPPPPPPPPPPPPPAAww6
AKQ1QFg1wAgwwAgQwQAwwQKwwwKgww1vPPPPPPPPPPPP
PAAww8waA1gVg0wQAw4VQwyASwyAag0AYgww1vPPPPPP
PPPPPPPAAwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww1v
PPPPPPPPPPPPPAAww6QAAAAGqQAAAAKssMOOMKwJCADA
CQww1vPPPPPPPPPPPPPAAwww0CAJJPQVDDAAKaBIIEAK
K1KEPIKAww1vPPPPPPPPPPPPPAAwwwwKAHNON5MJKAJ9
PBBNLComPHIAKAww1vPPPPPPPPPPPPPAAwwwwKAOEFre
JCJHBO0PCFPFVsIFDAKAww1vPPPPPPPPPPPPPAAwwwwK
ABIP0ZDJHIGSgLMIMCq1KFKAKAww1vPPPPPPPPPPPPPA
Awww1ODHCPQVPAPIKaAKABAKK8IAMIKAww1vPPPPPPPP
PPPPPAAwww0GNPAPQVPOJIKaAPPKAKKwAAAAKAww1vPP
PPPPPPPPPPPAAwwxwAEAANRXPPDDK7DDDHDIJTDDDDIA
ww1vPPPPPPPPPPPPPAAww4QQQQQQcccccYYQQQQQQYQ
QQQQYQww1vPPPPPPPPPPPPDzzzzzzzzzzzzzzzzzzzz
zzzzzzzzzzzzz3PPPPPPPPPPPPPPAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAPPPPPPPPPPPPOOHAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAABPMOPPPPPPPPPNHPPPK
AAAAAAAAAAAAAAAAAAAAAAAAAAAAANPPPLHPPPPPPPP
PDLOOPOMPPPPPPPPPPPPPPPPPPPPPPPPPOOMNNDHHPP
PPPPPPPPPPPPPDHPDDDLHPLHPDDDDPDHPPDHHPDHLPPP
PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP/EACo
RAAEDAgMHBQEBAAAAAAAAAEAEWEgMRBxoSFAQbHB4fA
wUFGB8YCR/9oACAEDAQE/EP4zAJLBAAf/AAgOwCiUSiU
SiUSiUSiUSiUSiUSiUSiUSiUSiUSiUSiUSiUSiUSJbgI
E+yUQuJ649pfXsYSzceuLVBGMLqFHAIwM2UKEDGBzhwU
KLANjCoUbBwI2WUKMA8GWChRyEioHBG6i/wC2F3Dn4WP
OFGs5jA2xs5Ya7Cx5xw5OF3OrSHcxYDryfsv8N93T/wA
LMP8AS8n7I8M3FP8Awvluy8n7I8Izp/4ToW1eT9l4P2T
/AMJ/4X0kryfsn3Gfan/hfHdv1eT9l8pk/wDCE1/peT9
ltWxmo0h3MWMsOlQOThe+ueGqw0lN3PDTYeDLDn4WcsO
lRpDuYEgKkCcPlgEBJAn+OBCApAh8iNmDwFIFIMSyLhS

BHGXDA0AqQI77AKIKkCcPlRpD7kNIdzBBsngplMplMpk
YuFGBsVMplMplMiJ2oQBMVMplMplMjEOQjBUymUymUyM
k3+aNIdzHLrF6rpK5VYnkyqu50aQ7mOXUe2t9JXKrK0u
50aQ7mOXVcV6suVWGsXc6NIdzHLrF6rpK5VYnkyqu50a
Q7mOXWL1XSVyqxPJlVdzo0h3McusXqukrlVieTKq7nRp
DuY5dYvVdJXKrE8mVV3OjSHdAyZMmTekyZMmTeppD64K
R2+xySSSSSSSSSSSSSSSEmAeH9+f/8QAKhEAAQIDCAM
BAQADAQAAAAAAQARYZGxECAhMaHB0fBBUXEwgVCA8eH
/2gAIAQIBAT8Q/wBMyIjAI4wx78n567kiTmP9KjZlRsy
o2ZUbMqNmVGzKjZlRsyo2ZUbMqNmVGzKjZlRsyo2ZUbM
qNmVGzKjZlRsyo2ZUbMqNmVGzKjZlRsyo2ZUbMqNmVGz
KIuQf0o2PqeRz3FApHB/cwYueJ2G8v8G8L0QP/o/cziF
BexjDBlF+FHaIqYuWxsdGwZ8P4o7REykhvXkgWCJFgTK
O0QyHgCfHgWBR2ijtEED4JFhFmCAPhR2iKk4DagGx2MF
yMFHaInYuHx+3iFj0r++u2F7J082a7azP+qiyios1os1
ylgzt1prZp1LKaiyo2s7UTe0Cv767YXDlyRbJ9wndOUD
lOwab5pnThHKDyPm/iCd05QzMEzZ+Xx9ekzpwiOBHO7N
jkw9e07pygf2LMztFzRM6cJni+Z8nwfJO6cpwx6TTOnC
Z04T/AAczNk+PtO6cpn9ydnbyzGqZ04RyUOzs2Gbj16T
unKGd9ru2cMfXtM6cI5M+Bs4w9p3TlfRnhsmifdzQK/v
rthc60BZm6eLNNvZQb2VVDZoTZplbDlbpRSzWq2V1TZS
b2dKAs6P5c0Cv767YXDoCSXwb0IrpDlYfEYs4txYBIlw
2C6Q5TywLg4tGNg83AfKIIXSHKOMcG8c2ANZAgyK6Q5R
/4XNpY4QA8crpDlZaxE2GzEh8m8kn2ukOU1sgMBjYdoX
L4N6+rpDlY/Bizby3FzQK/vrth/g9Ar++u2FwyIDi8QC
hZBQsgoWQULIKFkE38DE5fxBBAcT4UFIKCkFBSCgpBQU
gibTf8IxgJx8RKhZBQsgoWQULIKFkE1hsCiAAnHxEqBk
FAyCgZBQMgoGQQFBsNzc0Cv767YXM36oL2pOyyfpvZun
hV9Te0p2VVU3u1E3NAr++u2FzN+qC8Av+zsmAG9m8Gfr
JV9TeHGgULf0qb3aibmgV/fXbC5m/VBe1B2WT9N4HAiO
yFg/am9oTVG5vtTe7UTc0Cv767YXM36oL2pOyyfpvEQe
nhESUxqbxliB2VVU3u1E3NAr++u2FzN+qC9qTssn6b2b
p4VfU3tKdlVVN7tRNzQK/vrthczfqgvak7LJ+m9m6eFX
1N7SnZVVTe7UTc0Cv767YXM36oL2pOyyfpTp06dOs3Tw
q2pvaU7KqqU6dOnTrpRNzQK/vrthcdOU5TlOU5tdOnTp
7HTlOU5TlObHTlOU5TlObugV/d4ACYevqiZjlRMxyomY
5UTMcqJmOVEzHKiZjlRMxyomY5UTMcqJmOVEzHKiZjlR

MxyomY5UTMcqJmOVEzHKiZjlRMxyomY5UTMcqJmOVEzH
KiZjlRMxyomY5UTMcqJmOVEzHKiZjlRMxyomY5UTMco6
hgR69/f8Afz//xAAtEAABAgMGBgMBAQEBAQAAAAABABE
xUfAhQWGRwfEQIDBxgaFAsdFg4VCQoP/aAAgBAQABPxD
/AOMAiABaSSwCLFyms/0j9gjIWRL+lbtW7Vu1btW7Vu1
btW7Vu1btW7Vu1btW7Vu1btW7Vu1btW7Vu1btW7Vu1bt
W7Vu1btW7Vu1btW7Vu1btW7Vu1btW7Vu1btW7Vu1btW7
Vu1btW7Vu1btW7Vu1btW7Vu1btW7Vu1btW7Vu1btRCzC
A6txY6bEGDzh/wG0we1bkybHWZ/g7Jg3i491DAPg/z55
SltGxGf8AC3QURMSRd3A4+ac0m99/n8OYo0BjWPzbfuF
1CQASSwEeUoIIIfYhdwBgQeSwzgY2DAseQIIIIQxgMFp
B25Bk5gzg5gHPIEEETDjKJJAchIAcwHKEEEEPswu4CYP
ISBwBjYMCx5AggrqQslpB27Hktd4GcHMA55AggnwuHCU
gOq1rhJZPp82nx6ntfpGPJTZOSu48lGlyKpLkev8Aoq7
j7j65aNLkqcjyQd3IxaY8lfxV3UpMPm02PU9r9Ix5KbJ
yVXHko0uRRJcj1/0Vdx9x9ctGlyVOR5IO7kYtMeSv4q7
qUmHzabHqe1+kY8lNk5KrjyUaXIokuR6/6Ku4+4+uWjS
5KnI8kHdyMWmPJX8Vd1KTD5tNj1Pa/SJDxTiacTTiaps
nJTcU4mnE04mqNLkVSSOJpxNOJo8v9FCHH3H0nE04mnE
04mqNLkqcinE04mnE1aHdyNg6YpxNOJpxNHXzV3UpMPm
02PQMgoJBfaO4Rd1q3at2rDlvzQZ0q2YtmLZiFRgAAEA
E3at2rdqEgjYOYia2YtmLZixRfvrZFu1btW7UQ8L0xOI
zLZi2YtmIWMGxAAItC3at2rdqwfL81s1bMWzFsxCgwQA
IAWrdq3at2oTiNg5iJrZi2YtmLFUOvvFbtW7Vu1CJEJJ
xmWzFsxbMQuIZIAELdq3at2oOidOvQHQpMPm02PQ9v9h
GPGtTQh0LqDMclWnyKZPke5+wr+Ppvvlqk+SnzHJ6PIw
eVfv4iI7r0H10KTD5tNj0Pb/YRjxrU0IdC6gzHJVp8im
T5HufsK/j6b75apPkp8xyejyMHlX7+IiO69B9dCkw+bT
Y9D2/2EQXgmMkxkhNVFCHIKcxkmMkxkqDMclSmjGSYyT
GSrk+R7n7CYvBMZJjJAfG++UaS9MZJjJMZKnzHJ6KWMk
xkmMlB5V9i8ExkmMkAXHdegProUmHzabHoXb1QBIrbT+
rbT+rbT+oEAgZBcWH95TzJCi5Ik2rbT+rbT+rbT+pvkl
cXMTyF0AkGBii1uAW2n9W2n9W2n9RYZybl3REcByXbly
BIrbT+rbT+rbT+oCIIfFsD+8pp1hRcjEm1baf1baf1ba
f1DoNK4uYnkvOWwMTO1uAW2n9W2n9W2n9VszTMuxiI8h
LLoUBuK20/q20/q20/q20/qAYMIdCkw+bTY/w9Jh82mx
+OxkUxkUxkUxkUxkUxkUxkUxkUxkUxkUxkUxkfg+F45/

C8Lx0qTD5tNj0AlKSYHF7Qq71Vd6qu9VXeqrvVV3qq71
Vd6qu9VXeqrvVO1ftHDFALkkg5R2+QwyITi5VDqqh1VQ
6qodVUOqqHVVDqqh1VQ6qodVUOqDZOCLjF78BChzPZBj
tFVzqq51Vc6qudVXOqrnVVzqq51Vc6qudVXOqJOXDAEX
BtRie6Fo5lsUWOxtVY6oRFkhm90XDzd73veVkyDJHEil
cfVY6qsdUGroaAkQTb0aTD5tNj0ABAQCL3tWzLZlsy2Z
bMtmWzLZlsy2ZbMsCRuVQkQgSAQ2KtmWzLZlsy2ZbMtm
WzLZlsy2Zeq1iODxAGeEbMtmWzLZlsyMhhXsLU9d4hRg
YKLAStmWzLZkLl1hRie6aMM4Ypsy9AjEKitFRWiIwYcN
hA9ooPBxC9l26qhREjWBiIXqitEZcBYeYGNyNbKsiJA/
p1syDKBGSruhSYfNpsehWZeIrhkkcAEg2EvEHhiGJEwb
DjxFojiGIUM7OAJDEDeOGIGgEK7IEgMRMAeHoFVCReh4
wB/kFAHIgQMQeOLERrReKAOWIZiomFGIdwbeSjTHLsdL
ggKRwI2qrdEQAk2NMEywMxiCGwV5xRjzuT8b3DxZkRf0
Hsc5LMAONelRie6r0+CgTHH3+HP8AAcAWdAjAglBGCCW
w8qUfZ4UzFXdCkw+bTY9Csy8L+IGvy5b8BMsHDdxGltU
C58V6BVQkXoeEI8rBkZWLYjOy2jNTM6JIM+r2Bjwo0xy
7BuCZIDrADguiAQG8MgJsdWggRhr2RAAhi7GCEgpsJJC
GPCvSoxPdV6fBQJjj7/DtimR4EnwAT4QRghmgIGBF7Rw
KZiruhSYfNpsehWZeF/EDX5cl+bLkInAJtkMYaq9JegU
NkPEIFxG9xZsU7zIAVpfYLogkXEDEcAJRCRE5ox7oywP
2InuSWbvdxweq4GQN7zEe4T3EFO7Fn8hj5TDtuEgHTnI
CaC0RF/HYUIOyJuHlHaQA5YLBTpDisKwgGWB7RG9AjqL
OFb5KtJsvDw9pyNndxDnFAYDWdtRQLYYFWt3RVZkSYLK
zNnGHtN+YIXEKMT3VsYqKAhMGDwE6MBYMREX8bo+Vzo7
BrkHLWu34Ra9isNGLXNpmbpW8DpB2R2wdywTyx5oi5OZ
4UzFXdCkw+bTY9Csy8L+IGvy4ydk4BGmbEIgPMzId+0g
jgjEACD4or0kIt2MJvL8QDwSRBysW4ILAgg8SAROGXrO
b3Mgrdh7UEyCOckngE5CYDAsTfLFGFLIjoC1ll5RJykY
E4exRC+YTFFiyBIEDcU8IpAVYEAzOHDgC6yaM5ggKGIN
oIvdkW5wGQADkt9BE5AiQbMQ9lxTEiFyTIwqJAMkIcAJ
yMC0SVaq4EK2a9yt7BIBJAsPijnwRAReWAogFYCEAaZ4
oMEA5AQQtFZnawY4koCUgoBYQJ4hOIxiwHPgAhLVZIg9
mwiaNjx2NhwQWBzwS1xQYsG7Th3NyJ2mICLGcOfCIkHo
co0dOwDsJnAWp8YizQTHaAutvPGmYq7oUmHzabHoVmXh
fxA1+XHco/ipmQ3ZEAUwIYBFk0xAyRVUGVNtYKnd4Mok
EgIAkDHDuisMhICL9EF5l5nZ6RjhiC4R8tjF7H6g0S4Y

fsFH2FJMCICQYkh7BXv8A3XoPoL6f0qzJPrP3w9yu7RF
HKydYY6AZWlmToIvcP0Vn+B/3/wBKsyI1u6buEgN0ehx
Y/wCET9yQCvtFgw4nzMZgAYvIhAbHz8kSHFpe0W9xwFd
Gg5ZJAHcP5L1340zFXdCkw+bTY9Csy8L+IGvy4xa5Whk
zJTix2BixY3YyQtYQzC5r83BC6OCC8RVUeVEAFiDBR2O
Aubsb3wKeOTDdYqJ1EHIKC4mOSLidxR1EQAseReYEMhH
AQiDFoiFpYCWQAADuz4r6z9IVJ/cA4SRPZkaMmgiRLkn
yvf8AuvQfQTiBYSQfwh3MLmCWFrBOCMgYgO5PngFvB7S
RDadi1gIusEfoQSSwB2ACI3xKNxzFViu1hIDtxSigJrg
eMw2m+PCvv/pVmRGt3QQJYcQtcEaaDiRie4Q6MpgQIIY
jgA71mnLgxhh2Xvrm7NtwyfBbwQLh+ALjTiwOOQ7wRxp
mKu6FJh82mx6FZl4X8QNflyX5mAJRCsRaAIFnDbkYLQQ
4KIkguSJnYXoVPgmBQMbSpkU4hIYAgAAgEFyQXGLXoBc
HBgAhdeR2FvlMGci8EXIG4y9MCJvGF7xJCxmY0EFOGDA
AMAhfLktQdxaiHckkkl3xR8G9rY8COCiRJPAHGU5uggJ
G1AAQSJCD3cJXXGwkhEBVu47ABkDjAgUeYEQABgEPIeZ
0IVa4S5JRscR1kAAABBoo6nLoAHsIYMbGvQAXJe+xe+v
GwwUzgkSQ9jD3fgJ9TkWiTsDJuJTMVd0KTD5tNj0KzLw
v4ga/LnnzfoFVCReh4QjxYDxpQGICx3BmVuf4TCFraLz
ABB7uFjwYQDj74NTAYQCRAjwJbHhRpjn2VzFxQAuIbD2
HIsttYLFoEsBYLoca9KjE91Xp8FAmOPv8O7NRrcA4T8+
wcAEzBUAIYhDKNoJijys7g8KZiruhSYfNpsehWZeIBUT
N4/TgIYLkPZI4jwoAqmBqFwMqDDs5++HoFVCReh4QjxY
UGfCIolES4A8gIem4CiRkdgTXhRpjrbK5uvSoxPdV6fA
MjAJeC/GrVsZvUADo7EiI4HliwLguDmFYstsIsLWeB+F
UMoqQR4CbQch4UzFXdCkw+bTY9Csy8QNmAjEjY5cAFK1
PDxBdrjgShJoSAJYkAYhHRMAQHvJJj2FpTl5BAoleWJL
nzwC4IARcQAA5nB6BVQkXoeEI8WAdaMdoi12hAqu9E8g
waOvL0mUBMrAJ2gGHBy8XMJRbIM+FGmOfZXM43mMT41U
WiqLRVFoioAWkkAHrjXpUYnuq9PqKZiruhSYfNpsehWZ
eJCDoGMMGMCMQEBP0jYBqHLajGYcFMJNiOAHoAbAgAwA
AHjh6BVQkXoeEI87AQgyd9CZJsAvJQ2XUHu1Z4AAOFGm
Onsr0SSMeFelRie6r0+opmKu6FJh82mx6FZl5C6CgzG8
t70faAoctKBmVth8EoLNUCThEQfiEHtR+BIPOMIefCEG
PL4977CfDcfQKqEi9DwhFNmGlE4Eqr9VV+qq/VFoUA+7
VkE7RtYtAe1bMAEvTPYYXAcaNMdPZXokkY8K9KjE91Xp
9RTMVd0KTD5tNj0KzLym0MbQgjCpHD0V7Sr7cvoFVCRe

h4xJED4WwLYFsCAAgAPHLRpjl2NocWOHoqotVUWqqLVV
Fqqi1REQLCCYH3xr0qMT3Ven1FMxV3QpMPm02PQpMqYy
OSYyOSYyOSYyOSYyOSYyOSYyOSYyOSYyOSYyOSYyOSAw
DA3KoSISWgTZgraFtC2hbQtoW0LaFtC2hbQtoXpJYjgJ
LQWOkck6RyTpHJOkck6RyTpHJOkck6RyTpHJOkck6RyQ
GGe1hRie6ElkE4WKYjJYjJYjJYjJYjJYjJYjJYjJYjJY
jJYjJGhnsd1d0KTD5tNj0CAQxAIxWzFsxbMWzFsxbMWz
FsxbMWzFsxbM4EAxAKwmSwmSwmSwmSwmSwmSwmSwmSwm
SwmSwmSwGXAgYgHuFti2xbYtsW2LbFti2xbYtsW2LaOB
AMQD3C2hbQtoW0LaFtC2hbQtoW0LaEAQDLo0mHzabH+H
pMPm02P8AD0mHzT+UAEADg1qrfVVvqq31Vb6qt9VW+qr
fVVvqq31Vb6qt9VW+qrfVVvqq31Vb6qt9VW+qrfVVvqq
31Vb6qt9VW+qrfVVvqq31Vb6qt9VW+qrfVVvqq31Vb6q
t9VW+qrfVVvqq31Vb6qt9VW+qrfVVvqq31Vb6qt9VW+q
rfVVvqq31Vb6qt9VW+qrfVVvqq31Vb6qt9VW+qrfVVvq
q31Vb6qt9VW+qrfVVvqq31Vb6pqlGfAtG1uyu+YcAkxJ
ZA/hxLVYbMYew+aAX6BgWm0wdy9F2h/DN2WCm8jn2sDy
+aQCCCAQbij2GwCw4kxF2DSP8LaCq2FjyQ9lhemleB7X
nElycT8qP5J3OQVjm+zPaN9wH8UEBw9UO3Zw9q6kEJ/4
z1iZYMv4ONDa4MyuCOSARGMCQS8qEmGEVZnbyS1VjYsw
EdAFK629oEA4IIN4+IDgbDAHCaKFyVwtkHMI7rPwMQix
SW23gMR7RGDLoByt9JsERTBUDoqB0VA6KgdFQOioHRUD
oqB0VA6KgdFQOioHRUDoqB0VA6KgdFQOioHRUDoqB0VA
6KgdFQOioHRUDoqB0VA6KgdFQOioHRUDoqB0VA6KgdFQ
OioHRUDoqB0VA6KgdFQOioHRUDoqB0VA6KgdFQOioHRU
DoqB0VA6KgdFQOioHRUDoqB0VA6KgdFQOioHRAxYOU3I
eAY3IDMgr6CrUnuVg8AoZD9pERTJiT35nNhwNmUEGA8h
YjuPxAwBOCLx8EBCgQyMTwwA9mMuk5mc05mc05mc05mc
05mc05mc05mc05mc05mc05mc05mc05mc05mc05mc05mc
05mc05mc05mc05mc05mc05mc05mc05mc05mc05mc05mc
05mc05mc05mc05mc05mc05mc05mc05mc05mc05mc05mc
05mc05mc05mc05mc05mc05mc05mc05mc05mc05mc05mc
05mc05mc05mc05mc05mc05mc05mc05mc05mc05mc05mc
05mc05mc05mc+kNg5vHYHlAHQJ5n4ZslHFekK0kHcewr
XBVxZZ75Nv4k03eQqSEwBGW4I57KaCTFeT+oICAABAAQ
+R6R0p1cg3mfonR9wA/YQ7ymRQb+WcaKEdg6qOk8D9L3
UEjADuGTiY/6riYQBMAT2C9fBo/Z5Q32o32ihT4M5Oiu

```
QMP1K9AJA0UycYqLJMCgAAw/4hAIYh1EF3FRw+KiQ+zj
6UE7A0RA9j/KLg+7HRG6D3FXI8/6Rucyjdk7uERwP5fi
IYF7f5RnvgiGft+yIdMj9RF/j+ogUI/2IjjmkRxzCII5
hM/vW8Vu1bsW+lvJbyW8luxbtW8VutAkM0gaGcQLDPIH
/AEIGXBfxCBv3H6geXv8AssO8EDRH3QFiPz/EDRHmdEL
zNLRj/SF/4Q/UDEdmGiDj3wjRRDuiUCN3JOq0U0IY9hQ
BgDsP/f8A/9k=",
    CURRENCY: "RUB",
    SORT: "500",
    ACTIVE: "Y",
    CONFIG: [
        {
            CODE: "SETTING_1",
            VALUE: "SETTING_1 value",
        },
        {
            CODE: "SETTING_2",
            VALUE: "SETTING_2 value",
        },
    ],
}
```


Ответ JSON:


```
{
    result: {
        parent: {
            ID: "622",
            PARENT_ID: null,
            NAME: "Uber Taxi",
            ACTIVE: "Y",
            DESCRIPTION: "Uber Taxi Description",
            SORT: "500",
            CURRENCY: "RUB",
            LOGOTYPE: "954",
        },
        profiles: [
            {
                ID: "688",
                PARENT_ID: "622",
                NAME: "Taxi",
```

```
                ACTIVE: "Y",
                DESCRIPTION: "Taxi Delivery",
                SORT: "500",
                CURRENCY: "RUB",
                LOGOTYPE: null,
            },
            {
                ID: "689",
                PARENT_ID: "622",
                NAME: "Cargo",
                ACTIVE: "Y",
                DESCRIPTION: "Cargo Delivery",
                SORT: "500",
                CURRENCY: "RUB",
                LOGOTYPE: null,
            },
        ],
    },
    time: {
        start: 1642404734.307061,
        finish: 1642404734.582061,
        duration: 0.27500009536743164,
        processing: 0.08100008964538574,
        date_start: "2022-01-17T09:32:14+02:00",
        date_finish: "2022-01-17T09:32:14+02:00",
    },
}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.delivery.config.get

Метод позволяет получить информацию о службе доставки по её ID.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.config.get
```

Вызов JSON:

```
{
    ID: 622
}
```

Ответ JSON:

```
{
    result: [
        {
            CODE: "SETTING_1",
            VALUE: "SETTING_1 value",
        },
        {
            CODE: "SETTING_2",
            VALUE: "SETTING_2 value",
        },
    ],
```

```
    time: {
        start: 1638543677.049672,
        finish: 1638543677.459672,
        duration: 0.4100000858306885,
        processing: 0.023000001907348633,
        date_start: "2021-12-03T17:01:17+02:00",
        date_finish: "2021-12-03T17:01:17+02:00",
    },
}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.delivery.config.update

Метод для обновления службы доставки.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.config.update
```

Вызов JSON:

```
{
    ID: 687,
    CONFIG: [
        {
            CODE: "SETTING_1",
            VALUE: "SETTING_1 new value",
        },
        {
            CODE: "SETTING_2",
            VALUE: "SETTING_2 new value",
        },
    ],
}
```

Ответ JSON:

```
{
    result: true,
    time: {
        start: 1638369486.01651,
        finish: 1638369486.41351,
        duration: 0.3970000743865967,
        processing: 0.04800009727478027,
        date_start: "2021-12-01T16:38:06+02:00",
        date_finish: "2021-12-01T16:38:06+02:00",
    },
}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.delivery.delete

Метод удаляет службу доставки.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.delete
```

Вызов JSON:

```
{
    ID: 638
}
```

Ответ JSON:

```
{
    result: true,
    time: {
        start: 1638372339.03651,
        finish: 1638372339.48151,
        duration: 0.44499993324279785,
        processing: 0.09599995613098145,
        date_start: "2021-12-01T17:25:39+02:00",
        date_finish: "2021-12-01T17:25:39+02:00",
    },
}
```

sale.delivery.getList

Метод для получения списка доступных служб доставки.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.getList
```

Вызов JSON:

```
{
    FILTER: {
        PARENT_ID: 622
    }
}
```

Ответ JSON:

```
{
    result: [
        {
            ID: "688",
            PARENT_ID: "687",
            NAME: "Taxi",
            ACTIVE: "Y",
            DESCRIPTION: "Taxi Delivery",
            SORT: "500",
            CURRENCY: "RUB",
```

```
            LOGOTYPE: null,
        },
        {
            ID: "689",
            PARENT_ID: "687",
            NAME: "Cargo",
            ACTIVE: "Y",
            DESCRIPTION: "Cargo Delivery",
            SORT: "500",
            CURRENCY: "RUB",
            LOGOTYPE: null,
        },
    ],
    time: {
        start: 1638544721.243672,
        finish: 1638544721.621672,
        duration: 0.37800002098083496,
        processing: 0.019999980926513672,
        date_start: "2021-12-03T17:18:41+02:00",
        date_finish: "2021-12-03T17:18:41+02:00",
    },
}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.delivery.update

Метод для редактирования службы доставки.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.update
```

Вызов JSON:


```
{
    ID: 638,
    NAME: "Taxi (updated)",
    DESCRIPTION: "Taxi Delivery (updated)",
    CURRENCY: "USD",
    SORT: "100",
    ACTIVE: "N",
    LOGOTYPE:
"iVBORw0KGgoAAAANSUhEUgAAACMAAAAeCAQAAAAMNG+
hAAAABGdBTUEAALGPC/xhBQAAACBjSFJNAAB6JgAAgIQ
AAPoAAACA6AAAdTAAAOpgAAA6mAAAF3CculE8AAAAAmJ
LR0QA/4ePzL8AAAAHdElNRQflBRwFIxtqdTugAAACgkl
EQVRIx63VzWtcZRTH8c+9M6YztZS+ZhqSoMRISVukWMW
Ai2oQrEJB6E5Fu2tx1UU3ghtxJ/QPKG6M21pQF4IEIrF
JIylUyEIsSosmaUibaRMzzbxmHhcdhsyMTifE37N67j3
ne849557noVE7XLKiqKysKK+kJK+gJK+koKhoxSU7Gt2
```

iJsyACb1+dA89hlzDiPtmDZvVrazgNQtOuq2Nhiwpu2C
/Az42qkuXr11zwBkZX3nPiKIlQ41ucRMmIOkzJ7ziEz/
r12vCy064qsebNlTrdpuUbMknICUhoeCGNcENSx5Jyck
JNUBon01Uq9ZBGYsqTjut4gc5F6VM6ZYRScg0F3mz+o0
qC4JlWSVzysrmLVtQctdDD9wTBH+56sV/79ROX3jXn+7
YECOIhLpNqNtWJQzqM+OM+dZcTskbd1Ra6gkr7agxVRd
aIbHLcl7XqU5a831rhQ65ZcLTHWN2mzHn2eZOvWTAuEc
dY9bcdMiRx5uklFcdMeUNFRMdQwhmnDMs1usmI5YF037
3iz1bwHDcQ7+5K5hLumXa24bxreea/8622iPrMPK+i5D
xuffF8opbyoZdku741JUknnFMhLT0FjGwbtI6fSaFba0
r9iZ85IOWAd2aBi0wvc1cguCnSNY+zBq1X58ePQal2kY
v+MOiRfOyPvQCslQEwZguRFK6nbfaJvKq87qlRHjKmCC
oqL98qx4t3bbok5v6ecrK46eRoi5w27icoGq3dxz8z0+
67xt/i0V2GTEAKpEH9m6rT5CLrW4bwnps/X/A5JINp3F
ZtsPhjOyr1RSKyQa3X53t8ODa6UvH67uQdF1/7dKrum6
2dis+SbEpz9eGaMP0P8TcLFFxjK4UAAAAJXRFWHRkYXR
lOmNyZWF0ZQAyMDIxLTA1LTI4VDA1OjM1OjI3LTA0OjA
wPkzT7wAAACV0RVh0ZGF0ZTptb2RpZnkAMjAyMS0wNS0
yOFQwNTozNToyNy0wNDowME8Ra1MAAAAASUVORK5CYII
=",
}

Ответ JSON:


```
{
    result: true,
    time: {
        start: 1638372119.32451,
        finish: 1638372119.74951,
        duration: 0.4249999523162842,
        processing: 0.0840001106262207,
        date_start: "2021-12-01T17:21:59+02:00",
        date_finish: "2021-12-01T17:21:59+02:00",
    },
}
```

sale.delivery.extra.service.add

Метод для добавления дополнительной услуги службы доставки.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.extra.service.add
```

- **Для типа checkbox**

  Вызов JSON:

```
{
    DELIVERY_ID: 623,
    ACTIVE: "Y",
    CODE: "door_delivery",
    NAME: "Door Delivery",
    TYPE: "checkbox",
    PRICE: 59.99,
    SORT: 100,
}
```

  Ответ JSON:

```
{
    "result": 988,
    "time": {
```

```
        "start": 1638367387.32351,
        "finish": 1638367387.72551,
        "duration": 0.40199995040893555,
        "processing": 0.051999807357788086,
        "date_start": "2021-12-01T16:03:07+02:00",
        "date_finish": "2021-12-01T16:03:07+02:00"
    }
}
```


- **Для типа enum**

  Вызов JSON:

```
{
    DELIVERY_ID: 689,
    ACTIVE: "Y",
    CODE: "cargo_type",
    NAME: "Cargo Type",
    TYPE: "enum",
    ITEMS: [
        {
            TITLE: "Small Package(s)",
            CODE: "small_package",
            PRICE: 129.99,
        },
        {
            TITLE: "Documents",
            CODE: "documents",
            PRICE: 69.99,
        },
    ],
    SORT: 100,
}
```

  Ответ JSON:


```
{
    "result": 989,
    "time": {
        "start": 1638367586.86651,
        "finish": 1638367587.27851,
```

```
        "duration": 0.4120001792907715,
        "processing": 0.057000160217285156,
        "date_start": "2021-12-01T16:06:26+02:00",
        "date_finish": "2021-12-01T16:06:27+02:00"
    }
}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.delivery.extra.service.dele

Метод удаляет дополнительную услугу службы доставки.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.extra.service.del
ete
```

Вызов JSON:

```
{
    ID: 999
}
```

Ответ JSON:

```
{
    result: true,
    time: {
        start: 1638368327.37051,
        finish: 1638368327.78751,
        duration: 0.41699981689453125,
        processing: 0.03900003433227539,
        date_start: "2021-12-01T16:18:47+02:00",
        date_finish: "2021-12-01T16:18:47+02:00",
```

```
    },
}
```

sale.delivery.extra.service.get

Метод позволяет получить дополнительную услугу службы доставки по ID этой услуги.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.extra.service.get
```

Вызов JSON:

```
{
    DELIVERY_ID: 623
}
```

Ответ JSON:

```
{
    result: [
        {
            ID: "999",
            CODE: "cargo_type",
            NAME: "Cargo Type",
            DESCRIPTION: "",
            ACTIVE: "Y",
            SORT: "100",
            TYPE: "enum",
```

```
            ITEMS: [
                {
                    TITLE: "Small Package(s)",
                    CODE: "small_package",
                    PRICE: 129.99,
                },
                {
                    TITLE: "Documents",
                    CODE: "documents",
                    PRICE: 69.99,
                },
            ],
        },
        {
            ID: "1000",
            CODE: "door_delivery",
            NAME: "Door Delivery",
            DESCRIPTION: "",
            ACTIVE: "Y",
            SORT: "100",
            TYPE: "checkbox",
            PRICE: 14.99,
        },
    ],
    time: {
        start: 1638788570.27462,
        finish: 1638788570.68062,
        duration: 0.40599989891052246,
        processing: 0.04699993133544922,
        date_start: "2021-12-06T13:02:50+02:00",
        date_finish: "2021-12-06T13:02:50+02:00",
    },
}
```

sale.delivery.extra.service.upd

Метод для обновления дополнительной услуги доставки.

JSON схема запроса

Пример

Запрос:

```
sale.delivery.extra.service.update
```

Вызов JSON:

```
{
    ID: 988,
    ACTIVE: "Y",
    CODE: "door_delivery",
    NAME: "Door Delivery",
    TYPE: "checkbox",
    PRICE: 69.5,
    SORT: 100,
}
```

Ответ JSON:

```
{
    result: true,
```

```
    time: {
        start: 1638368168.66651,
        finish: 1638368169.09951,
        duration: 0.432999849319458,
        processing: 0.06200003623962402,
        date_start: "2021-12-01T16:16:08+02:00",
        date_finish: "2021-12-01T16:16:09+02:00",
    },
}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.delivery.request.delete

Метод удаляет транспортную заявку.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.request.delete
```

Вызов JSON:

```
{
    DELIVERY_ID: 694,
    REQUEST_ID:
"4757aca4931a4f029f49c0db4374d13d",
}
```

Ответ JSON:

```
{
    result: true,
    time: {
        start: 1638794782.459819,
        finish: 1638794783.581819,
        duration: 1.121999979019165,
        processing: 0.6140000820159912,
        date_start: "2021-12-06T14:46:22+02:00",
```

```
        date_finish: "2021-12-06T14:46:23+02:00",
    },
}
```

sale.delivery.request.sendmess

Метод для отправки сообщений о доставке.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.request.sendmessa
ge
```

Вызов JSON:

```
{
    DELIVERY_ID: 694,
    REQUEST_ID:
"4757aca4931a4f029f49c0db4374d13d",
    ADDRESSEE: "MANAGER",
    MESSAGE: {
        SUBJECT: "Your order is on its way",
        BODY: "Estimated delivery price:
#MONEY#",
        MONEY_VALUES: [
            351.2
        ],
        STATUS: {
```

```
                MESSAGE: "Success",
                SEMANTIC: "success",
            },
        },
    }
```

Ответ JSON:


```
{
    result: true,
    time: {
        start: 1638795181.272819,
        finish: 1638795181.944819,
        duration: 0.6719999313354492,
        processing: 0.23099994659423828,
        date_start: "2021-12-06T14:53:01+02:00",
        date_finish: "2021-12-06T14:53:01+02:00",
    },
}
```

sale.delivery.request.update

Метод для обновления транспортной заявки.

JSON схема запроса

Пример

Запрос:

```
https://my.bitrix24.ru/rest/sale.delivery.request.update
```

Вызов JSON:

```
{
    DELIVERY_ID: 694,
    REQUEST_ID:
"4757aca4931a4f029f49c0db4374d13d",
    STATUS: {
        TEXT: "Performer found",
        SEMANTIC: "process",
    },
    PROPERTIES: [
        {
            NAME: "Car",
            VALUE: "Gray Skoda Octavia,
a777zn",
        },
        {
```

```
                NAME: "Driver",
                VALUE: "John Smith",
            },
            {
                NAME: "Phone Number",
                VALUE: "+11111111111",
                TAGS: [
                    "phone"
                ],
            },
            {
                NAME: "Something else",
                VALUE: "Some value",
            },
        ],
    }
```

Ответ JSON:

```
{
    result: true,
    time: {
        start: 1638795285.299819,
        finish: 1638795285.778819,
        duration: 0.4790000915527344,
        processing: 0.13300013542175293,
        date_start: "2021-12-06T14:54:45+02:00",
        date_finish: "2021-12-06T14:54:45+02:00",
    },
}
```

Расчет стоимости доставки

JSON схема запроса

Пример запроса JSON:

```
{
    SHIPMENT: {
        ID: 354,
        DELIVERY_SERVICE: {
            ID: 716,
            CONFIG: [
                {
                    CODE: "PROFILE_TYPE",
                    VALUE: "TAXI",
                }
            ],
            PARENT: {
                ID: 715,
                CONFIG: [
                    {
                        CODE: "SETTING_1",
                        VALUE: "SETTING_1
value",
                    },
                    {
                        CODE: "SETTING_2",
                        VALUE: "SETTING_2
value",
                    },
```

```
                ],
            },
        },
        PRICE: 99999.99,
        CURRENCY: "RUB",
        WEIGHT: 230,
        PROPERTY_VALUES: [
            {
                ID: 451,
                TYPE: "ADDRESS",
                VALUE: {
                    LATITUDE:
"51.507625620491",
                    LONGITUDE:
"-0.12546300888062",
                    FIELDS: {
                        POSTAL_CODE: "WC2N
5NS",
                        COUNTRY: "United
Kingdom",
                        ADM_LEVEL_1:
"England",
                        LOCALITY:
"Westminster",
                        STREET: "Craven
Street",
                        BUILDING: "10",
                        ADDRESS_LINE_1:
"Craven Street, 10",
                    },
                },
            },
            {
                ID: 452,
                TYPE: "ADDRESS",
                VALUE: {
                    LATITUDE:
```

"51.511995991646",
                    LONGITUDE:
"-0.13612747192383",
                    FIELDS: {
                        POSTAL_CODE: "W1F
9UH",
                        COUNTRY: "United
Kingdom",
                        ADM_LEVEL_1:
"England",
                        LOCALITY:
"Westminster",
                        STREET: "Great
Pulteney Street",
                        BUILDING: "33-34",
                        ADDRESS_LINE_1:
"Great Pulteney Street, 33-34",
                    },
                },
            },
            {
                ID: 453,
                TYPE: "STRING",
                VALUE: "some comment",
            },
        ],
        ITEMS: [
            {
                NAME: "Apple xs 128gb",
                PRICE: 99999.99,
                WEIGHT: 230,
                CURRENCY: "RUB",
                QUANTITY: 1,
                DIMENSIONS: {
                    WIDTH: "5",
                    HEIGHT: "5",
                    LENGTH: "20",

```
                    },
                }
            ],
            EXTRA_SERVICES_VALUES: [
                {
                    ID: 1034,
                    CODE: "cargo_type",
                    VALUE: "small_package",
                },
                {
                    ID: 1033,
                    CODE: "door_delivery",
                    VALUE: "Y",
                },
            ],
            RESPONSIBLE_CONTACT: null,
            RECIPIENT_CONTACT: null,
        }
    }
```

Пример ответа с успешным расчетом стоимости:

```
{
    SUCCESS: "Y",
    PRICE: 79.99,
}
```

Пример ответа с ошибкой при расчете стоимости:

```
{
    "SUCCESS": "N",
    "REASON": {
        "TEXT": "Delivery is not available for the
specified address"
    }
}
```

© «Битрикс», 2001-2008, «1С-Битрикс»

Создание заказа на доставку

JSON схема запроса

Пример запроса JSON:

```
{
    SHIPMENTS: [
        {
            ID: 354,
            DELIVERY_SERVICE: {
                ID: 723,
                CONFIG: [
                    {
                        CODE:
"PROFILE_TYPE",
                        VALUE: "TAXI",
                    }
                ],
                PARENT: {
                    ID: 722,
                    CONFIG: [
                        {
                            CODE:
"SETTING_1",
                            VALUE:
"SETTING_1 value",
                        },
```

```
                        {
                            CODE:
"SETTING_2",
                            VALUE:
"SETTING_2 value",
                        },
                    ],
                },
            },
            PRICE: 99999.99,
            CURRENCY: "RUB",
            WEIGHT: 230,
            PROPERTY_VALUES: [
                {
                    ID: 451,
                    TYPE: "ADDRESS",
                    VALUE: {
                        LATITUDE:
"51.505779",
                        LONGITUDE:
"-0.127673",
                        FIELDS: {
                            POSTAL_CODE:
"SW1A 2WH",
                            COUNTRY: "United
Kingdom",
                            ADM_LEVEL_1:
"England",
                            LOCALITY:
"Westminster",
                            STREET:
"Whitehall",
                            BUILDING: "26",
                            ADDRESS_LINE_1:
"Whitehall, 26",
                            ADDRESS_LINE_2:
"Ripley Buildings",
```

```
                        },
                    },
                },
                {
                    ID: 452,
                    TYPE: "ADDRESS",
                    VALUE: {
                        LATITUDE:
"51.507642",
                        LONGITUDE:
"-0.125452",
                        FIELDS: {
                            POSTAL_CODE:
"WC2N 5NS",
                            COUNTRY: "United
Kingdom",
                            ADM_LEVEL_1:
"England",
                            LOCALITY:
"Westminster",
                            STREET: "Craven
Street",
                            BUILDING: "10",
                            ADDRESS_LINE_1:
"Craven Street, 10",
                        },
                    },
                },
                {
                    ID: 453,
                    TYPE: "STRING",
                    VALUE: "Some comments",
                },
            ],
            ITEMS: [
                {
                    NAME: "Apple xs 128gb",
```

```
                    PRICE: 99999.99,
                    WEIGHT: "230.00",
                    CURRENCY: "RUB",
                    QUANTITY: 1,
                    DIMENSIONS: {
                            WIDTH: "5",
                            HEIGHT: "5",
                            LENGTH: "20",
                    },
                }
            ],
            EXTRA_SERVICES_VALUES: [
                {
                    ID: 1036,
                    CODE: "cargo_type",
                    VALUE: "small_package",
                },
                {
                    ID: 1035,
                    CODE: "door_delivery",
                    VALUE: "Y",
                },
            ],
            CONTACTS: {
                RESPONSIBLE: {
                    NAME: "Liam Williams",
                    PHONES: [
                        {
                            TYPE: "WORK",
                            VALUE:
"+73472222009",
                        }
                    ],
                },
                RECIPIENT: {
                    NAME: "John Smith",
                    PHONES: [
```

```
                                {
                                    TYPE: "WORK",
                                    VALUE:
"+79097996161",
                                }
                            ],
                        },
                    },
                }
            ]
        }
```

Пример ответа об успешном оформлении заказа:

```
{
    REQUEST_ID: "4757aca4931a4f029f49c0db4374d13d",
    SUCCESS: "Y",
}
```

Пример ответа о неудачной попытке оформления заказа:

```
{
    SUCCESS: "N",
    REASON: {
        TEXT: "Delivery is not currently available"
    },
}
```

Отмена заказа на доставку

JSON схема запроса

Пример запроса JSON:

```
{
    DELIVERY_ID: 694,
    REQUEST_ID:
"4757aca4931a4f029f49c0db4374d13d",
}
```

Пример ответа об успешной отмене:

```
{
    SUCCESS: "Y"
}
```

Пример ответа о неудачной попытке отмены заказа:

```
{
    SUCCESS: "N",
    REASON: {
        TEXT: "Delivery can not be cancelled at this
moment. Performer has already arrived to the destination."
    },
}
```

Ресурс элемента соответствия физ. и юр. лицам

Элемент соответствия физическим и юридическим лицам:

```
{
        "domain": "I",
        "personTypeId": "1"
}
```

Поля соответствуют доступному списку полей getFields.

sale.businessValuePersonDoma

```
sale.businessValuePersonDomain.add(fields.bu
sinessvaluepersondomain)
```

Метод добавляет элемент соответствия физическим и юридическим лицам.

Если операция успешна, возвращается ресурс элемента соответствия физическим и юридическим лицам в теле ответа.

Параметры

Параметр	Тип	Описание
fields.businessvaluepersondomain	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.businessValuePersonDomain.add',
```

```
    {
          fields: {
                  personTypeId: 17,
                  domain: E
                }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.businessValuePersonDoma

```
sale.businessValuePersonDomain.deleteByFilte
r(fields)
```

Метод удаляет элемент соответствия физическим и юридическим лицам.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей getFields.

Примеры

```
BX24.callMethod(

'sale.businessValuePersonDomain.deleteByFilt
```

```
er',
     {
            fields: {
                 personTypeId: 17,
                 domain: E
             }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.businessValuePersonDoma

```
sale.businessValuePersonDomain.getFields()
```

Метод возвращает поля элемента соответствия физическим и юридическим лицам.

Возвращаемые поля:

Поле	Описание
personTypeId	ID типа плательщика.
domain	Домен.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(

'sale.businessValuePersonDomain.getFields',
    {},
```

```
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.businessValuePersonDoma

```
sale.businessValuePersonDomain.list(select, filter, order, navigation)
```

Метод получает список элементов соответствия физическим и юридическим лицам.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.businessValuePersonDomain.list',
    { select:{
                personTypeId,
                domain
          } ,
          filter:{
                domain: E
          },
          order:{
                personTypeId: desc
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс статуса

Статус:

```
{
    "id":"fw",
    "type":"O",
    "sort":"100",
    "notify":"Y",
    "color":"#FF5C5B"
}
```

Поля соответствуют доступному списку полей getFields.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.status.add

```
sale.status.add(id, type)
```

Метод создания статуса.

Если операция успешна, возвращается ресурс статуса в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Код статуса.
type	string	Тип статуса.

Примеры

```
BX24.callMethod(
    'sale.status.add',
    {
                fields:{
                        id: 'XX',
```

```
                            type: 'D',
                            notify: 'Y'
                    }
            },
        function(result)
        {
            if(result.error())

    console.error(result.error().ex);
            else
                console.log(result.data());
    });
```

sale.status.delete

```
sale.status.delete(id)
```

Метод для удаления статуса.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер статуса.

Примеры

```
BX24.callMethod(
    'sale.status.delete',
    { id: id },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
```

```
        else
            console.log(result.data());
});
```

sale.status.get

```
sale.status.get(id)
```

Метод для доступа к полям статуса.

Если операция успешна, возвращается ресурс статуса в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Код статуса.

Примеры

```
BX24.callMethod(
    'sale.status.get',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.status.getFields

```
sale.status.getFields()
```

Метод возвращает поля статуса.

Возвращаемые поля:

Поле	Описание
id	Идентификатор статуса.
type	Тип статуса.
sort	Сортировка.
notify	Уведомление.
color	Цвет статуса.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'sale.status.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.status.list

```
sale.status.list(select, filter, order,
navigation)
```

Метод получает список статусов.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.status.list',
    { select:{
                id
          } ,
          filter:{},
          order:{
                id: asc
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.status.update

```
sale.status.update(id)
```

Метод для обновления статуса.

Если операция успешна, возвращается ресурс статуса в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Код статуса.

Примеры

```
BX24.callMethod(
    'sale.status.update',
    {
                id: 'XX',
                fields:{
                        type: 'D',
                        notify: 'N'
```

```
                    }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс табличной части отгрузки

Элемент коллекции табличной части отгрузки:

```
{
  "basketId":"18",
  "dateInsert":"2018-04-03T14:31:37+03:00",
  "id":"18",
  "orderDeliveryId":"36",
  "quantity":"1.0000",
  "reservedQuantity":"0.0000",
  "xmlId":"1112"
}
```

Поля соответствуют доступному списку полей getFields.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.shipmentitem.add

```
sale.shipmentitem.add(fields.shipmentitem)
```

Метод добавляет элемент коллекции табличной части отгрузки.

Если операция успешна, возвращается ресурс табличной части отгрузки в теле ответа.

Параметры

Параметр	Тип	Описание
fields.shipmentitem	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.shipmentitem.add',
    {
        fields: {
            orderDeliveryId: 33,
```

```
                    basketId: 18,
                    quantity: 1
                }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.shipmentitem.delete

```
sale.shipmentitem.delete(id)
```

Метод удаляет элемент коллекции табличной части отгрузки.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер элемента коллекции табличной части отгрузки.

Примеры

```
BX24.callMethod(
    'sale.shipmentitem.delete',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.shipmentitem.get

```
sale.shipmentitem.get(id)
```

Метод для доступа к полям элемента табличной части отгрузки.

Если операция успешна, возвращается ресурс элемента коллекции табличной части отгрузки в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер элемента коллекции оплат.

Примеры

```
BX24.callMethod(
    'sale.shipmentitem.get',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.shipmentitem.getFields

Описание и пример

```
sale.shipmentitem.getFields()
```

Метод возвращает поля табличной части отгрузки.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'sale.shipmentitem.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
```

```
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание
id	ID отгрузки.
orderId	ID заказа.
paid	Оплачен ли заказ.
datePaid	Дата оплаты заказа.
empPaidId	ID пользователя, который оплатил заказ.
paySystemId	ID платежной системы.
psStatus	Данные, пришедшие от платежной системы: статус транзакции.
psStatusCode	Данные, пришедшие от платежной системы: код транзакции.
psInvoiceId	Идентификатор оплаты в платежной системе.
psStatusDescription	Данные, пришедшие от платежной системы: описание транзакции.
psStatusMessage	Данные, пришедшие от платежной системы: сообщение.
psSum	Данные, пришедшие от платежной системы: сумма транзакции.
psCurrency	Данные, пришедшие от платежной

	системы: валюта.
psResponseDate	Данные, пришедшие от платежной системы: дата ответа от платежной системы.
payVoucherNum	Номер платежного документа.
payVoucherDate	Дата платежного документа.
datePayBefore	Дата, до которой необходимо оплатить счет (в магазине не используется).
dateBill	Дата выставления счета.
xmlId	Внешний идентификатор.
sum	Сумма оплаты.
priceCod	Стоимость оплаты при доставке (используется, например, для наложенного платежа).
currency	Валюта.
paySystemName	Название платежной системы.
responsibleId	ID пользователя, ответственного за оплату.
empResponsibleId	ID пользователя, который назначил ответственного.
dateResponsibleId	Дата назначения ответственного.
comments	комментарий к оплате.
companyId	ID компании, которая будет принимать оплату.
payReturnNum	Номер документа возврата.

payReturnDate	Дата документа возврата.
empReturnId	ID пользователя, который выполнял возврат.
payReturnComment	Комментарий к возврату.
isReturn	Выполнялся ли возврат.
marked	Промаркирована ли оплата.
dateMarked	Дата маркировки.
empMarkedId	Кем промаркирована оплата.
reasonMarked	Причина маркировки.
updated1c	Обновлена ли через 1С.
id1c	ID в 1С.
version1c	Версия документа оплаты от 1с.
externalPayment	Флаг Y/N: внешняя оплата или нет.
statusId	Статус отгрузки.
basePriceDelivery	Базовая стоимость доставки.
priceDelivery	Стоимость доставки.
customPriceDelivery	Флаг, установлена ли цена вручную.
currency	Валюта.
discountPrice	Сумма итоговой скидки/наценки для отгрузки.
allowDelivery	Флаг разрешения доставки: Y/N.
dateAllowDelivery	Дата разрешения доставки.

empAllowDeliveryId	ID пользователя, который разрешал доставку.
deducted	Флаг отгрузки: Y/N.
dateDeducted	Дата отгрузки.
empDeductedId	ID пользователя, который отгружал отгрузку.
reasonUndoDeducted	Причина отмены отгрузки.
deliveryId	ID службы доставки.
deliveryDocNum	Номер документа отгрузки.
deliveryDocDate	Дата документа отгрузки.
trackingNumber	Трэк-номер.
trackingStatus	Статус отправления.
trackingDescription	Описание статуса отправления.
trackingLastCheck	Дата последней проверки статуса отправления.
xmlId	Внешний идентификатор.
deliveryName	Название службы доставки.
canceled	Отменена ли отгрузка.
dateCanceled	Дата отмены отгрузки.
empCanceledId	ID пользователя, который отменил отгрузку.
marked	Промаркирована ли отгрузка.
dateMarked	Дата маркировки.

empMarkedId	ID пользователя, который промаркировал отгрузку.
reasonMarked	Причина маркировки.
system	Флаг, устанавливающий, системная это отгрузка или нет.
responsibleId	ID пользователя, ответственного за отгрузку.
empResponsibleId	ID пользователя, который назначил ответственного.
dateResponsibleId	Дата назначения ответственного.
comments	Комментарий.
companyId	ID компании, занимающейся отгрузкой.
updated1c	Обновлена ли отгрузка через 1С
id1c	ID отгрузки в 1С.
version1c	Версия документа отгурзки от 1С.
externalDelivery	Флаг Y/N: доставка из внешней системы или нет.

sale.shipmentitem.list

```
sale.shipmentitem.list(select, filter,
order, navigation)
```

Метод получает список элементов табличной части отгрузки.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.shipmentitem.list',
    { select:{
                id
          } ,
          filter:{
                basketId: 2
          },
          order:{
                id: asc
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.shipmentitem.update

```
sale.shipmentitem.update(Id, fields)
```

Метод для обновления элемента коллекции табличной части отгрузки.

Если операция успешна, возвращается ресурс табличной части отгрузки в теле ответа.

Параметры

Параметр	Тип	Описание
Id	string	Номер элемента коллекции табличной части отгрузки.
fields	object	Поля, соответствующие доступному списку полей fields .

Примеры

```
BX24.callMethod(
    'sale.shipmentitem.update',
```

```
    {
          id: 101,
          fields: {
                basketId: 18,
                quantity: 1
          }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс типа плательщика

Тип плательщика:

```
{
  "id":"1",
  "lid":"s1",
  "name":"Физическое лицо",
  "sort":"100",
  "active":"Y",
  "code":"test"
}
```

Поля соответствуют доступному списку полей getFields.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.persontype.add

```
sale.persontype.add(fields)
```

Метод добавляет тип плательщика.

Если операция успешна, возвращается ресурс типа плательщика в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'sale.persontype.add',
    {
          fields: {
            name: 'Физическое лицо',
```

```
                }
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.persontype.delete

```
sale.persontype.delete(id)
```

Метод удаляет тип плательщика.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер типа плательщика.

Примеры

```
BX24.callMethod(
    'sale.persontype.delete',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.persontype.get

```
sale.persontype.get(id)
```

Метод для доступа к полям типа плательщика.

Если операция успешна, возвращается ресурс типа плательщика в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер типа плательщика.

Примеры

```
BX24.callMethod(
    'sale.persontype.get',
    { id: id },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.persontype.getFields

```
sale.persontype.getFields()
```

Метод возвращает поля типа плательщика.

Возвращаемые поля:

Поле	Описание
id	ID типа плательщика.
lid	Идентификатор сайта.
name	Название типа плательщика.
code	Код типа плательщика.
sort	Сортировка.
active	Флаг активности.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'sale.payment.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.persontype.list

```
sale.persontype.list(select, filter, order,
navigation)
```

Метод получает список типов плательщиков.

Если операция успешна, возвращается список элементов в теле ответа.

Для платёжных систем, которые используются в CRM (для счетов, сделок), типы плательщиков нужно получать через метод crm.persontype.list, в случае если создаётся платёжная система для заказов, то нужно использовать **sale.persontype.list**.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.

navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.persontype.list',
    { select:{
                id
          } ,
          filter:{
            code: 'test'
          },
          order:{
                id: ASC
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.persontype.update

```
sale.persontype.update(id, fields)
```

Метод для обновления полей типа плательщика.

Если операция успешна, возвращается ресурс типа плательщика в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер типа плательщика.
fields	object	Поля, соответствующие доступному списку полей fields .

Примеры

```
BX24.callMethod(
    'sale.persontype.update',
    {
          id: 101,
```

```
            fields: {
              name: 'Физическое лицо',
              }
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс элемента торговых платформ

Элемент торговых платформ:

```
{
        "active": "N",
        "catalogSectionTabClassName": null,
        "class": null,
        "code": "ymarket",
        "description": "Интеграция магазина
с программой Яндекса \"Покупка на
Маркете\"",
        "id": "1",
        "name": "Покупки на Яндекс-Маркете",
        "settings": "",
        "xmlId": "1"
}
```

Поля соответствуют доступному списку полей getFields.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.tradePlatform.getFields

```
sale.tradePlatform.getFields()
```

Возвращает поля торговых платформ.

Возвращаемые поля:

Поле	Описание
active	Флаг активности.
catalogSectionTabClassName	Служебное поле.
class	Наименование php-класса обработчика.
code	Код торговой платформы.
description	Описание торговой платформы.
id	Идентификатор торговой платформы.
name	Наименование торговой платформы.
settings	Настройки торговой платформы.
xmlId	Внешний идентификатор торговой платформы.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'sale.tradePlatform.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

sale.tradePlatform.list

```
sale.tradePlatform.list(select, filter,
order, navigation)
```

Метод для получения списка торговых платформ.

Если операция успешна, возвращается список элементов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей getFields.
filter	object	Поля, соответствующие доступному списку полей getFields.
order	object	Поля, соответствующие доступному списку полей getFields.
navigation	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'sale.tradePlatform.list',
    { select:{
                name,
                code
          } ,
          filter:{
            active: 'Y'
          },
          order:{
                name: asc
          },
          navigation: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

calendar.accessibility.get

Возвращает занятость пользователей из списка.

Параметры функции

Параметр	Описание
* users	Список id пользователей.
* from	Дата начала периода для определения занятости.
* to	Дата окончания периода для определения занятости.
*** - обязательные параметры**	

Возвращаемое значение

Возвращает информацию о занятости по каждому запрашиваемому пользователю.

Пример использования

```
BX24.callMethod("calendar.accessibility.get"
,
```

```
            {
                        from: '2013-06-20',
                        to: '2013-12-20',
                        users: [1, 2, 34]
            });
```

calendar.event.add

Описание

Добавляет новое событие.

Возвращаемое значение

Возвращает id нового события.

Параметры

Параметр	Описание
* type	Тип календаря: ▪ user; ▪ group.
* ownerId	Идентификатор владельца календаря.
* from	Дата начала выборки.
* to	Дата окончания выборки.
from_ts	Может быть установлен вместо **from**.
to_ts	Может быть установлен вместо **to**.
* section	Идентификатор раздела.

* name	Наименование события.
skip_time	**[Y|N]** Указывает, что значение даты передается без времени. Формат даты по стандарту ISO-8601.
timezone_from	Часовой пояс даты и времени начала события. Значение по умолчанию - таймзона текущего пользователя. Указывается в строковом виде, например: Europe/Riga.
timezone_to	Часовой пояс даты и времени окончания события. Значение по умолчанию - таймзона текущего пользователя. Указывается в строковом виде, например: Europe/Riga.
description	Описание события.
color	Цвет фона события. При передачи цвета добавляемого события символ `#` необходимо передавать в unicode, как `%23`.
text_color	Цвет текста события. При передачи цвета добавляемого события символ `#` необходимо передавать в unicode, как `%23`.
accessibility	Доступность на время события: ▪ busy (занят); ▪ absent (отсутствую); ▪ quest (под вопросом); ▪ free (свободен).
importance	Важность события: ▪ high (высокая); ▪ normal (средняя); ▪ low (низкая).
private_event	**[Y|N]** Отметка частного события.
rrule	Повторяемость события.

is_meeting	**[Y|N]** Признак встречи с участниками события.
location	Место проведения.
remind	Напоминание о событии: ▪ type - временной тип напоминания (min, hour, day); ▪ count - числовое значение временного промежутка.
attendees	Список участников события (если **is_meeting** == "Y").
host	Организатор события (если **is_meeting** == "Y").
meeting	Массив параметров, включающий в себя: ▪ text - текст приглашения; ▪ open - признак открытой встречи; ▪ notify - флаг оповещения о подтверждении\отказе участников; ▪ reinvite - флаг запроса повторного подтверждения участия (при редактировании события).
*** - обязательные параметры**	

Пример

```
BX24.callMethod("calendar.event.add",
          {
                        type: 'user',
                        ownerId: '2',
                        name: 'New Event
Name',
```

```
                        description:
'Description for event',
                        from: '2013-06-14',
                        to: '2013-06-14',
                        skipTime: 'Y',
                        section: 5,
                        color: '#9cbe1c',
                        text_color:
'#283033',
                        accessibility:
'absent',
                        importance:
'normal',
                        is_meeting: 'Y',
                        private_event: 'N',
                        remind: [{type:
'min', count: 20}],
                        location:
'Kaliningrad',
                        attendees: [1, 2,
3],
                        host: 2,
                        meeting: {
                                text:
'inviting text',
                                open: true,
                                notify:
true,
                                reinvite:
false
                        }
          });
```

calendar.event.delete

Удаляет событие.

Параметры функции

Параметр	Описание
* type	Тип календаря: ▪ user; ▪ group.
* ownerId	Идентификатор владельца календаря.
* id	Идентификатор события.
*** - обязательные параметры**	

Возвращаемое значение

Возвращает **true** в случае успешного удаления.

Пример использования

```
BX24.callMethod("calendar.event.delete",
  {
```

```
        id: 698,
        type: 'user',
        ownerId: '2'
    });
```

calendar.event.get

Возвращает список событий календаря.

Параметры функции

Параметр	Описание
* type	Тип календаря: ▪ user; ▪ group.
ownerId	Идентификатор владельца календаря.
from	Дата начала выборки. Значение по умолчанию - месяц до текущей даты.
to	Дата окончания выборки. Значение по умолчанию - три месяца после текущей даты.
section	Массив разделов.
*** - обязательные параметры**	

Пример использования

```
BX24.callMethod("calendar.event.get",
          {
                        type: 'user',
                        ownerId: '1',
                        from: '2013-06-20',
                        to: '2013-08-20',
                        section: [21, 44]
          });
```

Получить события календаря компании:

```
'type'=> 'company_calendar',
'ownerId' => '' // ownerid не указывается
при выборке событий календаря компании. Он
пустой для всех событий такого типа.
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

calendar.event.get.nearest

Возвращает список будущих событий для текущего пользователя.

Параметры функции

Параметр	Описание
type	Тип календаря: ▪ user; ▪ group.
ownerId	Идентификатор владельца календаря.
days	Число дней для выборки (по умолчанию - 60).
forCurrentUser	Вывод списка событий для текущего пользователя.
maxEventsCount	Максимальное число выводимых событий.
detailUrl	url для календаря.

Пример использования

```
BX24.callMethod("calendar.event.get.nearest"
,
```

```
            {
                            type: 'user',
                            ownerId: '2',
                            days: 10,
                            forCurrentUser:
true,
                            detailUrl:
'/company/personal/user/#user_id#/calendar/'
            });
```

calendar.event.getbyid

Метод возвращает событие календаря по идентификатору.

Параметры

Параметр	Описание	С версии
id	Обязательный параметр, число. Возвращает массив с полями сущности события или *null*	

Пример

```
BX24.callMethod("calendar.event.getbyid",
{id: 324});

/*
     * Returns event by it id
     *
     * @param array $params - incomoning
params:
     * $params['id'] - int, (required)
calendar event id
     * @return event or null
     * @throws \Bitrix\Rest\RestException
     *
```

```
     * @example (Javascript)
     *
BX24.callMethod("calendar.event.getbyid",
     * {
     *          id: 324
     * });
     *
     */
```

calendar.event.update

Описание

Редактирует существующее событие.

Возвращаемое значение

Возвращает id отредактированного события.

Параметры

Параметр	Описание
* id	Идентификатор события.
* type	Тип календаря.
* ownerId	Идентификатор владельца календаря.
from	Дата начала выборки.
to	Дата окончания выборки.
from_ts	Может быть установлен вместо **from**.
to_ts	Может быть установлен вместо **to**.
* section	Идентификатор раздела.
* name	Наименование события.

skip_time	**[Y|N]** Указывает, что значение даты передается без времени.
timezone_from	Часовой пояс даты и времени начала события. Значение по умолчанию - таймзона текущего пользователя.
timezone_to	Часовой пояс даты и времени окончания события. Значение по умолчанию - таймзона текущего пользователя.
description	Описание события.
color	Цвет фона события.
text_color	Цвет текста события.
accessibility	Доступность на время события: ▪ busy (занят); ▪ absent (отсутствую); ▪ quest (под вопросом); ▪ free (свободен).
importance	Важность события: ▪ high (высокая); ▪ normal (средняя); ▪ low (низкая).
private_event	**[Y|N]** Отметка частного события.
rrule	Повторяемость события.
is_meeting	**[Y|N]** Признак встречи с участниками события.
location	Место проведения.
remind	Напоминание о событии:

	▪ type - временной тип напоминания (min, hour, day); ▪ count - числовое значение временного промежутка.
attendees	Список участников события (если **is_meeting** == "Y").
host	Организатор события (если **is_meeting** == "Y").
meeting	Массив параметров, включающий в себя: ▪ text - текст приглашения; ▪ open - признак открытой встречи; ▪ notify - флаг оповещения о подтверждении\отказе участников; ▪ reinvite - флаг запроса повторного подтверждения участия (при редактировании события).
*** - обязательные параметры**	

Пример

```
BX24.callMethod("calendar.event.update",
          {
                        id: 699
                        type: 'user',
                        ownerId: '2',
                        name: 'Changed Event
Name',
                        description: 'New
description for event',
                        from: '2013-06-17',
                        to: '2013-06-17',
                        skipTime: 'Y',
```

```
                            section: 5,
                            color: '#9cbe1c',
                            text_color:
'#283033',
                            accessibility:
'free',
                            importance:
'normal',
                            is_meeting: 'N',
                            private_event: 'Y',
                            remind: [{type:
'min', count: 10}]
          });
```

calendar.meeting.params.set

Устанавливает параметры события для текущего пользователя, если он является его участником.

Параметры функции

Параметр	Описание
eventId	Идентификатор события.
accessibility	Доступность на время события: ▪ busy (занят); ▪ absent (отсутствую); ▪ quest (под вопросом); ▪ free (свободен).
remind	Напоминание о событии: ▪ type - временной тип напоминания (min, hour, day); ▪ count - числовое значение временного промежутка.

Возвращаемое значение

Возвращает **true** в случае успешного выполнения.

Пример использования

```
BX24.callMethod("calendar.meeting.params.set
",
          {
                        eventId: '651',
                        accessibility:
'free',
                        remind: [{type:
'min', count: 20}]
          });
```

calendar.meeting.status.get

Возвращает статус участия текущего пользователя в событии.

Параметры функции

Параметр	Описание
* eventId	Идентификатор события
*** - обязательные параметры**	

Возвращаемые значения

Возвращает статус ("Y", "N", "Q").

Пример использования

```
BX24.callMethod("calendar.meeting.status.get
",
          {
                         eventId: '651'
          });
```

calendar.meeting.status.set

Устанавливает статус участия в событии для текущего пользователя.

Параметры функции

Параметр	Описание
eventId	Идентификатор события
status	Статус: ▪ Y - подтвердил участие; ▪ N - отказался; ▪ Q - под вопросом.

Возвращаемые значения

Возвращает **true** если установка статуса прошла успешно.

Пример использования

```
BX24.callMethod("calendar.meeting.status.set
",
            {
```

```
                                eventId: '651',
                                status: 'Y'
            });
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

calendar.resource.add

```
calendar.resource.add(name)
```

Добавляет новый ресурс, принимает на вход массив с параметрами.

Параметры функции

Параметр	Описание
* name	Наименование ресурса (строка).
*** - обязательные параметры**	

Возвращаемые значения

Возвращает ID добавленного ресурса.

Пример использования

```
BX24.callMethod("calendar.resource.add",
 {
```

```
            name: 'My resource title'
  });
```

calendar.resource.booking.list

```
calendar.resource.booking.list(filter)
```

Предоставляет возможность выбрать бронирования ресурсов.

Возвращаемые значения

Возвращает данные о каждом бронировании. Бронирования имеют идентичные событиям поля, т.к. являются, по сути, событиями.

Параметры

Параметр	Описание
* filter	Поля фильтра.
*** - обязательные параметры**	

Примеры использования

Первый вариант: для возможности оценить бронирования (занятость) определенных ресурсов на какой-то период. Может использоваться для создания собственных представлений занятости или для использования в логике.

```
BX24.callMethod("calendar.resource.booking.list", {
   filter: {
       resourceTypeIdList: [10852, 10888, 10873, 10871, 10853] // передается список id ресурсов, которые можно выбрать методом calendar.resource.list
       from: '2018-06-20',
       to: '2018-08-20',
   }
});
```

Второй вариант: возможность выбрать бронирования по их id (это значения [dw]UF-поля[/dw][di]Маска "UF_*"- для выборки всех пользовательских полей (без множественных)[/di], привязанного к CRM сущности).

```
BX24.callMethod("calendar.resource.booking.list", {
   filter: {
       resourceIdList: [10, 18, 17] // эти ID берутся из значения UF-поля типа resourcebooking у CRM сущностей LEAD|DEAL
   }
});
```

calendar.resource.delete

```
calendar.resource.delete(resourceId)
```

Удаляет ресурс.

Возвращаемые значения

Возвращает true, если удаление успешно.

Параметры

Параметр	Описание
* resourceId	Идентификатор ресурса.
*** - обязательные параметры**	

Пример использования

```
BX24.callMethod("calendar.resource.delete",
 {
         resourceId: 521
 });
```

calendar.resource.list

```
calendar.resource.list()
```

Возвращает список (массив) всех ресурсов.

Возвращаемые значения

Возвращает массив, каждый элемент которого имеет поля "ID", "NAME", "CREATED_BY".

Параметры

Без параметров

Пример использования

```
BX24.callMethod("calendar.resource.list")
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

calendar.resource.update

```
calendar.resource.update(resourceId, name)
```

Изменяет ресурс.

Возвращаемые значения

Возвращает ID измененного раздела.

Параметры функции

Параметр	Описание
resourceId	Идентификатор ресурса.
* name	Имя ресурса.
*** - обязательные параметры**	

Пример использования

```
BX24.callMethod("calendar.resource.update",
 {
```

```
            resourceId: 325,
            name: 'Changed Resource Name'
    });
```

calendar.section.add

Добавляет новый календарь. Здесь и в дальнейшем **section** будет именоваться как "календарь".

На текущий момент метод добавляет новый календарь только для пользователя от которого выполняется метод calendar.section.add. В будущем это ограничение будет снято.

Параметры функции

Параметр	Описание
* type	Тип календаря: ▪ user; ▪ group.
* ownerId	Идентификатор владельца календаря.
* name	Название календаря.
description	Описание календаря.
color	Цвет календаря.
text_color	Цвет текста в календаре.
export	Список параметров: ▪ ALLOW - разрешить экспорт календаря;

	▪ SET - устанавливается период, за который производить экспорт.
access	Массив данных доступа к календарю.
*** - обязательные параметры**	

Возвращаемые значения

Возвращает id созданных календарей.

Пример использования

```
BX24.callMethod("calendar.section.add",
          {
                        type: 'user',
                        ownerId: '2',
                        name: 'New Section',
                        description:
'Description for section',
                        color: '#9cbeee',
                        text_color:
'#283000',
                        export: [{ALLOW:
false}]
                        access: {
                                'D114': 17,
                                'G2': 13,
                                'U2':15
                        }
          });
```

calendar.section.delete

Удаляет календарь. Здесь и в дальнейшем **section** будет именоваться как "календарь".

Параметры функции

Параметр	Описание
* type	Тип календаря: ▪ user; ▪ group.
* ownerId	Идентификатор владельца календаря.
* id	Идентификатор календаря.
*** - обязательные параметры**	

Возвращаемые значения

Возвращает **true** в случае успешного удаления.

Пример использования

```
BX24.callMethod("calendar.section.delete",
          {
                        type: 'user',
                        ownerId: '2',
                        id: 521
          });
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

calendar.section.get

Возвращает список календарей. Здесь и в дальнейшем **section** будет именоваться как "календарь".

Параметры функции

Параметр	Описание
* type	Тип календаря: ▪ user ▪ group ▪ company_calendar ▪ другие типы, в том числе пользовательские.
* ownerId	Идентификатор владельца календаря.
*** - обязательные параметры**	

Пример использования

```
BX24.callMethod("calendar.section.get",
          {
                        type: 'user',
                        ownerId: '1'
          });
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

calendar.section.update

Обновляет календарь. Здесь и в дальнейшем **section** будет именоваться как "календарь".

Параметры функции

Параметр	Описание
* type	Тип календаря: ▪ user; ▪ group.
* ownerId	Идентификатор владельца календаря.
* id	Идентификатор календаря.
name	Название календаря.
description	Описание календаря.
color	Цвет календаря.
text_color	Цвет текста в календаре.
export	Список параметров: ▪ ALLOW - разрешить экспорт календаря; ▪ SET - устанавливается период, за который производить экспорт.

access	Массив данных доступа к календарю.
*** - обязательные параметры**	

Возвращаемые значения

Возвращает id модифицированных календарей.

Пример использования

```
BX24.callMethod("calendar.section.update",
          {
                        id: 325,
                        type: 'user',
                        ownerId: '2',
                        name: 'Changed
Section Name',
                        description: 'New
description for section',
                        color: '#9cbeAA',
                        text_color:
'#283099',
                        export: [{ALLOW:
false}]
                        access: {
                                'D114': 17,
                                'G2': 13,
                                'U2':15
                        }
          });
```

calendar.settings.get

Возвращает основные настройки календаря. Относится к настройкам модуля, доступ только у администраторов.

Параметры функции

Функция не имеет параметров.

Пример использования

```
BX24.callMethod("calendar.settings.get",
{});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

calendar.user.settings.get

Возвращает пользовательские настройки календаря.

Параметры функции

Функция не имеет параметров.

Пример использования

```
BX24.callMethod("calendar.user.settings.get"
,{});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

calendar.user.settings.set

Сохраняет пользовательские настройки календаря.

Параметры функции

Параметр	Описание
* settings	Список пользовательских настроек.
*** - обязательные параметры**	

Возвращаемые значения

Возвращает **true** в случае успешного выполнения.

Пример использования

```
BX24.callMethod("calendar.user.settings.set"
,
          {
                        settings: {
                                tabId:
'month',
                                meetSection:
'23',
```

```
                                    blink: true,
showDeclined: false,
                                    showMuted:
true
                            }
          });
```

Частые кейсы::Пример коннектора Открытых линий для онлайн-чата на сайте

Внимание! Пример заработает только на локальном приложении, в случае использования вебхуков он не сработает.

Пример позволит создать на вашем сайте онлайн чат. Диалог привязывается к сессии пользователя на вашем сайте и домену.

1. После установки всех файлов, запускаем через браузер файл **install_connector.php** для первоначальной настройки ОЛ. При успешной первоначальной настройке у вас отобразится надпись `successfully`.
2. Открыть раздел с открытыми линиями и там находим блок с названием "ExampleSiteChat".
3. Открыв раздел с "ExampleSiteChat" нажимаем кнопку "подключить".
4. Если вы всё верно настроили вам выдаст также надпись `successfully`.
5. Установка закончена, вы можете открыть файл **index.php** и начать первый диалог в чате.

Внимание! Для использования данного примера необходимо настроить работу класса CRest и подключить файл crest.php в файлах, где используется данный класс подробнее.

В примере предусмотрена проверка: при вызове файла **install_connector.php** должна создаваться папка `\chats` в той же

директории, где расположен этот файл. В этой папке хранится информация по чатам.

Если появится сообщение: `error creat dir /chats`, то первоначальная настройка коннектора не выполнилась, т.к. неверно настроены права и скрипт не может создать папку `/chats`.

Создаём файл **function.php**:

```
<?
include_once('crest.php');

function getConnectorID()
{
        return 'example_connector_1';
}

function getChat($chatID)
{
        $result = [];
        if (file_exists(__DIR__ . '/chats/' . $chatID .
'.txt'))
        {
                $result =
json_decode(file_get_contents(__DIR__ . '/chats/' . $chatID
. '.txt'), 1);
        }

        return $result;
}

function saveMessage($chatID, $arMessage)
{
        $arMessages = getChat($chatID);
        $count = count($arMessages);
        $arMessages['message' . $count] = $arMessage;
        if (file_put_contents(__DIR__ . '/chats/' . $chatID
. '.txt', json_encode($arMessages)))
        {
                $return = $count;
        }
        else
        {
                $return = false;
        }

        return $return;
}
```

```
function getLine()
{
        return file_get_contents(__DIR__ . '/line_id.txt');
}

function setLine($line_id)
{
        return file_put_contents(__DIR__ . '/line_id.txt',
intVal($line_id));
}

function convertBB($var)
{

        $search = array(
                '/\[b\](.*?)\[\/b\]/is',
                '/\[br\]/is',
                '/\[i\](.*?)\[\/i\]/is',
                '/\[u\](.*?)\[\/u\]/is',
                '/\[img\](.*?)\[\/img\]/is',
                '/\[url\](.*?)\[\/url\]/is',
                '/\[url\=(.*?)\](.*?)\[\/url\]/is'
        );

        $replace = array(
                '<strong>$1</strong>',
                '<br>',
                '<em>$1</em>',
                '<u>$1</u>',
                '<img src="$1" />',
                '<a href="$1">$1</a>',
                '<a href="$1">$2</a>'
        );

        $var = preg_replace($search, $replace, $var);

        return $var;
}
```

Создаём файл **handler.php**:

Переменные **$widgetUri** и **$widgetName** обязательны, если необходимо чтобы данный коннектор отображался в списке коннекторов в виджете на сайте и соответственно выводился там. Иначе их можно не заполнять.

- **widgetUri** - Путь с иконки в виджете. (Например, при нажатии на иконку фейсбука открывается чат в фейсбуке.)
- **widgetName** - название коннектора в виджете.

```
<?
require_once('function.php');


$widgetUri = '';//detail page chat
"https://example.com/index.php"
$widgetName = 'ExampleSiteChatWidget';//name connector in
widget

$connector_id = getConnectorID();

if (!empty($_REQUEST['PLACEMENT_OPTIONS']) &&
$_REQUEST['PLACEMENT'] == 'SETTING_CONNECTOR')
{
        //activate connector
        $options =
json_decode($_REQUEST['PLACEMENT_OPTIONS'], true);
        $result = CRest::call(
                'imconnector.activate',
                [
                        'CONNECTOR' => $connector_id,
                        'LINE' => intVal($options['LINE']),
                        'ACTIVE' =>
intVal($options['ACTIVE_STATUS']),
                ]
        );
        if (!empty($result['result']))
        {
                //add data widget
                if(!empty($widgetUri) &&
!empty($widgetName))
                {
                        $resultWidgetData = CRest::call(

'imconnector.connector.data.set',
                                [
                                        'CONNECTOR' =>
$connector_id,
                                        'LINE' =>
intVal($options['LINE']),
                                        'DATA' => [
                                                'id' =>
$connector_id.'line'.intVal($options['LINE']),//
                                                'url_im' =>
$widgetUri,
                                                'name' =>
$widgetName
                                        ],
                                ]
                        );
```

```
if(!empty($resultWidgetData['result']))
                        {
                                setLine($options['LINE']);
                                echo 'successfully';
                        }
                }
                else
                {
                        setLine($options['LINE']);
                        echo 'successfully';
                }
        }
}

if(
        $_REQUEST['event'] == 'ONIMCONNECTORMESSAGEADD'
        && !empty($_REQUEST['data']['CONNECTOR'])
        && $_REQUEST['data']['CONNECTOR'] == $connector_id
        && !empty($_REQUEST['data']['MESSAGES'])
)
{
        foreach ($_REQUEST['data']['MESSAGES'] as
$arMessage)
        {
                $idMess = saveMessage($arMessage['chat']
['id'], $arMessage);
                $resultDelivery = CRest::call(
                        'imconnector.send.status.delivery',
                        [
                                'CONNECTOR' =>
$connector_id,
                                'LINE' => getLine(),
                                'MESSAGES' => [
                                        [
                                                'im' =>
$arMessage['im'],
                                                'message'
=> [

'id' => [$idMess]

                                                ],
                                                'chat' => [

'id' => $arMessage['chat']['id']
                                                ],
                                        ],
                                ]
                        ]
                );
        }
```

```
}
```

Создаём файл **install_connector.php**

необходимо указать в **$handlerUrl** путь до файла handler.php, созданного выше:

```
<?

require_once('function.php');

$handlerUrl = 'https://yourdomain.yyy/handler.php';


//creat dir for save chats (recommend using database)
@mkdir(__DIR__ . '/chats/', 0775, true);
if(!file_exists(__DIR__ . '/chats/'))
{
        echo 'error creat dir "chats"';
}
else
{
        $connector_id = getConnectorID();
        $result = CRest::call(
                'imconnector.register',
                [
                        'ID' => $connector_id,
                        'NAME' => 'ExampleSiteChat',
                        'ICON' => [
                                'DATA_IMAGE' =>
'data:image/svg+xml;charset=US-
ASCII,%3Csvg%20version%3D%221.1%22%20id%3D%22Layer_1%22%20x
mlns%3D%22http%3A//www.w3.org/2000/svg%22%20x%3D%220px%22%2
0y%3D%220px%22%0A%09%20viewBox%3D%220%200%2070%2071%22%20st
yle%3D%22enable-
background%3Anew%200%200%2070%2071%3B%22%20xml%3Aspace%3D%2
2preserve%22%3E%0A%3Cpath%20fill%3D%22%230C99BA%22%20class%
3D%22st0%22%20d%3D%22M34.7%2C64c-11.6%2C0-22-7.1-26.3-
17.8C4%2C35.4%2C6.4%2C23%2C14.5%2C14.7c8.1-8.2%2C20.4-
10.7%2C31-
6.2%0A%09c12.5%2C5.4%2C19.6%2C18.8%2C17%2C32.2C60%2C54%2C48
.3%2C63.8%2C34.7%2C64L34.7%2C64z%20M27.8%2C29c0.8-
0.9%2C0.8-2.3%2C0-3.2l-1-1.2h19.3c1-0.1%2C1.7-0.9%2C1.7-
1.8%0A%09v-0.9c0-1-0.7-1.8-1.7-1.8H26.8l1.1-1.2c0.8-
0.9%2C0.8-2.3%2C0-3.2c-0.4-0.4-0.9-0.7-1.5-0.7s-1.1%2C0.2-
1.5%2C0.7l-4.6%2C5.1%0A%09c-0.8%2C0.9-
0.8%2C2.3%2C0%2C3.2l4.6%2C5.1c0.4%2C0.4%2C0.9%2C0.7%2C1.5%2
C0.7C26.9%2C29.6%2C27.4%2C29.4%2C27.8%2C29L27.8%2C29z%20M44
%2C41c-0.5-0.6-1.3-0.8-2-0.6%0A%09c-0.7%2C0.2-1.3%2C0.9-
```

1.5%2C1.6c-0.2%2C0.8%2C0%2C1.6%2C0.5%2C2.2l1%2C1.2H22.8c-
1%2C0.1-1.7%2C0.9-
1.7%2C1.8v0.9c0%2C1%2C0.7%2C1.8%2C1.7%2C1.8h19.3l-
1%2C1.2%0A%09c-0.5%2C0.6-0.7%2C1.4-
0.5%2C2.2c0.2%2C0.8%2C0.7%2C1.4%2C1.5%2C1.6c0.7%2C0.2%2C1.5
%2C0%2C2-0.6l4.6-5.1c0.8-0.9%2C0.8-2.3%2C0-
3.2L44%2C41z%20M23.5%2C32.8%0A%09c-1%2C0.1-1.7%2C0.9-
1.7%2C1.8v0.9c0%2C1%2C0.7%2C1.8%2C1.7%2C1.8h23.4c1-
0.1%2C1.7-0.9%2C1.7-1.8v-0.9c0-1-0.7-1.8-1.7-
1.9L23.5%2C32.8L23.5%2C32.8z%22/%3E%0A%3C/svg%3E%0A',
                                'COLOR' => '#a6ffa3',
                                'SIZE' => '100%',
                                'POSITION' => 'center',
                        ],
                        'ICON_DISABLED' => [
                                'DATA_IMAGE' =>
'data:image/svg+xml;charset=US-
ASCII,%3Csvg%20version%3D%221.1%22%20id%3D%22Layer_1%22%20x
mlns%3D%22http%3A//www.w3.org/2000/svg%22%20x%3D%220px%22%2
0y%3D%220px%22%0A%09%20viewBox%3D%220%200%2070%2071%22%20st
yle%3D%22enable-
background%3Anew%200%200%2070%2071%3B%22%20xml%3Aspace%3D%2
2preserve%22%3E%0A%3Cpath%20fill%3D%22%230C99BA%22%20class%
3D%22st0%22%20d%3D%22M34.7%2C64c-11.6%2C0-22-7.1-26.3-
17.8C4%2C35.4%2C6.4%2C23%2C14.5%2C14.7c8.1-8.2%2C20.4-
10.7%2C31-
6.2%0A%09c12.5%2C5.4%2C19.6%2C18.8%2C17%2C32.2C60%2C54%2C48
.3%2C63.8%2C34.7%2C64L34.7%2C64z%20M27.8%2C29c0.8-
0.9%2C0.8-2.3%2C0-3.2l-1-1.2h19.3c1-0.1%2C1.7-0.9%2C1.7-
1.8%0A%09v-0.9c0-1-0.7-1.8-1.7-1.8H26.8l1.1-1.2c0.8-
0.9%2C0.8-2.3%2C0-3.2c-0.4-0.4-0.9-0.7-1.5-0.7s-1.1%2C0.2-
1.5%2C0.7l-4.6%2C5.1%0A%09c-0.8%2C0.9-
0.8%2C2.3%2C0%2C3.2l4.6%2C5.1c0.4%2C0.4%2C0.9%2C0.7%2C1.5%2
C0.7C26.9%2C29.6%2C27.4%2C29.4%2C27.8%2C29L27.8%2C29z%20M44
%2C41c-0.5-0.6-1.3-0.8-2-0.6%0A%09c-0.7%2C0.2-1.3%2C0.9-
1.5%2C1.6c-0.2%2C0.8%2C0%2C1.6%2C0.5%2C2.2l1%2C1.2H22.8c-
1%2C0.1-1.7%2C0.9-
1.7%2C1.8v0.9c0%2C1%2C0.7%2C1.8%2C1.7%2C1.8h19.3l-
1%2C1.2%0A%09c-0.5%2C0.6-0.7%2C1.4-
0.5%2C2.2c0.2%2C0.8%2C0.7%2C1.4%2C1.5%2C1.6c0.7%2C0.2%2C1.5
%2C0%2C2-0.6l4.6-5.1c0.8-0.9%2C0.8-2.3%2C0-
3.2L44%2C41z%20M23.5%2C32.8%0A%09c-1%2C0.1-1.7%2C0.9-
1.7%2C1.8v0.9c0%2C1%2C0.7%2C1.8%2C1.7%2C1.8h23.4c1-
0.1%2C1.7-0.9%2C1.7-1.8v-0.9c0-1-0.7-1.8-1.7-
1.9L23.5%2C32.8L23.5%2C32.8z%22/%3E%0A%3C/svg%3E%0A',
                                'SIZE' => '100%',
                                'POSITION' => 'center',
                                'COLOR' => '#ffb3a3',
                        ],
                        'PLACEMENT_HANDLER' => $handlerUrl,
                ]

```
        );

        if (!empty($result['result']))
        {
                $resultEvent = CRest::call(
                        'event.bind',
                        [
                                'event' =>
'OnImConnectorMessageAdd',
                                'handler' => $handlerUrl,
                        ]
                );
                if (!empty($resultEvent['result']))
                {
                        echo 'successfully';
                }
        }
}
```

Создаём файл **ajax.php**:

```
<?

require_once('function.php');
session_start();
$chatID = 'chat' . md5($_SERVER['HTTP_ORIGIN']) .
md5(session_id());

$type = $_POST['type'];
$connector_id = getConnectorID();
$line_id = getLine();

/*
        simple example save chat, must lost any data
        recommend using database
*/

if ($type == 'chat_history'):
        $arChat = getChat($chatID);

        if (!empty($arChat)):
                foreach ($arChat as $item):?>
                        <div class="col-12 alert alert-
warning text-<?=(!empty($item['im'])) ? 'left' : 'right'?>"
                                 style=" background-color:
<?=(!empty($item['im'])) ? '#fbfbfb' : '#ccf2ff'?>">
                                <?
=convertBB($item['message']['text'])?>
```

```
                        </div>
                <? endforeach;
        endif;

elseif ($type == 'send_message'):
        $arMessage = [
                'user' => [
                        'id' => $chatID,
                        'name' =>
htmlspecialchars($_POST['name']),
                ],
                'message' => [
                        'id' => false,
                        'date' => time(),
                        'text' =>
htmlspecialchars($_POST['message']),
                ],
                'chat' => [
                        'id' => $chatID,
                        'url' =>
htmlspecialchars($_SERVER['HTTP_REFERER']),
                ],
        ];
        $id = saveMessage($chatID, $arMessage);
        $result['error'] = 'error_save';
        if ($id !== false)
        {
                $arMessage['message']['id'] = $id;
                $result = CRest::call(
                        'imconnector.send.messages',
                        [
                                'CONNECTOR' =>
$connector_id,
                                'LINE' => $line_id,
                                'MESSAGES' => [$arMessage],
                        ]
                );
        }

        echo json_encode(
                [
                        'chat' => $chatID,
                        'post' => $_POST,
                        'result' => $result
                ]
        );

endif;
```

Создаём файл **index.php**:

```
<body>
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/cs
s/bootstrap.min.css"
          crossorigin="anonymous">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jqu
ery.min.js"></script>
<div class="container-fluid">
        <div class=" m-5">
                <div id="chat_history" class="row">
                        <div class="spinner-border m-5
text-success" role="status">
                                <span class="sr-
only">Loading...</span>
                        </div>
                </div>
                <div id="chat_form" class=" mt-5 mr-auto
ml-auto mb-5">
                        <form id="form_message">
                                <div class="form-group">
                                        <label
for="name">Name</label>
                                        <input type="text"
class="form-control" placeholder="Name">
                                </div>

                                <div class="form-group">
                                        <label
for="message">Message</label>
                                        <textarea
class="form-control" name="message" rows="3"
placeholder="your message here"></textarea>
                                </div>


                                <input class="btn btn-
primary" type="submit" name="send" value="send">
                        </form>
                </div>
        </div>
</div>
<script>
        $(document).ready(function () {
                function updateChat()
                {
                        $.ajax({
                                'method': 'POST',
```

```
                                'dataType': 'html',
                                'url': 'ajax.php',
                                'data':
'type=chat_history',
                                success: function (data)
{//success callback

$('#chat_history').text('').html(data);
                                }
                        });
                }

                setInterval(updateChat, 5000);
                updateChat();

                $('#form_message').on('submit', function
(el) {//event submit form
                        el.preventDefault();//the default
action of the event will not be triggered
                        $('#chat_form').addClass('spinner-
border');
                        $('#form_message').hide();
                        var formData = $(this).serialize();
                        $.ajax({
                                'method': 'POST',
                                'dataType': 'json',
                                'url': 'ajax.php',
                                'data': formData +
'&type=send_message',
                                success: function (data)
{//success callback
                                        updateChat();

$('#chat_form').removeClass('spinner-border');
                                        $('#form_message
textarea[name=message]').val('');

$('#form_message').show();

                                }
                        });
                });
        });
</script>
</body>
```

© «Битрикс», 2001-2008, «1С-

imconnector.register

Метод регистрации нового типа коннектора.

Примечание. Если нужно чтобы коннектор отображался в общем списке коннекторов в виджете на сайте, нужно использовать метод imconnector.connector.data.set. Пример использования.

Параметры

Параметр	Описание	ве
ID	Уникальный идентификатор коннектора. Лучше предварять своим префиксом, дабы избежать пересечения с текущими и возможными будущими ID коннекторов. Можно использовать: цифры, буквы, знак подчеркивания.	
NAME	Отображаемое имя коннектора.	
ICON	Массив описания иконки коннектора, где: ▪ **DATA_IMAGE** - DATA-представление иконки SVG. Для конвертации можно использовать сервис	

	http://www.grumpicon.com/. **Пример** ▪ ***COLOR*** - цвет. Пример: `#1900ff` ▪ ***SIZE*** - размер. Пример: `90%` ▪ ***POSITION*** - позиция SVG. Пример: `center`	
PLACEMENT_HANDLER	Ссылка на URL обработчика встройки, который будет вызываться для показа пользователям интерфейса настройки коннектора в слайдере. Подробнее о механизме встройки читайте в учебном курсе. В отличии от базового механизма встройки, в данном случае указывается только URL обработчика, т. е. нет нужды указывать на конкретный идентификатор места встройки.	
ICON_DISABLED	Массив описания иконки коннектора для **неактивного** варианта, где: ▪ **DATA_IMAGE** - DATA-представление иконки SVG. Для конвертации можно использовать сервис http://www.grumpicon.com/. **Пример** ▪ ***COLOR*** - цвет. Пример: `#1900ff` ▪ ***SIZE*** - размер. Пример: `90%` ▪ ***POSITION*** - позиция SVG. Пример: `center`	

DEL_EXTERNAL_MESSAGES	Можно ли удалять входящие сообщения? По умолчанию: да.	
EDIT_INTERNAL_MESSAGES	Можно ли редактировать свои сообщения? По умолчанию: да.	
DEL_INTERNAL_MESSAGES	Можно ли удалять свои сообщения? По умолчанию: да.	
NEWSLETTER	Можно ли использовать канал для CRM-рассылки? По умолчанию: да.	
NEED_SYSTEM_MESSAGES	Можно ли отправлять системные сообщения в канал? По умолчанию: да. Примеры: приветствие, завершающая реплика и т. п.	
NEED_SIGNATURE	Можно ли отправлять подпись в самом сообщении? По умолчанию: да. Пример: перед текстом сообщения добавляется строка с именем оператора.	
CHAT_GROUP	Y\N. По умолчанию - Y/ Является ли чат данного канала с внешней стороны групповым. По умолчанию писать не могут. Так же не создаются лиды и прочие crm сущности.	
COMMENT	Описание для обработчика места встройки (см. параметр PLACEMENT_HANDLER).	

Примечание: *курсивом* выделены необязательные параметры.

imconnector.unregister

Метод удаления коннектора.

Параметры

Параметр	Описание	С версии
ID	Уникальный идентификатор коннектора.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

imconnector.send.messages

Метод отправки сообщений в ОЛ.

Параметры

Параметр	Описание	С версии
CONNECTOR	ID коннектора (который был указан при регистрации обработчика).	
LINE	ID открытой линии.	
MESSAGES	Массив сообщений, где сообщения описываются массивом следующего формата:	

```
array(
  array(
     //Массив описания
пользователя
     'user' => array(
        'id',//ID
пользователя во внешней
системе *
        'last_name',//
Фамилия
        'name',//Имя
        'picture' =>
           array(
              'url'//Ссылка
на аватарку пользователя,
```

```
доступную для портала
            ),
         'url',//Ссылка на
профиль пользователя
         'sex',//Пол.
Допустимо male и female
         'email', //email
         'phone', //телефон
      ),
      //Массив описания
сообщения
      'message' => array(
         'id', //ID сообщения
во внешней системе.*
         'date', //Время
сообщения в формате
timestamp *
         'disable_crm' => 'Y'
,//отключить чат трекер (CRM
трекер)
         'text', //Текст
сообщения. Должен быть
указан элемент text или
files.
                //Допустимое
форматирование (BB коды)
описаны
                //здесь:
https://dev.1c-
bitrix.ru/learning/course/?
COURSE_ID=93&LESSON_ID=7679
         'files' => array(//
Массив описаний файлов, где
каждый файл описывается

//массивом, со ссылкой,
которая доступна порталу
            array('url' =>
'Ссылка на файл'),
            array('url' =>
'Ссылка на файл'),
            ...
         )
      ),
      //Массив описания чата
      'chat' => array(
         'id',//ID чата во
внешней системе *
         'name', //Имя чата
во внешней системе
         'url', //Ссылка на
```

```
чат во внешней системе
      ),
  ),
  array(...),
);
```

Формат передаваемого файла не имеет ограничений. В чате вложение в сообщение может форматироваться как картинка для типов: jpe, jpg, jpeg, png, webp, gif, bmp.

imconnector.update.messages

Метод обновления сообщений в ОЛ.

Параметры

Параметр	Описание	С версии
CONNECTOR	ID коннектора (который был указан при регистрации обработчика).	
LINE	ID открытой линии.	
MESSAGES	Массив сообщений, где сообщения описываются массивом следующего формата:	

```
array(
  array(
     //Массив описания
пользователя
     'user' => array(
        'id',//ID
пользователя во внешней
системе *
        'last_name',//
Фамилия
        'name',//Имя
        'picture' =>
           array(
              'url'//Ссылка
на аватарку пользователя,
```

```
доступную для портала
            ),
         'url',//Ссылка на
профиль пользователя
         'sex',//Пол.
Допустимо male и female
      ),
      //Массив описания
сообщения
      'message' => array(
         'id', //ID сообщения
во внешней системе.*
         'date', //Время
сообщения в формате
timestamp *
         'text', //Текст
сообщения. Должен быть
указан элемент text или
files.
                //Допустимое
форматирование (BB коды)
описаны
                //здесь:
https://dev.1c-
bitrix.ru/learning/course/?
COURSE_ID=93&LESSON_ID=7679
         'files' => array(//
Массив описаний файлов, где
каждый файл описывается

//массивом, со ссылкой,
которая доступна порталу
            array('url'),
            array('url'),
            ...
         )
      ),
      //Массив описания чата
      'chat' => array(
         'id',//ID чата во
внешней системе *
         'name', //Имя чата
во внешней системе
         'url', //Ссылка на
чат во внешней системе
      ),
   ),
   array(...),
);
```

imconnector.delete.messages

Метод удаления сообщений в ОЛ.

Параметры

Параметр	Описание	С версии
CONNECTOR	ID коннектора (который был указан при регистрации обработчика).	
LINE	ID открытой линии.	
MESSAGES	Массив сообщений, где сообщения описываются массивом следующего формата:	

```
array(
  array(
     //Массив описания пользователя
     'user' => array(
        'id',//ID пользователя во внешней системе *
     ),
     //Массив описания сообщения
     'message' => array(
        'id', //ID сообщения во внешней системе.*
     ),
```

```
     //Массив описания чата
     'chat' => array(
        'id',//ID чата во
внешней системе *
     ),
  ),
  array(...)
);
```

imconnector.send.status.delive

Метод подтверждения доставки сообщения из ОЛ во внешнюю систему.

Параметры

Параметр	Описание	С версии
CONNECTOR	ID коннектора (который был указан при регистрации обработчика).	
LINE	ID открытой линии.	
MESSAGES	Массив сообщений, где сообщения описываются массивом следующего формата:	

```
array(
  array(
     'im',//Пересылается
элемент 'im' из входящего
сообщения ОЛ
     'message' => array(
        'id'//Массив ID во
внешней системе. Именно
массив, а не единичное
            //значение, даже
если ID один.
     ),
     'chat' => array(
        'id'//ID чата во
```

```
внешней системе.
      ),
  ),
  array(...),
);
```

imconnector.connector.data.se

Метод устанавливает данные для rest-коннектора.

Параметры

Метод	Описание	С версии
CONNECTOR	Идентификатор коннектора	
LINE	Идентификатор линии, к которой он привязан.	
DATA	Массив с данными для сохранения: ▪ **id** - идентификатор учетной записи, которая подключена к этому коннектору. ▪ **url** и **url_im** - ссылки на чат. url_im используется в виджете, но если не установлена, то будет использован просто url. ▪ **name** - название канала, которое будет отображаться в виджете.	

Пример

```
//imconnector.connector.data.set
        function connectorDataSet()
        {
                var params = {
                        CONNECTOR:
'myrestconnector',
                        LINE: 1,
                        DATA: {
                                id: 123,
                                url:
'http://localhost',
                                url_im:
'http://localhost',
                                name: 'My
rest connector name'
                        }
                };
                BX24.callMethod(

'imconnector.connector.data.set',
                        params,
                        function (result) {
                                if
(result.error())

alert("Error: " + result.error());
                                else

alert("Успешно: " + result.data());
                        }
                );
        }
```

imconnector.send.status.readin

Метод подтверждения прочтения сообщения из ОЛ во внешнюю систему.

Параметры

Параметр	Описание	С версии
CONNECTOR	ID коннектора (который был указан при регистрации обработчика).	
LINE	ID открытой линии.	
MESSAGES	Массив сообщений, где сообщения описываются массивом следующего формата:	

```
array(
  array(
     'im',//Пересылается
элемент 'im' из входящего
сообщения ОЛ
     'message' => array(
        'id'//Массив ID во
внешней системе. Именно
массив, а не единичное
            //значение, даже
если ID один.
     ),
     'chat' => array(
        'id'//ID чата во
```

```
внешней системе.
      ),
  ),
  array(...),
);
```

imconnector.set.error

Метод отключения настроенного коннектора в ОЛ из внешней системы при статусе **ошибка**.

Параметры

Параметр	Описание	С версии
CONNECTOR	ID коннектора (который был указан при регистрации обработчика).	
LINE	ID открытой линии.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

imopenlines.crm.chat.getLastId

Метод получает ID последнего чата, который привязан к CRM сущности.

Параметры

Параметр	Описание	С версии
CRM_ENTITY_TYPE	Тип CRM сущности (LEAD/DEAL/COMPANY/CONTACT, обязательный)	
CRM_ENTITY	Идентификатор CRM сущности (обязательный)	

Пример

```
//imopenlines.crm.chat.getLastId
function crmChatGetLastId() {
   var params = {
      CRM_ENTITY_TYPE: 'LEAD',
//LEAD|DEAL|COMPANY|CONTACT
      CRM_ENTITY: 1,
   };
```

```
   BX24.callMethod(
      'imopenlines.crm.chat.getLastId',
      params,
      function (result) {
         if (result.error())
            alert("Error: " +
result.error());
         else
            alert("Успешно: " +
result.data());
      }
   );
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

imopenlines.crm.chat.user.add

Метод добавляет пользователя в чат, полученый по CRM-сущности.

Параметры

Параметр	Описание	С версии
CRM_ENTITY_TYPE	Тип CRM сущности (LEAD/DEAL/COMPANY/CONTACT, обязательный)	
CRM_ENTITY	Идентификатор CRM сущности (обязательный)	
USER_ID	Идентификатор пользователя или бота, которого мы хотим добавить в чат (обязательный)	

Пример

```
//imopenlines.crm.chat.user.add
function crmChatUserAdd()
{
        var params = {
```

```
                        CRM_ENTITY_TYPE:
'LEAD', //LEAD|DEAL|COMPANY|CONTACT
                        CRM_ENTITY: 1,
                        USER_ID: 1
                };
                BX24.callMethod(

'imopenlines.crm.chat.user.add',
                        params,
                        function (result) {
                                if
(result.error())

alert("Error: " + result.error());
                                else

alert("Успешно: " + result.data());
                        }
                );
```

imopenlines.config.add

Описание и пример

Метод для добавления линии.

Пример

```
        //imopenlines.config.add
        function configAdd()
        {
                var params = {
                        PARAMS: {
                                LINE_NAME:
'New line name',
                                ...
                        }
                };
                BX24.callMethod(

'imopenlines.config.add',
                        params,
                        function (result) {
                                if
(result.error())

alert("Error: " + result.error());
```

```
                                    else

alert("Успешно: " + result.data());
                                }
                        );
                }
```

Параметры

Метод	Описание	С версии
PARAMS	Массив параметров для добавления (необязательный). Список полей - ниже.	

Поля	Описание
ID	Идентификатор линии
WELCOME_BOT_ENABLE	При обращении клиента назначить отв чат-бота. [Y/N (по умолчанию)] - эта о быть Y, чтобы бот работал
WELCOME_BOT_JOIN	Когда подключать чат-бота (*first* (по-у always),
WELCOME_BOT_ID	Идентификатор бота (int, по-умолчани
WELCOME_BOT_TIME	Через какое время переводить разгово в очередь (int, 60 по умолчанию)
WELCOME_BOT_LEFT	Когда отключать чат-бота (*queue* (по-у close),
ACTIVE	Активность линии [Y/N (по умолчанию)

LINE_NAME	Название линии (необязательный)
CRM	Проверять пользователя по CRM [Y/N (умолчанию)]
CRM_CREATE	Если клиент не найден в CRM (строка, *none*)
CRM_FORWARD	Направлять обращение на ответственн сотрудника в случае идентификации кл умолчанию)/N]
CRM_SOURCE	Источник для нового лида (строка, по-у 'create')
CRM_TRANSFER_CHANGE	Автоматически менять ответственного з ручном перенаправлении обращения н оператора [Y (по умолчанию)/N]
QUEUE_TIME	Время до перехода обращения к следуі сотруднику из очереди (int, 60 по умол
NO_ANSWER_TIME	Время до отметки сообщения как неотв 60 по умолчанию)
QUEUE_TYPE	Тип очереди (*evenly* (по-умолчанию), s
TIMEMAN	Не направлять обращение на оператор начат рабочий день или установлен пе (по умолчанию)]
CHECK_ONLINE	При распределении обращений провер доступность оператора [Y/N (по умолча
CHECKING_OFFLINE	Постоянная проверка доступности опер распределении обращений [Y/N (по ум
WELCOME_MESSAGE	Отправить автоматический ответ на пер сообщение клиента [Y (по умолчанию/N
WELCOME_MESSAGE_TEXT	Текст автоматического ответа (строка, null)

AGREEMENT_MESSAGE	Отправить предупреждение о сборе пе данных [Y/N (по умолчанию)]
AGREEMENT_ID	ID соглашения в системе (int, 0 по-умо
NO_ANSWER_RULE	Действие если операторы не ответили (*none* (по-умолчанию), text)
NO_ANSWER_TEXT	Текст автоматического ответа (строка, *null*)
WORKTIME_ENABLE	Настройка рабочего времени Открытой (по умолчанию)]
WORKTIME_FROM	Режим работы "с" (строка формата '00:
WORKTIME_TO	Режим работы "до" (строка формата '00
WORKTIME_TIMEZONE	Временная зона (формат типа 'Europe/
WORKTIME_HOLIDAYS	Список праздничных дней (строка, При 1.01,2.01,7.01,23.02,8.03,1.05,9.05,12.
WORKTIME_DAYOFF	Список кодов выходных дней (массив, 'TU'])
WORKTIME_DAYOFF_RULE	Обработка обращения в нерабочее вре умолчанию), 'text')
WORKTIME_DAYOFF_TEXT	Текст автоматического ответа (в нераб (строка, по-умолчанию *null*),
CLOSE_RULE	Действие при завершении обращения к ('none', *text* (по-умолчанию))
CLOSE_TEXT	Текст автоматического ответа (строка, *null*)
FULL_CLOSE_TIME	Время до полного закрытия обращения закрытия его оператором) (int, по-умол минут)
AUTO_CLOSE_RULE	Будет выполнено действие при автомат

	закрытии (*none* (по-умолчанию), 'text')
AUTO_CLOSE_TEXT	Текст автоматического ответа (строка, null)
AUTO_CLOSE_TIME	Время последней активности для автоз диалога (int, по-умолчанию 0)
VOTE_MESSAGE	Отправлять запрос клиенту на оценку н обслуживания, char(1),[Y (по умолчани
VOTE_CLOSING_DELAY	Закрывать сессию сразу после оценки char(1), [Y/N (по умолчанию)]
VOTE_MESSAGE_1_TEXT	Текст для запроса оценки в онлайн-чат network
VOTE_MESSAGE_1_LIKE	Текст для положительной оценки в онл bitrix24 network
VOTE_MESSAGE_1_DISLIKE	Текст для отрицательной оценки в онла bitrix24 network
VOTE_MESSAGE_2_TEXT	Текст для запроса оценки в других кан Telegram, Facebook*, Вконтакте и другі
VOTE_MESSAGE_2_LIKE	Текст для положительной оценки в дру (Viber, Telegram, Facebook*, Вконтакте
VOTE_MESSAGE_2_DISLIKE	Текст для отрицательной оценки в друг (Viber, Telegram, Facebook*, Вконтакте
QUICK_ANSWERS_IBLOCK_ID	Идентификатор инфоблока быстрых от умолчанию, 0)
LANGUAGE_ID	Настройка языковых предпочтений (ch умолчанию *null*)
OPERATOR_DATA	Информация об операторах в очереди умолчанию), queue, hide)
DEFAULT_OPERATOR_DATA	Информация об операторах по-умолчан поля:

	▪ NAME - имя ▪ AVATAR - ссылка на аватар ▪ AVATAR_ID - идентификатор файл портале
QUEUE	Очередь ответственных сотрудников. М ▪ U - массив id пользователей, кото в очередь Можно передавать в формате `QUEUE: [{ENTITY_TYPE: "user", ENTITY_ID: '`
QUEUE_OPERATOR_DATA	Данные операторов для отображения в поля: ▪ U - массив пользователей с данны пользователя" => массив данных: ▪ NAME - имя ▪ USER_WORK_POSITION - дол ▪ AVATAR - ссылка на аватар ▪ AVATAR_ID - идентификатор на портале

* деятельность организации запрещена в Российской Федерации.

imopenlines.config.delete

Метод для удалении линии.

Параметры

Параметр	Описание	С версии
CONFIG_ID	Идентификатор линии (обязательный).	

Пример

```
//imopenlines.config.delete
        function configDelete()
        {
                var params = {
                        CONFIG_ID: 1
                };
                BX24.callMethod(

'imopenlines.config.delete',
                        params,
                        function (result) {
```

```
                                    if
(result.error())

alert("Error: " + result.error());
                                    else

alert("Успешно: " + result.data());
                            }
                    );
            }
```

imopenlines.config.get

Метод позволяет получить линию по её ID.

Параметры

Параметр	Описание	С версии
CONFIG_ID	ID линии (обязательный)	
WITH_QUEUE	Выводить со списком операторов линии (Y/N, по-умолчанию: Y)	
SHOW_OFFLINE	Показывать ли список вместе с операторами, которые не в сети (Y/N, по-умолчанию: Y)	

Примеры

```
//imopenlines.config.get
function configGet()
{
        var params = {
                CONFIG_ID:    1,
```

```
                WITH_QUEUE:   'Y',
                SHOW_OFFLINE: 'Y'
        };
        BX24.callMethod(
                'imopenlines.config.get',
                params,
                function (result) {
                        if (result.error())

alert("Error: " + result.error());
                        else

alert("Успешно: " + result.data());
                }
        );
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

imopenlines.config.list.get

Метод для получения списка линий.

Параметры

Параметр	Описание	С версии
PARAMS	Массив [dw]параметров[/dw] [di]Список доступных параметров идентичен параметрам метода imopenlines.config.add[/di] для выборки (select, order, filter) (необязательный)	
OPTIONS	Массив дополнительных опций (на данный момент только поле 'QUEUE' => 'Y'/'N' - Очередь ответственных сотрудников) (необязательный)	

Пример

```
//imopenlines.config.list.get
        function configListGet()
```

```
        {
                var params = {
                        PARAMS: {
                                select: {

'ID',
                                        ...
                                },
                                order: {
                                        ID:
'ASC',
                                        ...
                                },
                                filter: {
                                        ID:
1,
                                        ...
                                }
                        },
                        OPTIONS: {
                                QUEUE: Y
                        }
                };
                BX24.callMethod(

'imopenlines.config.list.get',
                        params,
                        function (result) {
                                if
(result.error())

alert("Error: " + result.error());
                                else

alert("Успешно: " + result.data());
                        }
```

```
                        );
            }
```

imopenlines.config.path.get

Метод получения путей к публичной части открытых линий и домену портала.

Без параметров

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

imopenlines.config.update

Метод для обновления линии.

Параметры

Метод	Описание	С версии
CONFIG_ID	ID линии (обязательный)	
PARAMS	Массив [dw]параметров[/dw] [di]Список возможных параметров идентичен списку метода imopenlines.config.add. [/di] для обновления (необязательный)	

Пример

```
//imopenlines.config.update
function configUpdate()
{
        var params = {
                CONFIG_ID: 1,
                PARAMS: {
```

```
                                    LINE_NAME:
'New line name',
                                    ...
                            }
                    };
                    BX24.callMethod(

'imopenlines.config.update',
                            params,
                            function (result) {
                                    if
(result.error())

alert("Error: " + result.error());
                                    else

alert("Успешно: " + result.data());
                            }
                    );
            }
```

OnImConnectorLineDelete

Удаление открытой линии. Нужно удалить свои подключенные коннекторы.

Параметры

Параметр	Описание	С версии
LINE_ID	ID удаляемой открытой линии.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

OnImConnectorMessageAdd

Событие отмечает новое сообщение из ОЛ.

Обязательно нужно вызвать в ответ метод **imconnector.send.status.delivery**, иначе в мессенджере сообщение будет значиться как недоставленное.

Параметры

Параметр	Описание	С версии
CONNECTOR	ID коннектора (по нему проводится проверка, относится ли это событие к проверяющему).	
LINE	ID открытой линии.	
DATA	Массив сообщений, где каждое сообщение описывается массивом следующего вида:	

```
array (
    //Массив параметров для
связи (возвращается в
сообщении о доставке)
      'im' =>
      array (
        //ID чата в рамках
Битрикс24
                'chat_id' =>
```

```
845,
                //ID
сообщения в рамках Битрикс24
                'message_id'
=> 344029,
      ),
      'message' =>
      array (
//Текст сообщения
                'text' =>
'[b]Сергей "Покоев":[/b][br]
Тестовое сообщение'
      ),
      'chat' =>
      array (
        //ID внешнего чата
         'id' => '2',
    ),
);
```

OnImConnectorMessageDelete

Событие удаления сообщения из ОЛ.

Параметр	Описание	С версии
CONNECTOR	ID коннектора (по нему проводится проверка, относится ли это событие к проверяющему).	
LINE	ID открытой линии.	
DATA	Массив сообщений, где каждое сообщение описывается массивом следующего вида:	

```
 array (
      'im' =>
      array (
        //ID чата в рамках
Битрикс24
        'chat_id' => '845',
        //ID сообщения в
рамках Битрикс24
        'message_id' =>
'344029',
      ),
      'message' =>
      array (
        //Массив ID
редактируемых сообщений во
внешней
        //системе (в
sendStatusDelivery должны
вернуться
```

```
        //новые ID, даже
если они такие же).
        //Предусмотреть
вариант, когда может прийти
одно значение!
        'id' =>
        array (
          0 => '99',
        ),
        //Новый текст
сообщения
        'text' => '[b]Сергей
"Покоев":[/b][br] Тестовое
сообщение 55',
      ),
      'chat' =>
      array (
        //ID чата во внешней
системе
        'id' => '2',
      ),
 );
```

OnImConnectorMessageUpdate

Событие изменения сообщения из ОЛ.

Обязательно нужно вызвать в ответ метод **imconnector.send.status.delivery**, чтобы в мессенджере сообщение значилось как успешно измененное.

Параметры

Параметр	Описание	С версии
CONNECTOR	ID коннектора (по нему проводится проверка, относится ли это событие к проверяющему).	
LINE	ID открытой линии.	
DATA	Массив сообщений, где каждое сообщение описывается массивом следующего вида:	

```
array (
      'im' =>
      array (
        //ID чата в рамках Битрикс24
        'chat_id' => '845',
        //ID сообщения в рамках Битрикс24
        'message_id' =>
```

```
'344029',
      ),
      'message' =>
      array (
        //Массив ID
редактируемых сообщений во
внешней
        //системе (в
sendStatusDelivery должны
вернуться
        //новые ID, даже
если они такие же)
        //Предусмотреть
вариант, когда может прийти
одно значение!
        'id' =>
        array (
          0 => '99',
        ),
        //Новый текст
сообщения
        'text' => '[b]Сергей
"Покоев":[/b][br] Тестовое
сообщение 55',
      ),
      'chat' =>
      array (
        //ID чата во внешней
системе
        'id' => '2',
      ),
 );
```

OnImConnectorStatusDelete

Срабатывает, когда клиент отключает подключенный канал на конкретной линии. На портале данные уже удалены, необходимо у себя произвести необходимые действия по отключению.

Параметры

Параметр	Описание	С версии
CONNECTOR	ID отключённого коннектора.	
LINE	ID удаляемой открытой линии.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

imconnector.activate

Метод устанавливает, активировать или деактивировать канал конкретной ОЛ.

Примечание. Если нужно чтобы коннектор отображался в общем списке коннекторов в виджете на сайте, нужно использовать метод imconnector.connector.data.set. Пример использования.

Параметры

Параметр	Описание	С версии
CONNECTOR	ID коннектора (который был указан при регистрации обработчика).	
LINE	ID открытой линии.	
ACTIVE	1 или 0. Активировать или деактивировать.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

imconnector.status

Метод возвращает текущее состояние коннектора.

Параметры

Параметр	Описание	С версии
LINE	ID открытой линии.	
CONNECTOR	ID коннектора.	
ERROR	Была ли ошибка в канале.	
CONFIGURED	Настроен ли канал.	
STATUS	Текущий статус канала (может ли передавать данные (настроен и не содержит ошибку).	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.paysystem.handler.add

Описание

```
sale.paysystem.handler.add(
 fields
)
```

Метод добавляет рест-обработчик.

Параметры

Параметр	Описание	С версии
fields	Набор полей(массив вида `array("поле":"значение"[, ...]))`, содержащий значения, описывающие обработчик. Доступные поля: ▪ NAME - Название обработчика ▪ CODE - Уникальный код обработчика в системе ▪ SETTINGS - Настройки обработчика ▪ SORT - Сортировка	

Примеры

Пример 1

```
BX24.callMethod(
        "sale.paysystem.handler.add",
        {
                'NAME' : 'Обработчик.Rest',
// Название обработчика
                'SORT' : 100,
// Сортировка
                'CODE' : 'resthandlercode',
// Уникальный код обработчика в системе
                'SETTINGS' : {
// Настройки обработчика
                        'CURRENCY' :
{'RUB'},
// Список валют, которые поддерживает
обработчик
                        'FORM_DATA' : {
// Настройки формы
                                'ACTION_URI'
: 'https://payment_service_url',        //
URL, на который будет отправляться форма
                                'METHOD' :
'POST',
// Метод отправки формы
                                'PARAMS' : {
// Карта соответствия полей между полями
формы и параметрами обработчика: массив вида
array(код_поля => код_параметра_обработчика)

'serviceid' : 'REST_SERVICE_ID',

'invoiceNumber' : 'PAYMENT_ID',
```

```
'Sum' : 'PAYMENT_SHOULD_PAY',

'customer' : 'PAYMENT_BUYER_ID',
                                }
                        },
                        'CODES' : {
// Список параметров обработчика

"REST_SERVICE_ID" : {
// Код параметра

"NAME" : 'Номер магазина',
// Название параметра

"DESCRIPTION" : 'Номер магазина',
// Описание параметра

'SORT' : 100,
// Сортировка
                                },

"REST_SERVICE_KEY" : {

"NAME" : 'Секретный ключ',

"DESCRIPTION" : 'Секретный ключ',

'SORT' : 300,
                                },
                                "PAYMENT_ID"
: {

"NAME" : 'Номер оплаты',

'SORT' : 400,
```

```
'GROUP' : 'PAYMENT',

'DEFAULT' : {

'PROVIDER_KEY' : 'PAYMENT',

'PROVIDER_VALUE' : 'ACCOUNT_NUMBER'
                                        }
                                  },

"PAYMENT_SHOULD_PAY" : {

"NAME" : 'Сумма оплаты',

'SORT' : 600,

'GROUP' : 'PAYMENT',

'DEFAULT' : {

'PROVIDER_KEY' : 'PAYMENT',

'PROVIDER_VALUE' : 'SUM'
                                        }
                                  },

"PS_CHANGE_STATUS_PAY" : {

"NAME" : 'Автоматическая смена статуса
оплаты',

'SORT' : 700,

"INPUT" : {

'TYPE' : 'Y/N'
                                        },
```

```
                                    },

"PAYMENT_BUYER_ID" : {

"NAME" : 'Код покупателя',

'SORT' : 1000,

'GROUP' : 'PAYMENT',
// тип значения строка, чекбокс и т.д.

'DEFAULT' : {
// Значение по умолчанию

'PROVIDER_KEY' : 'ORDER',
// Тип значения(см. доступный список ниже)

'PROVIDER_VALUE' : 'USER_ID'
// Значение(см. доступный список ниже)
                                            }
                                    }
                            }
                    }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.info("Обработчик добавлен с ID " +
result.data());
        }
);
```

Пример 2

Начиная с версии **20.5.0** модуля **sale** добавлено новое значение **FIELDS** вместо старого **PARAMS** (появилась возможность передавать произвольные поля для REST-обработчиков платёжных систем).

Если значение ключа `CODE` - строка, то значение будет использоваться для поиска соответствия между полями формы и параметрами обработчика: массив вида `array('CODE' => 'код_параметра_обработчика')`. Название и значение будут получены из параметров обработчика (`'CODES'`).

Если в ключе `CODE` передан объект, то в форме оплаты будет добавлено поле указанного типа. Поддерживаются типы:

- **STRING** (строка)
- **Y/N** (чекбокс)
- **ENUM** (список)

По умолчанию поля формы скрыты. Чтобы сделать поле видимым, необходимо передать `'VISIBLE' : 'Y'`.

```
BX.rest.callMethod(
    "sale.paysystem.handler.add",
    {
        'NAME' : 'Обработчик.Rest',
        'SORT' : 100,
        'CODE' : 'resthandlercodedoc',
        'SETTINGS' : {
            'CURRENCY' : ['RUB'],
            'FORM_DATA' : {
                'ACTION_URI' :
'https://payment_service_url',
                'METHOD' : 'POST',
                'FIELDS' : {
                    'phone' : {
                        'CODE': {
                            'NAME' : 'Номер
```

```
телефона',
                                'TYPE' :
'STRING',
                            }
                            'VISIBLE' : 'Y',
                        },
                        'paymentId' : {
                            'CODE' :
'PAYMENT_ID',
                            'VISIBLE' : 'Y'
                        },
                        'serviceid' : {
                            'CODE' :
'REST_SERVICE_ID'
                        }
                    }
                },
                'CODES' : {
                    "REST_SERVICE_ID" : {
                        "NAME" : 'Номер
магазина',
                        "DESCRIPTION" : 'Номер
магазина',
                        'SORT' : 100,
                    },
                    "REST_SERVICE_KEY" : {
                        "NAME" : 'Секретный
ключ',
                        "DESCRIPTION" :
'Секретный ключ',
                        'SORT' : 300,
                    },
                    "PAYMENT_ID" : {
                        "NAME" : 'Номер оплаты',
                        'SORT' : 400,
                        'GROUP' : 'PAYMENT',
                        'DEFAULT' : {
```

```
                        'PROVIDER_KEY' :
'PAYMENT',
                        'PROVIDER_VALUE' :
'ACCOUNT_NUMBER'
                    }
                },
                "PAYMENT_SHOULD_PAY" : {
                    "NAME" : 'Сумма оплаты',
                    'SORT' : 600,
                    'GROUP' : 'PAYMENT',
                    'DEFAULT' : {
                        'PROVIDER_KEY' :
'PAYMENT',
                        'PROVIDER_VALUE' :
'SUM'
                    }
                },
                "PS_CHANGE_STATUS_PAY" : {
                    "NAME" : 'Автоматическая
смена статуса оплаты',
                    'SORT' : 700,
                    "INPUT" : {
                        'TYPE' : 'Y/N'
                    },
                },
                "PAYMENT_BUYER_ID" : {
                    "NAME" : 'Код
покупателя',
                    'SORT' : 1000,
                    'GROUP' : 'PAYMENT',
                    'DEFAULT' : {
                        'PROVIDER_KEY' :
'ORDER',
                        'PROVIDER_VALUE' :
'USER_ID'
                    }
                }
```

```
            }
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.info("Обработчик
добавлен с ID " + result.data());
    }
);
```

Примечание: Если передавать и **FIELDS**, и **PARAMS**, то будет использоваться только **FIELDS**.

Пример 3

Начиная с версии **21.700.0** модуля **sale** добавлены новые сценарии оплаты:

- Форма
- iFrame
- Checkout

Форма

При добавлении обработчика в **SETTINGS** нужно передать **FORM_DATA** (как в примерах выше). Способ подходит если от покупателя ничего не нужно запрашивать, либо запрашивается небольшой набор данных.

Поля формы автоматически выводятся в соответствии с дизайном страницы оплаты.

Данные формы (значения **FIELDS** из **FORM_DATA**) будут отправлены на **ACTION_URI**.

iFrame

При добавлении обработчика в **SETTINGS** нужно передать **IFRAME_DATA** (вместо **FORM_DATA**). По адресу из **ACTION_URI** должна располагаться страница, которая будет загружена в iframe на сайт продавца.

При загрузке iframe через метод Window.postMessage() на **ACTION_URI** будут переданы следующие данные:

- BX_SYSTEM_PARAMS:
  - RETURN_URL- текущая страница;
  - PAYSYSTEM_ID - идентификатор платежной системы;
  - PAYMENT_ID - идентификатор оплаты;
  - SUM - сумма платежа;
  - CURRENCY - валюта.
- BX_COMPUTED_STYLE (стили родительского элемента iframe, полученные методом window.getComputedStyle()
- значения FIELDS из IFRAME_DATA

Получить значения в iframe можно через обработчик события **message**, например:

```
//js
document.addEventListener("DOMContentLoaded"
, function() {
    window.addEventListener("message",
function (event) {
        // получение данных от сайта (от
платёжной системы)
        var paymentData = event.data;

                // работа с BX_SYSTEM_PARAMS
        if (paymentData.BX_SYSTEM_PARAMS)
        {
            // ...
        }
```

```
        // использование стилей сайта
        if (paymentData.BX_COMPUTED_STYLE)
        {
            document.body.style.background =
paymentData.BX_COMPUTED_STYLE.background;
            document.body.style.color =
paymentData.BX_COMPUTED_STYLE.color;
        }
    }, false);
});
```

По умолчанию ширина iframe - 100% родительского элемента, а высота - 350px.

Размеры iframe можно изменить. Для этого нужно из iframe передать высоту и/или ширину на сайт продавца. Например:

```
//js
document.addEventListener("DOMContentLoaded"
, function() {
        var size = {
                width:
document.body.scrollWidth,
                height:
document.body.scrollHeight
        };

        // отправка данных на сайт продавца
        parent.postMessage(size, "*");
});
```

width и **height** зарезервированные названия переменных и на сайте продавца обрабатываются только они.

Checkout

При добавлении обработчика в **SETTINGS** нужно передать **CHECKOUT_DATA** (вместо **FORM_DATA**).

По адресу из **ACTION_URI** должен располагаться скрипт, который обработает полученные данные, создаст оплату и вернёт идентификатор созданной оплаты и url страницы оплаты.

На **ACTION_URI** будут передаваться данные для оплаты:

- BX_SYSTEM_PARAMS:
  - RETURN_URL - текущая страница;
  - PAYSYSTEM_ID - идентификатор платежной системы;
  - PAYMENT_ID - идентификатор оплаты;
  - SUM - сумма платежа;
  - CURRENCY - валюта.
- значения FIELDS из CHECKOUT_DATA

В ответ на запрос к **ACTION_URI**, скрипт должен вернуть идентификатор созданной оплаты и url страницы оплаты.

Например:

```
//php
// ... код обработки полученных данных и
создания оплаты

$result = [
    'PAYMENT_URL' => '', // url страницы
оплаты
    'PAYMENT_ID' => '', // идентификатор
оплаты
];

header('Content-Type:application/json;
charset=UTF-8');
echo json_encode($result);
```

Покупатель перейдёт на ссылку из **PAYMENT_URL** автоматически или по клику ну кнопку "Купить".

Возможные значения

Возможные значения PROVIDER_KEY	
ORDER	Параметры
PROPERTY	Свойства счета
PAYMENT	Оплата
USER	Пользователь
VALUE	Значение типа строка
Y\N	Чекбокс

Возможные значения PROVIDER_VALUE	
ORDER	ID: ID (для счетов соответствует ID счета), ACCOUNT_NUMBER: Номер заказа (для счетов соответствует номеру счета), ORDER_TOPIC: Тема, DATE_INSERT: Дата заказа (для счетов соответствует дате счета), DATE_INSERT_DATE: Дата заказа (без времени) (для счетов соответствует дате счета (без времени)), DATE_BILL: Дата и время выставления, DATE_BILL_DATE: Дата выставления, DATE_PAY_BEFORE: Срок оплаты, SHOULD_PAY: Сумма счета (для счетов соответствует сумме счета), CURRENCY: Валюта, PRICE: Стоимость заказа (для счетов соответствует стоимости счета), PRICE_DELIVERY: Стоимость доставки,

	DISCOUNT_VALUE: Величина скидки, USER_ID: Код покупателя, PAY_SYSTEM_ID: Код платежной системы, DELIVERY_ID: Код службы доставки, TAX_VALUE: Налог, USER_DESCRIPTION: Комментарий
PAYMENT	ID: ID ACCOUNT_NUMBER: Номер оплаты, DATE_BILL: Дата и время выставления, DATE_BILL_DATE: Дата выставления (без времени), SUM: Сумма оплаты, CURRENCY: Валюта
USER	ID: Код покупателя, LOGIN: Логин, NAME: Имя, SECOND_NAME: Отчество, LAST_NAME: Фамилия, EMAIL: EMail, PERSONAL_PROFESSION: Профессия, PERSONAL_WWW: Персональный веб-сайт, PERSONAL_ICQ: Номер ICQ, PERSONAL_GENDER: Пол, PERSONAL_FAX: Номер факса, PERSONAL_MOBILE: Номер телефона, PERSONAL_STREET: Адрес, PERSONAL_MAILBOX: Почтовый ящик, PERSONAL_CITY: Город, PERSONAL_STATE: Штат, PERSONAL_ZIP: Индекс, PERSONAL_COUNTRY: Страна, WORK_COMPANY: Компания, WORK_DEPARTMENT: Отдел, WORK_POSITION: Должность, WORK_WWW: Сайт компании, WORK_PHONE: Рабочий телефон, WORK_FAX: Рабочий факс, WORK_STREET: Адрес компании, WORK_MAILBOX: Рабочий почтовый ящик, WORK_CITY: Город компании, WORK_STATE: Штат компании, WORK_ZIP: Индекс компании, WORK_COUNTRY: Страна компании

sale.paysystem.handler.delete

```
sale.paysystem.handler.delete(
 id
)
```

Метод удаляет рест-обработчик.

Параметры

Параметр	Описание	С версии
id	Идентификатор обработчика	

Пример

```
var id = prompt("Введите ID");

BX24.callMethod('sale.paysystem.handler.dele
te', {"id": id},
        function(result)
        {
```

```
                if(result.error())

console.error(result.error());
                else
                {

console.info("Обработчик удален");
                }
        }
);
```

sale.paysystem.handler.list

```
sale.paysystem.handler.list(
)
```

Метод для получения списка рест-обработчиков.

Параметры

Без параметров

Пример

```
BX24.callMethod(
        "sale.paysystem.handler.list",
        {},
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.dir(result.data());
        }
);
```

sale.paysystem.handler.update

```
sale.paysystem.handler.update (
   id,
   fields
)
```

Метод выполняет обновление рест-обработчика.

Параметры

Параметр	Описание	С версии
id	Идентификатор обработчика	
fields	Набор полей(массив вида `array("поле":"значение"[, ...]))`, содержащий значения, описывающие обработчик. Доступные поля: ▪ NAME - Название обработчика ▪ CODE - Уникальный код обработчика в системе ▪ SETTINGS - Настройки обработчика ▪ SORT - Сортировка	

Пример

```
var id = prompt("Введите ID");
var name = prompt("Введите имя");

BX24.callMethod('sale.paysystem.handler.upda
te', {"id": id, "fields": {NAME : name}},
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                {

console.info("Обработчик обновлён");
                }
        }
);
```

sale.paysystem.add

Описание

```
sale.paysystem.add(
 fields
)
```

Метод добавляет платежную систему.

Параметры

Параметр	Описание	С версии
fields	Набор полей (массив вида `array("поле":"значение"[, ...])`), содержащий значения, описывающие платежную систему. Доступные поля: ▪ NAME - Название системы. ▪ DESCRIPTION - описание платёжной системы. ▪ XML_ID - Символьный код. ▪ PERSON_TYPE_ID - ID типа плательщика. ▪ BX_REST_HANDLER - Код обработчика в системе. ▪ ACTIVE - Флаг активности платежной системы.	

- ENTITY_REGISTRY_TYPE - Привязка платежной системы (значение для магазина: `ORDER`; значение для счетов crm: `CRM_INVOICE`)
- LOGOTYPE - Логотип платёжной системы (картинка в формате [ds]Base64[/ds][di] **Base64** — стандарт кодирования двоичных данных при помощи только 64 символов ASCII. Алфавит кодирования содержит текстово-цифровые латинские символы A-Z, a-z и 0-9 (62 знака) и 2 дополнительных символа, зависящих от системы реализации. Каждые 3 исходных байта кодируются 4 символами (увеличение на 1/3).

  Подробнее...[/di]). Поле доступно с версии **20.0.550**
- NEW_WINDOW - Поле, отвечающее за настройку "Открывать в новом окне" (по умолчанию значение `N`). Доступно с версии **20.100.0**

Примеры

```
BX24.callMethod(
        "sale.paysystem.add",
        {
                'NAME' : 'Платежная система
```

```
c Rest',
// Название платежной системы
                'PERSON_TYPE_ID' : 1,
// ID типа плательщика
                'ACTIVE' : 'Y',
// Флаг активности платежной системы
                'BX_REST_HANDLER' :
'resthandlercode',
// Код обработчика в системе
                'SETTINGS' : {
// Настройки обработчика для данной
платежной системы
                        'REST_SHOP_ID' : {
// Код параметра
                                'TYPE' :
'VALUE',
// Тип значение
                                'VALUE' :
'111111111'
// Значение
                        },
                        'REST_SCID' : {
                                'TYPE' :
'VALUE',
                                'VALUE' :
'2222222'
                        }
                }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.info("Платежная система добавлен с
```

```
ID " + result.data());
        }
);
```

```
BX24.callMethod(
        "sale.paysystem.add",
        {
                'NAME' : 'Платежная система
с Rest', // Название платежной системы
                'PERSON_TYPE_ID' : 1, // ID
типа плательщика
                'ACTIVE' : 'Y', // Флаг
активности платежной системы
                'BX_REST_HANDLER' :
'resthandlercode', // Код обработчика в
системе
                'LOGOTYPE' :
'iVBORw0KGgoAAAANSUhEUgAAASQAAACmCAID...',
// Логотип платёжной системы в base64
                'SETTINGS' : { // Настройки
обработчика для данной платежной системы
                        'REST_SHOP_ID' : {
// Код параметра
                                'TYPE' :
'VALUE', // Тип значение
                                'VALUE' :
'111111111' // Значение
                        },
                        'REST_SCID' : {
                                'TYPE' :
'VALUE',
                                'VALUE' :
'2222222'
                        }
                }
```

```
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.info("Платежная система добавлен с
ID " + result.data());
        }
);
```

Возможные значения

Возможные значения TYPE	
ORDER	Параметры
PROPERTY	Свойства счета
PAYMENT	Оплата
USER	Пользователь
VALUE	Значение типа строка
INPUT	Чекбокс

Возможные значения VALUE	
ORDER	ID: ID, ACCOUNT_NUMBER: Номер заказа, ORDER_TOPIC: Тема, DATE_INSERT: Дата заказа, DATE_INSERT_DATE: Дата заказа (без времени),

	DATE_BILL: Дата и время выставления, DATE_BILL_DATE: Дата выставления, DATE_PAY_BEFORE: Срок оплаты, SHOULD_PAY: Сумма счета, CURRENCY: Валюта, PRICE: Стоимость заказа, PRICE_DELIVERY: Стоимость доставки, DISCOUNT_VALUE: Величина скидки, USER_ID: Код покупателя, PAY_SYSTEM_ID: Код платежной системы, DELIVERY_ID: Код службы доставки, TAX_VALUE: Налог, USER_DESCRIPTION: Комментарий
PAYMENT	ID: ID ACCOUNT_NUMBER: Номер оплаты, DATE_BILL: Дата и время выставления, DATE_BILL_DATE: Дата выставления(без времени), SUM: Сумма счета, CURRENCY: Валюта,
USER	ID: Код покупателя, LOGIN: Логин, NAME: Имя, SECOND_NAME: Отчество, LAST_NAME: Фамилия, EMAIL: EMail, PERSONAL_PROFESSION: Профессия, PERSONAL_WWW: Персональный веб-сайт, PERSONAL_ICQ: Номер ICQ, PERSONAL_GENDER: Пол, PERSONAL_FAX: Номер факса, PERSONAL_MOBILE: Номер телефона, PERSONAL_STREET: Адрес, PERSONAL_MAILBOX: Почтовый ящик, PERSONAL_CITY: Город, PERSONAL_STATE: Штат, PERSONAL_ZIP: Индекс, PERSONAL_COUNTRY: Страна, WORK_COMPANY: Компания, WORK_DEPARTMENT: Отдел, WORK_POSITION: Должность, WORK_WWW: Сайт компании, WORK_PHONE: Рабочий телефон, WORK_FAX: Рабочий факс, WORK_STREET: Адрес компании,

	WORK_MAILBOX: Рабочий почтовый ящик, WORK_CITY: Город компании, WORK_STATE: Штат компании, WORK_ZIP: Индекс компании, WORK_COUNTRY: Страна компании

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sale.paysystem.pay.payment

```
sale.paysystem.pay.payment(
 payment_id,
 pay_system_id
)
```

Метод для оплаты заказа с использованием конкретной платёжной системы (вызывается после обработки ответа от платёжной системы).

Параметры

Параметр	Описание	С версии
payment_id	ID оплаты.	
pay_system_id	ID платёжной системы.	

Пример

```
var paymentId = prompt("Введите ID оплаты");
var paySystemId = prompt("Введите ID
платёжной системы");
```

```
BX24.callMethod('sale.paysystem.pay.payment'
, {"payment_id": paymentId, "pay_system_id":
paySystemId},
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
        {
            console.dir(result.data());
        }
    }
);
```

sale.paysystem.settings.payme

```
sale.paysystem.settings.payment.get(
 payment_id,
 PAY_SYSTEM_ID
)
```

Метод для получения настроек платежной системы для конкретной оплаты.

Параметры:

Параметр	Описание	С версии
payment_id	Идентификатор оплаты	
PAY_SYSTEM_ID	Идентификатор платёжной системы.	

Пример

```
var id = prompt("Введите ID оплаты");
var ps_id = prompt("ID платежной системы");
```

```
BX24.callMethod('sale.paysystem.settings.pay
ment.get', {"payment_id": id,
"pay_system_id": ps_id},
    function(result)
   {
       if(result.error())
           console.error(result.error());
       else
       {
           console.dir(result.data());
       }
   }
);
```

sale.paysystem.delete

```
sale.paysystem.delete(
 id
)
```

Метод удаляет платежную систему.

Параметры

Параметр	Описание	С версии
id	Идентификатор платежной системы	

Пример

```
var id = prompt("Введите ID");

BX24.callMethod('sale.paysystem.delete',
{"id": id},
        function(result)
        {
                if(result.error())
```

```
console.error(result.error());
                else
                {

console.info("Платежная система удалена");
                }
        }
);
```

sale.paysystem.list

```
sale.paysystem.list(params)
```

Метод для получения списка платежных систем.

Параметры

Параметр	Описание
params	Массив [dw]параметров[/dw][di]Список доступных полей идентичен полям метода sale.paysystem.add[/di] для выборки (select, order, filter) (необязательный).

Пример

```
BX24.callMethod(
        "sale.paysystem.list",
        {},
        function(result)
        {
                if(result.error())

console.error(result.error());
```

```
                    else

console.dir(result.data());
        }
);
```

sale.paysystem.update

```
sale.paysystem.update(
    id,
    fields
)
```

Метод для редактирования платежной системы.

Параметры

Параметр	Описание	С версии
id	Идентификатор платежной системы, обязательный	
fields	Набор полей (массив вида `array("поле":"значение"[, ...])`), содержащий значения, описывающие платежную систему. Доступные поля: ▪ NAME - Название системы ▪ PERSON_TYPE_ID - ID типа плательщика, обязательный. ▪ BX_REST_HANDLER - Код обработчика в системе, обязательный.	

- ACTIVE - Флаг активности платежной системы
- LOGOTYPE - Логотип платёжной системы (картинка в формате [ds]Base64[/ds][di] **Base64** — стандарт кодирования двоичных данных при помощи только 64 символов ASCII. Алфавит кодирования содержит текстово-цифровые латинские символы A-Z, a-z и 0-9 (62 знака) и 2 дополнительных символа, зависящих от системы реализации. Каждые 3 исходных байта кодируются 4 символами (увеличение на 1/3).

  Подробнее...[/di]). Поле доступно с версии **20.0.550**
- NEW_WINDOW - Поле, отвечающее за настройку "Открывать в новом окне" (по умолчанию значение `N`). Доступно с версии **20.100.0**

Пример

```
var id = prompt("Введите ID");
var name = prompt("Введите имя");

BX24.callMethod('sale.paysystem.update',
{"id": id, "fields": {"NAME" : name},
{"PERSON_TYPE_ID" : id}, {"BX_REST_HANDLER"
: handler}},
```

```
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info("Платежная система обновлёна");
                }
        }
);
```

sale.paysystem.settings.invoic

```
sale.paysystem.settings.invoice.get(
 invoice_id,
 bx_rest_handler,
 PAY_SYSTEM_ID
)
```

Метод для получения настроек платежной системы для конкретного счета.

Параметры:

Параметр	Описание	С версии
invoice_id	Идентификатор счета	
bx_rest_handler	Код рест обработчика. Если платёжных систем несколько, то вместо этого параметра используйте PAY_SYSTEM_ID.	
PAY_SYSTEM_ID	Идентификатор платёжной системы.	

Пример

```
var id = prompt("Введите ID счета");
var handler_code = prompt("Введите код рест
обработчика");

BX24.callMethod('sale.paysystem.settings.inv
oice.get', {"invoice_id": id,
"bx_rest_handler": handler_code},
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
                {

console.dir(result.data());
                }
        }
);
```

sale.paysystem.pay.invoice

```
sale.paysystem.pay.invoice(
 invoice_id,
 pay_system_id
)
```

Метод для оплаты счета с использованием конкретной платёжной системы (вызывается после обработки ответа от платёжной системы).

Параметры

Параметр	Описание	С версии
invoice_id	ID счета.	
pay_system_id	ID платёжной системы.	

Пример

```
var id = prompt("Введите ID счета");
var paySystemId = prompt("Введите ID
платёжной системы");
```

```
BX24.callMethod('sale.paysystem.pay.invoice'
, {"invoice_id": id, "pay_system_id":
paySystemId},
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
        {
            console.dir(result.data());
        }
    }
);
```

mailservice.fields

```
mailservice.fields(
)
```

Возвращает описание полей почтового сервиса.

Параметры

Без параметров

Пример

```
BX24.callMethod(
        "mailservice.fields",
        {
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {
```

```
console.info(result.data());
                    }
            }
);
```

mailservice.list

```
mailservice.list(
)
```

Возвращает список всех почтовых сервисов.

Параметры

Без параметров

Пример

```
BX24.callMethod(
        "mailservice.list",
        {
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {
```

```
console.info(result.data());
                    }
            }
);
```

mailservice.get

```
mailservice.get(
   ID
)
```

Возвращает параметры указанного почтового сервиса.

Параметры

Метод	Описание	С версии
ID	Идентификатор почтового сервиса	

Пример

```
BX24.callMethod(
        "mailservice.get",
        {
                'ID': 10
        },
        function(result)
        {
```

```
                    if(result.error())
                    {

console.error(result.error());
                    }
                    else
                    {

console.info(result.data());
                    }
          }
);
```

mailservice.add

Описание

```
mailservice.add(
   ACTIVE,
   NAME,
   SERVER,
   PORT,
   ENCRYPTION,
   LINK,
   SORT
)
```

Добавляет почтовый сервис.

Параметры

Параметр	Описание	С версии
ACTIVE	Активность сервиса (Y / N)	
NAME	Имя добавляемого почтового сервиса	
SERVER	Адрес добавляемого сервера	
PORT	Номер порта	

ENCRYPTION	Необходимость шифрования (Y / N)	
LINK	Ссылка на почтовый сервис	
SORT	Индекс сортировки	

Пример

```
BX24.callMethod(
        "mailservice.add",
        {
                'ACTIVE': 'Y',
                'NAME': 'Почтовый сервис
Yandex',
                'SERVER': 'imap.yandex.ru',
                'PORT': '993',
                'ENCRYPTION': 'Y',
                'LINK':
'https://mail.yandex.ru/',
                'SORT': '500'
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
```

```
            }
);
```

mailservice.update

Описание

```
mailservice.update(
   ID,
   ACTIVE,
   NAME,
   SERVER,
   PORT,
   ENCRYPTION,
   LINK,
   SORT
)
```

Обновляет параметры почтового сервиса.

Параметры

Параметр	Описание	С версии
ID	Идентификатор почтового сервиса	
ACTIVE	Активность сервиса (Y / N)	
NAME	Имя добавляемого почтового сервиса	

SERVER	Адрес добавляемого сервера	
PORT	Номер порта	
ENCRYPTION	Необходимость шифрования (Y / N)	
LINK	Ссылка на почтовый сервис	
SORT	Индекс сортировки	

Пример

```
BX24.callMethod(
        "mailservice.update",
        {
                'ID': 5,
                'ACTIVE': 'N',
                'NAME': 'Почтовый сервис
Yandex',
                'SERVER': 'imap.yandex.ru',
                'PORT': '993',
                'ENCRYPTION': 'Y',
                'LINK':
'https://mail.yandex.ru/',
                'SORT': '666'
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {
```

```
console.info(result.data());
                    }
            }
);
```

mailservice.delete

```
mailservice.delete(
   ID
)
```

Удаляет почтовый сервис.

Параметры

Параметр	Описание	С версии
ID	Идентификатор почтового сервиса	

Пример

```
BX24.callMethod(
        "mailservice.delete",
        {
                'ID': 8
        },
        function(result)
        {
```

```
                    if(result.error())
                    {

console.error(result.error());
                    }
                    else
                    {

console.info(result.data());
                    }
          }
);
```

department.fields

Получение списка названий полей подразделения.

Параметры

Метод не имеет параметров.

Вызов

```
BX24.callMethod('department.fields');
```

Запрос (xml для наглядности ответа)

```
https://my.bitrix24.ru/rest/department.field
s.xml?auth=7c9d8f00ea0ddd9e02cab3eb2b3bd0d1
```

Ответ

```
<response>
   <result>
        <ID>ID</ID>
        <NAME>Название подразделения</NAME>
        <SORT>Порядок сортировки</SORT>
        <PARENT>Вышестоящее
подразделение</PARENT>
        <UF_HEAD>Руководитель</UF_HEAD>
```

```
    </result>
</response>
```

department.add

Метод создает подразделение. Возможно только от имени пользователя с правами изменения структуры компании.

Параметры

Параметр	Описание
NAME	наименование подразделения (обязательный)
SORT	параметр сортировки подразделения
PARENT	идентификатор родительского подразделения.
UF_HEAD	идентификатор руководителя подразделения

Вызов

```
BX24.callMethod('department.add', {"NAME":
"Подразделение", "PARENT": 155, "UF_HEAD":
1});
```

Запрос

```
https://my.bitrix24.ru/rest/department.add.j
son?
NAME=%D0%9F%D0%BE%D0%B4%D1%80%D0%B0%D0%B7%D0
%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5&PARE
```

```
NT=155&UF_HEAD=1&auth=1b2234a7412080205f4318
e42c7298dc
```

Ответ

```
{"result":222}
```

department.update

Метод изменяет подразделение. Возможно только от имени пользователя с правами изменения структуры компании.

Параметры

Параметр	Описание
ID	идентификатор подразделения (обязательный)
NAME	наименование подразделения (обязательный)
SORT	параметр сортировки подразделения
PARENT	идентификатор родительского подразделения.
UF_HEAD	идентфикатор руководителя подразделения

Вызов

```
BX24.callMethod('department.update', {"ID":
222, "NAME": "Старое подразделение",
"PARENT": 114, "UF_HEAD": 11});
```

Запрос

```
https://my.bitrix24.ru/rest/department.updat
e.json?
```

```
ID=222&NAME=%D0%A1%D1%82%D0%B0%D1%80%D0%BE%D
0%B5%20%D0%BF%D0%BE%D0%B4%D1%80%D0%B0%D0%B7%
D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5&PA
RENT=114&UF_HEAD=11&auth=2577a0459ed8f183c99
6f44f0995ebe5
```

Ответ

```
{"result":true}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

department.get

Получение фильтрованного списка подразделений.

Параметры

Параметр	Описание
sort	поле, по которому сортируются результаты
order	направление сортировки ▪ ASC - по возрастанию ▪ DESC - по убыванию
ID	фильтр по идентификатору подразделения
NAME	фильтр по имени подразделения
PARENT	фильтр по родительскому подразделению
UF_HEAD	фильтр по руководителю подразделения

Параметры фильтрации могут принимать значение массивов.

Вызов

```
BX24.callMethod('department.get', {"ID":
222});
```

Запрос

```
https://my.bitrix24.ru/rest/department.get.j
son?
ID=222&auth=b961afd4148963c5d54ebeb17123d1fc
```

Ответ


```
{"result":
[{"ID":"222","NAME":"\u0421\u0442\u0430\u044
0\u043e\u0435
\u043f\u043e\u0434\u0440\u0430\u0437\u0434\u
0435\u043b\u0435\u043d\u0438\u0435","SORT":5
00,"PARENT":"114","UF_HEAD":"11"}],"total":1
}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

department.delete

Удаляет подразделение. Возможно только от имени пользователя с правами изменения структуры компании.

Параметры

Параметр	Описание
ID	идентификатор подразделения (обязательный)

Вызов

```
BX24.callMethod('department.delete', {"ID":
222});
```

Запрос

```
https://my.bitrix24.ru/rest/department.delet
e.json?
ID=222&auth=70a32986f1bf204dec4567147ca6a2af
```

Ответ

```
{"result":true}
```

user.fields

Перечень полей пользователей Битрикс24, который будет получен в результате выполнения метода, зависит от скоупа приложения/вебхука. Подробности о доступе к данным пользователей можно узнать в статье.

Описание

Получение списка названий полей пользователя. Метод отдаёт стандартный список полей, использование пользовательских полей не предусмотрено.

Параметры

Метод параметров не имеет.

Вызов

```
BX24.callMethod('user.fields');
```

Запрос (xml для наглядности ответа)

```
https://my.bitrix24.ru/rest/user.fields.xml?
auth=7c9d8f00ea0ddd9e02cab3eb2b3bd0d1
```

Ответ

```
<response>
        <result>
                <ID>ID</ID>
                <ACTIVE>Активность</ACTIVE>
                <EMAIL>E-Mail</EMAIL>
                <NAME>Имя</NAME>

<LAST_NAME>Фамилия</LAST_NAME>

<SECOND_NAME>Отчество</SECOND_NAME>

<PERSONAL_GENDER>Пол</PERSONAL_GENDER>

<PERSONAL_PROFESSION>Профессия</PERSONAL_PRO
FESSION>
                <PERSONAL_WWW>Домашняя
страничка</PERSONAL_WWW>
                <PERSONAL_BIRTHDAY>Дата
рождения</PERSONAL_BIRTHDAY>

<PERSONAL_PHOTO>Фотография</PERSONAL_PHOTO>

<PERSONAL_ICQ>ICQ</PERSONAL_ICQ>
                <PERSONAL_PHONE>Личный
телефон</PERSONAL_PHONE>

<PERSONAL_FAX>Факс</PERSONAL_FAX>
                <PERSONAL_MOBILE>Личный
мобильный</PERSONAL_MOBILE>

<PERSONAL_PAGER>Пейджер</PERSONAL_PAGER>
                <PERSONAL_STREET>Улица
проживания</PERSONAL_STREET>
                <PERSONAL_CITY>Город
проживания</PERSONAL_CITY>
                <PERSONAL_STATE>Область /
край</PERSONAL_STATE>
```

```
                <PERSONAL_ZIP>Почтовый
индекс</PERSONAL_ZIP>

<PERSONAL_COUNTRY>Страна</PERSONAL_COUNTRY>

<WORK_COMPANY>Компания</WORK_COMPANY>

<WORK_POSITION>Должность</WORK_POSITION>

<UF_DEPARTMENT>Подразделения</UF_DEPARTMENT>

<UF_INTERESTS>Интересы</UF_INTERESTS>

<UF_SKILLS>Навыки</UF_SKILLS>
                <UF_WEB_SITES>Другие
сайты</UF_WEB_SITES>
                <UF_XING>Xing</UF_XING>

<UF_LINKEDIN>LinkedIn</UF_LINKEDIN>
                <UF_FACEBOOK>Facebook*
</UF_FACEBOOK>

<UF_TWITTER>Twitter</UF_TWITTER>
                <UF_SKYPE>Skype</UF_SKYPE>

<UF_DISTRICT>Район</UF_DISTRICT>
                <UF_PHONE_INNER>Внутренний
телефон</UF_PHONE_INNER>
        </result>
</response>
```


* деятельность организации запрещена в Российской Федерации.

Пример

Изменить подразделение пользователя:

```
$res = CRest::call(
    'user.update',
    [
        'ID' => 4,
        'UF_DEPARTMENT' => [
            3
        ],
    ]
);
```

Добавить пользователя в рабочую группу:

```
$res = CRest::call(
    'sonet_group.user.invite',
    [
        'USER_ID' => 4,
        'GROUP_ID' => 3,
        'MESSAGE' => 'Invitation'
    ]
);
```

user.current

Перечень полей пользователей Битрикс24, который будет получен в результате выполнения метода, зависит от скоупа приложения/вебхука. Подробности о доступе к данным пользователей можно узнать в статье.

Получение информации о [dw]текущем[/dw][di]Тот чей токен вы использовали при вызове рест. Если вы используете сохраненный админский токен - то выведется администратор, если используете токен, который приходит в POST-запросе во фрейм приложения, то будет пользователь, который зашел в приложение[/di] пользователе.

Параметры

Метод параметров не имеет. Однако, сделав rest запрос с использованием данных из $_REQUEST к домену DOMAIN и добавив AUTH_ID к запросу для доступа к Битрикс24, можно узнать какой пользователь открыл страницу в контексте Битрикс24.

Вызов

```
BX24.callMethod('user.current', {},
function(res){
    alert('Привет, ' + res.data().NAME +
'!');
});
```

Запрос (xml для наглядности ответа)

```
https://my.bitrix24.ru/rest/user.current.xml
?auth=7c9d8f00ea0ddd9e02cab3eb2b3bd0d1
```

Ответ


```
<response>
        <result>
                <ID>1</ID>
                <ACTIVE>1</ACTIVE>

<EMAIL>sigurd@example.com</EMAIL>
                <NAME>Одмин</NAME>
                <LAST_NAME>
                <SECOND_NAME>
                <PERSONAL_GENDER>
                <PERSONAL_PROFESSION>
                <PERSONAL_WWW>
                <PERSONAL_BIRTHDAY>1955-04-
10T00:00:00+03:00</PERSONAL_BIRTHDAY>

<PERSONAL_PHOTO>/upload/main/80c/44169_C5_Pr
imalWaterE500CC.jpg</PERSONAL_PHOTO>
                <PERSONAL_ICQ>
                <PERSONAL_PHONE>
                <PERSONAL_FAX>
                <PERSONAL_MOBILE>
                <PERSONAL_PAGER>
                <PERSONAL_STREET>
                <PERSONAL_CITY>
                <PERSONAL_STATE>
                <PERSONAL_ZIP>

<PERSONAL_COUNTRY>0</PERSONAL_COUNTRY>
                <WORK_COMPANY>
                <WORK_POSITION>
```

```
<UF_DEPARTMENT>128</UF_DEPARTMENT>
                <UF_INTERESTS>
                <UF_SKILLS>
                <UF_WEB_SITES>
                <UF_XING>
                <UF_LINKEDIN>
                <UF_FACEBOOK>
                <UF_TWITTER>
                <UF_SKYPE>
                <UF_DISTRICT>
                <UF_PHONE_INNER>
        </result>
</response>
```

user.add

Приглашает пользователя. Возможно только от имени пользователя с правами приглашения пользователей. В случае успеха пользователю будет выслано стандартное приглашение на портал. В result возвращается id нового пользователя.

Если нужно добавить пользователя экстранета, то в полях передать необходимо передать: `EXTRANET: Y` и `SONET_GROUP_ID: [...]`. Если нужно добавить пользователя интранета, то **обязательно** передаётся: `UF_DEPARTMENT: [...]`.

Параметры

Все поля из user.fields кроме **ID**. Поле **EMAIL** - обязательное. Текст приглашения можно указать параметром **MESSAGE_TEXT**.

Вызов

```
BX24.callMethod('user.add', {"EMAIL":
"newuser@example.com"});
```

Запрос

```
https://my.bitrix24.ru/rest/user.add.json?
EMAIL=newuser@example.com&auth=1b2234a741208
0205f4318e42c7298dc
```

Ответ

```
{"result":222}
```

user.update

Обновляет данные пользователя. Возможно только от имени пользователя с правами приглашения пользователей.

Параметры

Все поля из user.fields. Поле **ID** - обязательное.

Вызов

```
BX24.callMethod('user.update', {"ID": 1,
"NAME": "Administrator"});
```

Запрос

```
https://my.bitrix24.ru/rest/user.update.json
?
ID=1&NAME=Administrator&auth=1b2234a74120802
05f4318e42c7298dc
```

Ответ

```
{"result":true}
```

user.get

Перечень полей пользователей Битрикс24, который будет получен в результате выполнения метода, зависит от скоупа приложения/вебхука. Подробности о доступе к данным пользователей можно узнать в статье.

Описание

Получение фильтрованного списка пользователей. Метод вернет всех пользователей за исключением: ботов, пользователей для e-mail, пользователей для Открытых Линий, пользователей Реплики.

Вызов

```
BX24.callMethod('user.get', {"ID": 1});
```

Запрос

```
https://my.bitrix24.ru/rest/user.get.json?
ID=1&auth=b961afd4148963c5d54ebeb17123d1fc
```

Ответ


```
{"result":
[{"ID":"1","ACTIVE":true,"EMAIL":"sigurd@exa
mple.com","NAME":"Administrator","LAST_NAME"
:"","SECOND_NAME":"","PERSONAL_GENDER":"",
```

"PERSONAL_PROFESSION":"","PERSONAL_WWW":"","PERSONAL_BIRTHDAY":"1955-04-10T00:00:00+03:00","PERSONAL_PHOTO":"\/upload\/main\/80c\/44169_C5_PrimalWaterE500CC.jpg",
"PERSONAL_ICQ":"","PERSONAL_PHONE":"","PERSONAL_FAX":"","PERSONAL_MOBILE":"","PERSONAL_PAGER":"","PERSONAL_STREET":"","PERSONAL_CITY":"","PERSONAL_STATE":"",
"PERSONAL_ZIP":"","PERSONAL_COUNTRY":"0","WORK_COMPANY":"","WORK_POSITION":"","UF_DEPARTMENT":
[128],"UF_INTERESTS":null,"UF_SKILLS":null,"UF_WEB_SITES":null,
"UF_XING":null,"UF_LINKEDIN":null,"UF_FACEBOOK":null,"UF_TWITTER":null,"UF_SKYPE":null,"UF_DISTRICT":null,"UF_PHONE_INNER":null}],"total":1}

Параметры

Параметр	Описание
sort	поле, по которому сортируются результаты
order	направление сортировки ▪ **ASC** - по возрастанию. ▪ **DESC** - по убыванию
FILTER	Дополнительно можно указывать любые параметры из user.fields для фильтрации по их значениям. Кроме основных полей, доступны дополнительные: ▪ UF_DEPARTMENT - принадлежность к структуре компании;

	▪ UF_PHONE_INNER - внутренний телефонный номер; ▪ IS_ONLINE - [Y/N] - позволяет показать только авторизованных или нет пользователей. ▪ NAME_SEARCH - быстрый поиск по персональным данным. ▪ USER_TYPE - тип пользователя. Может принимать следующие значения: **employee** - сотрудник, **extranet** - пользователь экстранета, **email** - почтовый пользователь ▪ ACTIVE - при значении *true* исключает из запроса уволенных пользователей. Параметры фильтрации могут принимать значение массивов.
ADMIN_MODE	[dw]Ключ для работы[/dw][di]`'ADMIN_MODE': 'True'` [/di] в режиме администратора, служит для получения данных о любых пользователях.

user.search

Перечень полей пользователей Битрикс24, который будет получен в результате выполнения метода, зависит от скоупа приложения/вебхука. Подробности о доступе к данным пользователей можно узнать в статье.

Метод для получения списка пользователей с ускоренным поиском по персональным данным (имя, фамилия, отчество, название подразделения, должность). Работает в двух режимах: быстро с помощью **Fulltext Index** и более медленный вариант через [dw]правый LIKE[/dw][di]USER_NAME LIKE "Текст%" - это называется правый лайк, когда поиск осуществляется только по тексту который начинается на заданную фразу, но может содержать разные окончания - такой поиск существенно быстрее чем у двухстороннего лайка "%текст%" или левостороннего "%текст" - за счет архитектуры хранения идексированных полей в БД[/di] (поддержка определяется автоматически).

Параметры

Параметр	Описание	С версии
FILTER	Массив может содержать поля в любом сочетании: ▪ **NAME** - имя ▪ **LAST_NAME** - фамилия ▪ **SECOND_NAME** - отчество ▪ **WORK_POSITION** - должность ▪ **UF_DEPARTMENT_NAME** - название подразделения	

- **USER_TYPE** - тип пользователя. Может принимать следующие значения:
  employee - сотрудник,
  extranet - пользователь экстранета,
  email - почтовый пользователь

Или **FIND** - поле которое будет искать во всех перечисленных полях (Это аналог режима старого CUser::GetList в котором можно было задать фильтр NAME_SEARCH и получить результат)

> Метод может работать либо с фильтрацией с помощью ключа FIND или со всеми другими полями. Одновременно использовать FIND и любое другое поле - нельзя.

Метод наследует поведение метода user.get все параметры из этой функции так же доступны.

Пример

Результат:

```
{
  "result": [
    {
      "ID": "2",
      "ACTIVE": true,
      "EMAIL": "m.ivshina@example.com",
```

```
      "NAME": "Мария",
      "LAST_NAME": "Ившина",
      "SECOND_NAME": "",
      "PERSONAL_GENDER": "F",
      "PERSONAL_PROFESSION": "",
      "PERSONAL_WWW":
"http://shelenkov.com/horse/events.html",
      "PERSONAL_BIRTHDAY": "1984-04-
29T02:00:00+04:00",
      "PERSONAL_PHOTO":
"http://www.hazz/upload/main/982/42-
17082203.gif",
      "PERSONAL_ICQ": "431-874-61",
      "PERSONAL_PHONE": "",
      "PERSONAL_FAX": "",
      "PERSONAL_MOBILE": "",
      "PERSONAL_PAGER": "",
      "PERSONAL_STREET": "",
      "PERSONAL_CITY": "",
      "PERSONAL_STATE": "",
      "PERSONAL_ZIP": "",
      "PERSONAL_COUNTRY": "0",
      "WORK_COMPANY": "",
      "WORK_POSITION": "IT-\"специалист\"",
      "WORK_PHONE": "+7 495 188 46 29",
      "UF_DEPARTMENT": [
        51,
        55,
        84
      ],
      "UF_INTERESTS": null,
      "UF_SKILLS": null,
      "UF_WEB_SITES": null,
      "UF_XING": null,
      "UF_LINKEDIN": null,
      "UF_FACEBOOK": null,
      "UF_TWITTER": null,
```

```
        "UF_SKYPE": null,
        "UF_DISTRICT": null,
        "UF_PHONE_INNER": "4629"
      }
    ],
    "total": 1
  }
```

user.userfield.add

Метод добавляет пользовательское поле

Параметры

<

Параметр	Описание	С версии
USER_TYPE_ID	Тип пользовательского поля. Возможны значения: ▪ `string` - строка; ▪ `integer` - целое число; ▪ `double` - число; ▪ `date` - дата; ▪ `datetime` - дата со временем; ▪ `boolean` - Да / Нет; ▪ `file` - файл; ▪ `enumeration` - список; ▪ `url` - ссылка; ▪ `address` - адрес Google карты; ▪ `money` - деньги; ▪ `iblock_section` - Привязка к разделу инфоблока; ▪ `iblock_element` - Привязка к элементу инфоблока;	

	▪ `employee` - Привязка к пользователю; ▪ `crm` - Привязка к элементу CRM; ▪ `crm_status` - Привязка к справочнику CRM. У некоторых типов есть свои дополнительные настройки.	

Пример

```
CRest::call(
   'user.userfield.add',
   [
      'fields' => [
         'FIELD_NAME' =>
'MY_TEST_FIELD_STR3',
         'USER_TYPE_ID' => 'string',
         'XML_ID' =>
'MY_TEST_FIELD_STR_xml',
         'MULTIPLE' => 'Y',
         'SHOW_FILTER' => 'Y',
         'SORT' => 100,
         'LIST_FILTER_LABEL' => 'Title',
         'LIST_COLUMN_LABEL' => 'List
Title',
         'EDIT_FORM_LABEL' => 'Title',
         'ERROR_MESSAGE' => 'Title',
         'HELP_MESSAGE' => 'Title',
         'SETTINGS' => [
            'DEFAULT_VALUE' => 'value'
         ]
      ],
   ]
);
```

Вместо
'LIST_FILTER_LABEL',
'LIST_COLUMN_LABEL',
'EDIT_FORM_LABEL',
'ERROR_MESSAGE',
'HELP_MESSAGE',
можно указать ключ 'LABEL', который заполнит все указанные выше ключи.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

user.userfield.delete

Метод удаляет пользовательское поле.

```
CRest::call(
   'user.userfield.delete',
   [
      'id' => 42,
   ]
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

user.userfield.file.get

Метод позволяет получить файл из пользовательского поля.

Пример

Есть поле `UF_USR_1604998606834` типа файл. Вызвав метод user.current можно получить файл в этом поле у текущего пользователя, где:
showUrl - это URL который покажет файл в браузере если пользователь авторизован;
downloadData - данные, которые нужно подавать на этот метод.

```
    [UF_USR_1604998606834] => Array
                (
                    [id] => 774
                    [showUrl] =>
/bitrix/services/main/ajax.php?
action=rest.file.get&SITE_ID=s1&entity=USER&
id=1&field=UF_USR_1604998606834&value=774
                    [downloadData] => Array
                        (
                            [id] => 1
                            [field] =>
UF_USR_1604998606834
                            [value] => 774
                        )

                )
```

Запрос вебхуком:

```
/rest/1/a2ebx1rfao5pq5cr/user.userfield.file
.get?
id=1&field=UF_USR_1604998606834&value=774
```

Метод возвращает файл как контент на загрузку, а не json/xml.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

user.userfield.list

Метод позволяет получить список пользовательских полей. Доступны только фильтр и сортировка.

Фильтр доступен по ключам:

- **ID**
- **FIELD_NAME**
- **USER_TYPE_ID**
- **XML_ID**
- **SORT**
- **MULTIPLE**
- **MANDATORY**
- **SHOW_FILTER**
- **SHOW_IN_LIST**
- **EDIT_IN_LIST**
- **IS_SEARCHABLE**
- **LANG**

Пример

```
CRest::call(
   'user.userfield.list',
   [
      'order' => ['ID' => 'desc'],
      'filter' => ['ID' => 42],
   ]
);
```

user.userfield.update

Метод обновляет пользовательское поле.

Внимание! Поля MULTIPLE не редактируемые.

Пример

```
CRest::call(
   'user.userfield.update',
   [
      'id' => 42,
      'fields' => [
         'LIST_FILTER_LABEL' => 'Title',
         'LIST_COLUMN_LABEL' => 'List
Title',
      ],
   ]
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

userconsent.agreement.list

Получение списка соглашений.

Без параметров.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

userconsent.agreement.text

Получение текста соглашения.

Параметры

Параметр	Описание	С версии
id	Код соглашения.	
replace	Массив: ▪ **button_caption** - название кнопки; ▪ **fields** - массив названий полей формы.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

userconsent.consent.add

Сохранение полученного согласия пользователя.

Параметры

Параметр	Описание	С версии
agreement_id	Код соглашения.	
ip	ip-адрес пользователя.	
url	Страница, на которой получено согласие.	
origin_id	Источник, к примеру, **my_contact_form**.	
originator_id	Уникальный код в рамках источника, к примеру e-mail.	

sonet_group.create

Описание

Создает группу соцсети, используя метод API CSocNetGroup::CreateGroup(), указывая владельцем группы текущего пользователя.

Запрос:

```
https://mydomain.bitrix24.ru/rest/sonet_group.create.json?auth=803f65e30340ff39703f8061c8b63a10&NAME=Test%20sonet%20group&VISIBLE=Y&OPENED=N&INITIATE_PERMS=K
```

Ответ:

```
{"result":11}
```

Параметры

Параметр	Описание	С версии

arFields	Массив параметров новой группы. Допустимые ключи массива: **NAME** - название группы (обязательное поле), **DESCRIPTION** - описание группы, **VISIBLE** - флаг Y/N - видна ли группа в списке групп, **OPENED** - флаг Y/N - открыта ли группа для свободного вступления, **KEYWORDS** - ключевые слова, **INITIATE_PERMS** - кто имеет право на приглашение пользователей в группу (обязательное поле): ▪ **A** - только владелец группы, ▪ **E** - владелец группы и модераторы группы, ▪ **K** - все члены группы. **CLOSED** - флаг Y/N - является ли группа архивной, **SPAM_PERMS** - кто имеет право на отправку сообщений в группу (обязательное поле). Значения аналогичны параметру INITIATE_PERMS. **PROJECT** - флаг Y/N - является ли группа проектом или нет. по умолчанию - не является. (С версии 18.0.0) **PROJECT_DATE_FINISH** - задаётся окончание проекта. (С версии 18.0.0) **PROJECT_DATE_START** - задаётся начало проекта. (С версии 18.0.0) **SCRUM_MASTER_ID** - если заполнен идентификатором пользователя, то этот проект станет скрамом. (С версии 22.300)	

bAutoSubscribe	Автоподписывание на созданную тему. Необязательный параметр. По умолчанию имеет значение True.	10.0.0

В случае успешного создания группы, возвращает ее ID, иначе - текст ошибки.

Примечание: Создавать экстранет-группы с помощью REST API пока нельзя.

Пример

```
// Создадим видимую и открытую для
вступления группу соцсети с именем 'Test
sonet group' с правом приглашать новых
членов группы для всех текущих членов группы

BX24.callMethod('sonet_group.create', {
        'NAME': 'Test sonet group',
        'VISIBLE': 'Y',
        'OPENED': 'N',
        'INITIATE_PERMS': 'K'
});
```

sonet_group.delete

Удаляет группу соцсети. Для осуществления операции текущий пользователь должен быть либо владельцем группы, либо иметь права администратора соцсети.

Параметры функции

Параметр	Описание
GROUP_ID	ID группы, которую необходимо удалить.

В случае успешного изменения группы возвращает **true**, иначе - текст ошибки.

Пример

```
// Удаляем группы соцсети с ID=11

BX24.callMethod('sonet_group.delete', {
        'GROUP_ID': 11
});
```

Запрос:

```
https://mydomain.bitrix24.ru/rest/sonet_group.delete.json?auth=803f65e30340ff39703f8061c8b63a10&GROUP_ID=11
```

Ответ:

```
{"result":true}
```

sonet_group.feature.access

Описание

Проверяет, имеет ли текущий пользователь право на совершение операции в группе соцсети, осуществляя вызов функции CSocNetFeaturesPerms::CurrentUserCanPerformOperation()

Запрос:

```
https://mydomain.bitrix24.ru/rest/sonet_group.feature.access.json?auth=52423d4a5f19f5f964f9b4e96a925cfa&GROUP_ID=1&FEATURE=blog&OPERATION=write_post
```

Ответ:

```
{"result":true}
```

Параметры

Параметр	Описание

GROUP_ID	ID группы соцсети.
FEATURE	Символьный код функционала.
OPERATION	Символьный код операции.

Возвращает **true**, если пользователь имеет право на совершение операции, **false** - если не имеет и ошибку в случае некорректных параметров..

Примечание: коды операций и функционала смотрите в описании метода CanPerformOperation.

Пример

```
// Получаем список групп текущего
пользователя

BX24.callMethod('sonet_group.feature.access'
, {
        'GROUP_ID': 1,
        'FEATURE': 'blog',
        'OPERATION': 'write_post'
});
```

socialnetwork.api.workgroup.g

Метод возвращает данные по рабочей группе.

Параметры

Параметр	Описание	С версии
groupId	Идентификатор группы. Обязательный параметр, целое число.	

Поля

Доступные поля:

Поле	Описание	С версии
ID	Идентификатор группы	
ACTIVE	Флаг Y/N - является ли группа активной.	
SUBJECT_ID	Код темы (обязательное	

	поле).	
SUBJECT_DATA	Поля тематики группы.	
NAME	Название группы	
DESCRIPTION	Описание группы	
KEYWORDS	Ключевые слова	
CLOSED	Флаг Y/N - является ли группа архивной,	
VISIBLE	Флаг Y/N - видна ли группа в списке групп,	
OPENED	Флаг Y/N - открыта ли группа для свободного вступления,	
PROJECT	Флаг Y/N - является ли группа проектом или нет. по умолчанию - не является. (С версии 18.0.0)	
LANDING	Флаг Y/N - является ли группой для публикацией.	
DATE_CREATE	Дата создания	
DATE_UPDATE	Дата изменения	
DATE_ACTIVITY	Дата активности	
IMAGE_ID	Идентификатор файла аватара группы	

AVATAR	URL аватара	
AVATAR_TYPES	Данные о наборе существующих типов аватаров.	
AVATAR_TYPE	Тип аватара группы (используется, если не задано значение IMAGE_ID). Допустимые значения: folder, checks, pie, bag, members.	
OWNER_ID	Идентификатор владельца группы	
OWNER_DATA	Поля владельца группы.	
NUMBER_OF_MEMBERS	Количество членов группы	
NUMBER_OF_MODERATORS	Количество модераторов группы.	
INITIATE_PERMS	Кто имеет право на приглашение пользователей в группу (обязательное поле): ▪ A - только владелец группы, ▪ E - владелец группы и модераторы группы, ▪ K - все члены группы.	

PROJECT_DATE_START	Дата начала проекта	
PROJECT_DATE_FINISH	Дата завершения проекта	
SCRUM_OWNER_ID	Идентификатор SCRUM	
SCRUM_MASTER_ID	Идентификатор SCRUM мастера	
SCRUM_SPRINT_DURATION	Длительность спринта в скраме в секундах	
SCRUM_TASK_RESPONSIBLE	Ответственный по умолчанию в скрам проекте. доступные значения: ▪ A - постановщик ▪ M - скрам-мастер	
TAGS	Теги группы	
ACTIONS	Данные о доступных текущему пользователю операциях над группой.	
USER_DATA	Данные о роли текущего пользователя в группе	

Пример

```
BX.rest.callMethod('socialnetwork.api.workgr
oup.get', {
        params: {
           groupId: 424,
           select: [ 'ID', 'NAME' ],
     },
});
```

© «Битрикс», 2001-2008, «1С-

socialnetwork.api.workgroup.li

Метод возвращает список групп.

Параметры

Параметр	Описание	С версии
filter	Соответствует параметру arFilter для метода API CSocNetGroup::getList.	
select	Массив, задающий выбираемые поля. Содержит список полей, которые должны быть возвращены методом. Если массив пустой, то выбираются поля **ID**, **SITE_ID**, **NAME**, **DESCRIPTION**, **DATE_CREATE**, **DATE_UPDATE**, **DATE_ACTIVITY**, **ACTIVE**, **VISIBLE**, **OPENED**, **CLOSED**, **SUBJECT_ID**, **OWNER_ID**, **KEYWORDS**, **IMAGE_ID**, **NUMBER_OF_MEMBERS**, **INITIATE_PERMS**, **SPAM_PERMS**, **SUBJECT_NAME**. В массиве допустимы любые поля из списка полей.	

Пример

```
BX.rest.callMethod('socialnetwork.api.workgr
oup.list', {
    filter: {
        ID: 424
    },
    select: [ 'ID', 'NAME' ]
});
```

© «Битрикс», 2001-2008, «1С-

sonet_group.get

Описание

Возвращает массив групп соцсети, каждая из которых содержит массив полей, осуществляя вызов CSocNetGroup::GetList(), при этом возвращаются только те группы, которые доступны пользователю по правам.

Запрос:

```
https://mydomain.bitrix24.ru/rest/sonet_group.get.json?auth=bbc392f317df617d02c942a78ad43aab&ORDER[NAME]=ASC&FILTER[%25NAME]=%D0%9F%D1%80%D0%BE%D0%B4
```

Ответ:


```
{"result":[{"ID":"3","SITE_ID":"s1","NAME":"\u041f\u0440\u043e\u0434\u0430\u0436\u0438",
"DESCRIPTION":"\u041c\u0430\u0440\u043a\u0435\u0442\u0438\u043d\u0433\u043e\u0432\u0430\u044f \u0433\u0440\u0443\u043f\u043f\u0430 \u043f\u043e
```

```
\u043f\u0440\u043e\u0434\u0430\u0436\u0430\u
043c",
"DATE_CREATE":"2013-11-
06T07:45:12+04:00","DATE_UPDATE":"2013-11-
06T07:45:12+04:00","ACTIVE":"Y","VISIBLE":"Y
","OPENED":"N","CLOSED":"N","SUBJECT_ID":"1"
,"OWNER_ID":"1",
"KEYWORDS":"\u043f\u0440\u043e\u0434\u0430\u
0436\u0430, \u0442\u043e\u0432\u0430\u0440,
\u043c\u0430\u0440\u043a\u0435\u0442\u0438\u
043d\u0433, \u0440\u044b\u043d\u043e\u043a",
"NUMBER_OF_MEMBERS":"1","DATE_ACTIVITY":"201
3-11-
06T07:45:12+04:00","SUBJECT_NAME":"\u0420\u0
430\u0431\u043e\u0447\u0438\u0435
\u0433\u0440\u0443\u043f\u043f\u044b",
"IMAGE":"https:\/\/cdn.bitrix24.ru\/b211545\
/socialnetwork\/ba9\/ba9533b38f60ade077b64f0
6a60d7082\/2.jpg","IS_EXTRANET":"Y"}],"total
":1}
```


Параметры

Параметр	Описание
ORDER	Соответствует параметру `arOrder` метода CSocNetGroup::GetList().
FILTER	Соответствует параметру `arFilter` метода CSocNetGroup::GetList().
IS_ADMIN	При передаче Y, проверяется, является ли текущий пользователь администратором соцсети и, если да, отключается проверка прав при выборке групп.

Возвращает те же поля, что и CSocNetGroup::GetList(), за исключением INITIATE_PERMS, SPAM_PERMS и IMAGE_ID (вместо последнего возвращается поле IMAGE, с полями файла, соответствующего IMAGE_ID).

Пример

```
        // Получим список всех доступных
групп соцсети, название которых начинается с
подстроки "Прод", отсортированный по
названию в алфавитном порядке

        BX24.callMethod('sonet_group.get', {
        'ORDER': {
        'NAME': 'ASC'
        },
        'FILTER': {
        '%NAME': 'Прод'
        }
        });
```

Смотрите также

- Использование методов REST

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sonet_group.setowner

Метод изменяет владельца группы. Может быть запущен либо администратором социальной сети, либо текущим владельцем группы.

Параметры: **GROUP_ID** - идентификатор группы, владелец которой меняется и **USER_ID** - идентификатор нового владельца.

В случае успеха возвращает *true*.

Пример

```
BX24.callMethod('sonet_group.setowner', {
    'GROUP_ID': 11,
   'USER_ID': 2
});
```

sonet_group.update

Изменяет параметры группы соцсети, используя метод API CSocNetGroup::Update() Для осуществления операции текущий пользователь должен быть либо владельцем группы, либо иметь права администратора соцсети.

Получает в параметрах все поля, необходимые для работы метода CSocNetGroup::Update(), а также **GROUP_ID** - ID группы, которую необходимо изменить.

В случае успешного изменения группы, возвращает ее ID, иначе - текст ошибки.

Пример

```
// Изменяем название группы соцсети с ID=11
на 'Test sonet group XXX'

BX24.callMethod('sonet_group.update', {
        'GROUP_ID': 11,
        'NAME': 'Test sonet group XXX'
});
```

Запрос:

```
https://mydomain.bitrix24.ru/rest/sonet_grou
p.update.json?
```

```
auth=803f65e30340ff39703f8061c8b63a10&GROUP_
ID=11&NAME=Test%20sonet%20group%20XXX
```

Ответ:

```
{"result":11}
```

sonet_group.user.add

Добавляет пользователей в качестве участников рабочей группы (без приглашения и подтверждения).

Для осуществления операции текущий пользователь должен иметь права администратора соцсети.

В случае, если добавляется пользователь экстранет, группа станет доступной в экстранете (если не была до этого доступна).

Параметры вызова

Параметр	Описание
GROUP_ID	ID рабочей группы.
USER_ID	ID пользователя (или массив ID), который/ые добавляются в участники группы.

Пример

```
// Добавляем пользователей с ID=10 и 21 в
группу соцсети с ID=15

BX24.callMethod('sonet_group.user.add', {
    GROUP_ID: 15,
    USER_ID: [ 10, 21 ]
});
```

sonet_group.user.delete

Позволяет удалить пользователя/пользователей из рабочей группы.

Для осуществления операции текущий пользователь должен иметь права администратора соцсети.

В случае, если текущая роль пользователя - владелец группы, он не может быть удален из группы этим методом.

Параметры вызова

Параметр	Описание
GROUP_ID	ID рабочей группы.
USER_ID	ID пользователя (или массив ID), который/ые удаляются из группы.

Пример

```
// Удаляем пользователей с ID=10 и 21 из
группы соцсети с ID=15

BX24.callMethod('sonet_group.user.delete', {
    GROUP_ID: 15,
    USER_ID: [ 10, 21 ]
});
```

sonet_group.user.get

Возвращает массив участников группы соцсети, осуществляя вызов CSocNetUserToGroup::GetList(), при этом проверяются права на доступ текущего пользователя к группе.
Метод не возвращает неактивных пользователей (уволенных сотрудников).

Каждый из участников представляет собой массив с полями:

- **USER_ID** - ID пользователя
- **ROLE** - роль пользователя в группе:
  - **SONET_ROLES_OWNER (A)** - владелец,
  - **SONET_ROLES_MODERATOR (E)** - модератор,
  - **SONET_ROLES_USER (K)** - пользователь

Параметры функции

Параметр	Описание
ID	ID группы, участников которой необходимо получить.

Пример

```
// Получаем список участников группы соцсети
с ID=15
```

```
BX24.callMethod('sonet_group.user.get', {
        'ID': 15
});
```

Запрос:

```
https://mydomain.bitrix24.ru/rest/sonet_grou
p.user.get.json?
auth=67df5afc8ce59732e4a21ed3e336979f&ID=15
```

Ответ:

```
{"result":[{"USER_ID":"1","ROLE":"A"}]}
```

sonet_group.user.groups

Возвращает массив групп соцсети текущего пользователя, осуществляя вызов CSocNetUserToGroup::GetList().

Каждая из групп представляет собой массив с полями:

- **GROUP_ID** - ID группы соцсети
- **GROUP_NAME** - название группы соцсети
- **ROLE** - роль пользователя в группе:
  - **SONET_ROLES_OWNER (A)** - владелец,
  - **SONET_ROLES_MODERATOR (E)** - модератор,
  - **SONET_ROLES_USER (K)** - пользователь

Пример

```
// Получаем список групп текущего
пользователя

BX24.callMethod('sonet_group.user.groups',
{});
```

Запрос:

```
https://mydomain.bitrix24.ru/rest/sonet_grou
p.user.groups.json?
```

```
auth=52423d4a5f19f5f964f9b4e96a925cfa
```

Ответ:


```
{"result":
[{"GROUP_ID":"1","GROUP_NAME":"\u041c\u0430\
u0440\u043a\u0435\u0442\u0438\u043d\u0433
\u0438
\u0440\u0435\u043a\u043b\u0430\u043c\u0430",
"ROLE":"A"},
{"GROUP_ID":"3","GROUP_NAME":"\u041f\u0440\u
043e\u0434\u0430\u0436\u0438","ROLE":"A"},
{"GROUP_ID":"5","GROUP_NAME":"\u041e\u0442\u
0434\u044b\u0445","ROLE":"A"},
{"GROUP_ID":"7","GROUP_NAME":"\u0422\u0435\u
0445\u043d\u043e\u043b\u043e\u0433\u0438\u04
38","ROLE":"A"},
{"GROUP_ID":"9","GROUP_NAME":"\u0424\u0440\u
0438\u043b\u0430\u043d\u0441","ROLE":"A"},
{"GROUP_ID":"13","GROUP_NAME":"Test sonet
group","ROLE":"A"},
{"GROUP_ID":"15","GROUP_NAME":"Test sonet
group","ROLE":"A"}]}
```

sonet_group.user.invite

Описание

Выполняет приглашение пользователей в группу соцсети от лица текущего пользователя, при этом проверяются права на доступ текущего пользователя к группе.

Возвращает массив ID пользователей, успешно приглашенных в группу.

Запрос:

```
https://mydomain.bitrix24.ru/rest/sonet_group.user.invite.json?auth=52423d4a5f19f5f964f9b4e96a925cfa&GROUP_ID=15&USER_ID=3&MESSAGE=Invitation
```

Ответ:

```
{"result":["3"]}
```

Параметры

Параметр	Описание
GROUP_ID	ID группы, в которую происходит приглашение.
USER_ID	ID пользователя (или массив ID пользователей), который приглашается (которые приглашаются) в группу
MESSAGE	Текст приглашения.

Пример

```
// Приглашаем пользователя с ID=3 в группу
соцсети с ID=15

BX24.callMethod('sonet_group.user.invite', {
        'GROUP_ID': 15,
        'USER_ID': 3,
        'MESSAGE': 'Invitation'
});
```

sonet_group.user.request

Описание

Отправляет запрос текущего пользователя на вступление в группу соцсети, при этом проверяются права на доступ текущего пользователя к группе. Если группа открыта для свободного вступления, пользователь сразу становится ее участником.

Возвращает **true**, если запрос произошел успешно, или текст ошибки.

Запрос:

```
https://mydomain.bitrix24.ru/rest/sonet_group.user.request.json?auth=52423d4a5f19f5f964f9b4e96a925cfa&GROUP_ID=17&MESSAGE=Request
```

Ответ:

```
{"result":true}
```

Параметры

Параметр	Описание
GROUP_ID	ID группы, в которую отправляется запрос
MESSAGE	Текст запроса

Пример

```
// Отправляем запрос на вступление в группу
с ID=15

BX24.callMethod('sonet_group.user.request',
{
        'GROUP_ID': 17,
        'MESSAGE': 'Request'
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

sonet_group.user.update

Позволяет изменить роль пользователя/пользователей в рабочей группе.

Для осуществления операции текущий пользователь должен иметь права администратора соцсети.

В случае, если текущая роль пользователя - владелец группы, она не может быть изменена этим методом.

Параметры вызова

Параметр	Описание
GROUP_ID	ID рабочей группы.
USER_ID	ID пользователя (или массив ID), которому/которым меняется роль.
ROLE	код новой роли участника группы (доступны значения E - модератор и K - участник).

Пример

```
// Меняем роли пользователей с ID=10 и 21 в
группе соцсети с ID=15 на модераторов

BX24.callMethod('sonet_group.user.update', {
```

```
    GROUP_ID: 15,
    USER_ID: [ 10, 21 ],
    ROLE: 'E'
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События рабочих групп

Событие	Описание	в
ONSONETGROUPADD	Событие вызывается после добавления новой рабочей группы. Прокси к событию OnSocNetGroupAdd.	
ONSONETGROUPUPDATE	Событие вызывается после изменения рабочей группы. Прокси к событию onSocnetGroupUpdate	
ONSONETGROUPDELETE	Вызывается в момент удаления рабочей группы. Прокси к событию OnSocNetGroupDelete	
ONSONETGROUPSUBJECTADD	Событие вызывается после создания темы рабочих групп. Прокси к событию OnSocNetGroupSubjectAdd.	
ONSONETGROUPSUBJECTUPDATE	Событие вызывается после изменения темы рабочих групп. Прокси к событию OnSocNetGroupSubjectUpdate	
ONSONETGROUPSUBJECTDELETE	Вызывается перед удалением темы рабочих групп.. Прокси к событию OnSocNetGroupSubjectDelete	

© «Битрикс», 2001-2008, «1С-

Процессы

Подробнее о процессах можно прочитать в документации по модулю rpa

rpa.type.*

Метод	Описание	С версии
`rpa.type.get({id: number})`	Отдает информацию о процессе по его ID. **Параметр:** ▪ `id` - ID процесса **Ответ:** `{ "type": { "id":1, "title":"Название процесса", "image":"list", "createdBy":1, "settings":[], "permissions":[ { "id":"1", "entity":"TYPE", "entityId":"1", "accessCode":"UA", "action":"ITEMS_CREATE",`	

```
"permission":"X"
            }
        ]
    }
}
```

- `title` - название процесса
- `image` - это идентификатор иконки из списка
- `createdBy` - идентификатор пользователя, который создал процесс
- `settings` - набор настроек процесса
- `permissions` - набор настроек прав доступа этого процесса

`rpa.type.list({select: ?[] = ['*'], order: ?{} = null, filter: ?{} = null, start: ?number = 0})`

Метод вернет массив процессов с их полями.

```
{
    "types": [
        {},
        {}
    ]
}
```

Параметры:

- `select` - массив полей для вывода. По умолчанию выводятся все
- `order` - список для сортировки, где ключ - поле, а значение - `ASC` или `DESC`
- `filter` - список для фильтрации

	▪ `start` - сдвиг для постраничной навигации	
`rpa.type.add({fields: {}})`	Метод создаст новый процесс и вернет в ответе данные, аналогичные ответу на запрос `rpa.type.get`. `fields` - список с полями процесса: ▪ `fields[title]` - название процесса (обязательное) ▪ `fields[image]` - изображение процесса из списка ▪ `fields[settings]` - список с произвольным набором настроек процесса ▪ `fields[permissions]` - массив с правами доступа к этому процессу **Важно! Автоматические сценарии (создание стадий, роботов и полей по умолчанию) не будут запущены при создании процесса через `rest`.** **Важно! В запросе обязательно должны быть указаны права доступа на изменение процесса** **Пример запроса** Этот запрос создаст новый процесс под названием "Мой процесс". Все пользователи смогут создавать элементы этого процесса. Только пользователь с ID 1 сможет	

	менять настройки этого процесса
	
	``
	{
	"fields": {
	"title": "Мой процесс",
	"image": "list",
	"permissions": [
	{
	"accessCode": "UA",
	"permission": "X",
	"action": "ITEMS_CREATE"
	},
	{
	"accessCode": "U1",
	"permission": "X",
	"action": "MODIFY"
	},
	]
	}
	}
	``	

`rpa.type.update({id: number, fields: {}})`	Метод обновит процесс с `id` и вернет в ответе данные, аналогичные ответу на запрос `rpa.type.get`. **Параметры**: ▪ `id` - идентификатор процесса ▪ `fields` - список с полями процесса. Полностью аналогичен набору из метода `rpa.type.add`	
`rpa.type.delete({id: number})`	Метод удаляет процесс. **Параметры**:	

	▪ `id` - идентификатор процесса	

Стадии

Подробнее о стадиях можно прочитать в документации по модулю rpa.

rpa.stage.*

Метод	Описание
`rpa.stage.get({id: number})`	Отдает информацию о стадии по е **Параметр**: ▪ *id* - ID стадии Ответ: { "id": 1, "name": "Запуск", "code": "", "color": "22B9FF", "sort": 1000, "semantic": null, "typeId": 1, "isFirst": true, "isSuccess": false, "isFail": false, "tasks": [{ "title": "Задание "robotType": "RpaApproveActivity", "robotName": "A43555_78925_98855_46118", "canAppendRespons true,

```
            "users": [
                {
                    "id": "U1
                    "entityId
                    "name": "
                    "photoSrc
                    "url":
"\\/company\\/personal\\/user
                    "entityTy
"users"
                },
                {
                    "id": "U4
                    "entityId
                    "name": "
                    "photoSrc
                    "url":
"\\/company\\/personal\\/user
                    "entityTy
"users"
                }
            ]
        }
    ],
    "robotsCount": 0,
    "possibleNextStages": [1,
    "permissions": {
        "droppable": true,
        "canMoveFrom": true
    }
}
```


- `name` - название стадии
- `code` - символьный код. Можн использовать как внешний идентификатор
- `color` - HEX-код цвета стади символов
- `sort` - индекс сортировки
- `semantic` - код семантики ста Может быть либо `SUCCESS`, ли
- `typeId` - идентификатор проц
- `isFirst` - вычисляемое поле. это первая стадия процесса
- `isSuccess` - вычисляемое пол если эта стадия является усп

	▪ `isFail` - вычисляемое поле. эта стадия является провалы
	▪ `tasks` - массив заданий стади запись состоит имеет следую структуру:
	  ▪ `title` - заголовок задан
	  ▪ `robotType` - тип задания принимать одно из знач
	  ▪ `RpaApproveActivit` утвердить или откл
	  ▪ `RpaMoveActivity` - передвинуть
	  ▪ `RpaRequestActivit` информации
	  ▪ `RpaReviewActivity` ознакомиться с инс
	  ▪ `robotName` - имя активит
	  ▪ `users` - массив участник (для отрисовки в канбан стадии)
	▪ `robotsCount` - вычисляемое п Количество роботов на стади
	▪ `possibleNextStages` - массив идентификаторов стадий, в к можно перенести элемент. Не используется
	▪ `permissions` - набор разреше канбана).
	  ▪ `droppable` - в эту стадии быть перемещены элеме
	  ▪ `canMoveFrom` - из этой ст быть перемещены элеме

`rpa.stage.listForType({typeId: number, offset: ?number})`	Метод вернет список стадий проце отсотированный в порядке сортир финальными стадиями в конце. **П**

- `typeId` - идентификатор проц
  Обязательное.
- `start` - сдвиг для постраничн
  навигации

В информации о каждой стадии бу
основные данные, без `tasks`, robo
`possibleNextStages`, `permissions`

```
{
    "stages": [
        {
            "id": 1,
            "name": "Запуск",
            "code": "",
            "color": "22B9FF"
            "sort": 1000,
            "semantic": null,
            "typeId": 1,
            "isFirst": true,
            "isSuccess": fals
            "isFail": false,
        }
    ]
}
```

`rpa.stage.add({fields: {}})`

Метод создаст новую стадию и вер
ответе данные, аналогичные отве
запрос `rpa.stage.get`. **Параметр**

`fields` - список с полями стадии

- `fields[name]` - название ста
  (обязательное)
- `fields[typeId]` - идентифик
  процесса (обязательное)
- `fields[code]` - символьный к
- `fields[color]` - цвет стадии
  HEX (6 символов)
- `fields[sort]` - идентификат
  сортировки
- `fields[semantic]` - код сема
  стадии. Может быть либо `SUC`

	FAIL **У процесса может быть только успешная стадия**
`rpa.stage.update({id: number, fields: {}})`	Метод обновит стадию с `id` и верн данные, аналогичные ответу на за `rpa.stage.get`. **Параметры**: ▪ `id` - идентификатор стадии ▪ `fields` - список с полями ста Полностью аналогичен набор метода `rpa.stage.add`, но бе **У процесса всегда должна быт успешная стадия. Изменить се успешной стадии нельзя**
`rpa.stage.delete({id: number})`	Метод удаляет стадию. **Параметр** ▪ `id` - идентификатор стадии **У процесса всегда должна быт успешная стадия. Удалить успе стадию нельзя**

Элементы

Подробнее об элементах можно прочитать в документации по rpa.

Так как элементы каждого процесса хранятся в отдельной таблице - идентификаторы элементов разных процессов будут совпадать.

Поэтому во все методы необходимо передавать идентификатор процесса `typeId`

rpa.item.*

Метод	Описание
`rpa.item.get({typeId: number, id: number})`	Отдает информацию об элементе с идентификатором `id` процесса с идентификатором `typeId`. **Параметр** ▪ `typeId` - идентификатор процес ▪ `id` - идентификатор элемента Ответ `{ "item": { "id": 43, "stageId": 4, "previousStageId": 3, "name": "Название элемен "typeId": 1, "createdBy": 1, "updatedBy": 1, "createdTime": "19.03.20 13:07:39",`

```
        "updatedTime": "23.03.20
18:34:05",
        "movedTime": "23.03.2020
18:34:05",
        "detailUrl":
"/rpa/item/1/43/",
        "movedBy": 1,
        "UF_RPA_1_NAME": "Назван
элемента",
        "tasksCounter": 0,
        "tasksFaces": {
            "completed": [
                1
            ],
            "running": [

            ],
            "all": [
                1
            ]
        },
        "users": {
            "1": {
                "id": "1",
                "name": "Anton",
                "secondName": nu
                "lastName": "",
                "title": null,
                "workPosition":
null,
                "fullName": "Ant
                "link":
"/company/personal/user/1/"
            }
        }
    }
}
```


Здесь

- `stageId` - идентификатор стадии
которой находится элемент
- `previousStageId` - идентификат
предыдущей стадии элемента
- `name` - название элемента
- `typeId` - идентификатор процесс
- `createdBy` - id пользователя,
создавшего элемент

- `updatedBy` - id пользователя, изменившего элемент
- `movedBy` - id пользователя, изменившего стадию элемента
- `createdTime` - время создания элемента
- `updatedTime` - время изменения элемента
- `movedTime` - время изменения ст элемента
- `detailUrl` - ссылка на карточку элемента
- `tasksCounter` - количество зада на элементе для пользователя
- `tasksFaces` - информация для отрисовки последовательности ответственных при утверждении
  - `completed` - кто выполнил задание
  - `running` - кто выполняет задание
  - `all` - все участники
- `users` - агрегированная информа о всех пользователях, имеющих отношение к элементу. Список, ключ - ид пользователя
  - `id` - идентификатор
  - `name` - имя
  - `secondName` - отчество
  - `lastName` - фамилия
  - `title` - обращение
  - `workPosition` - должность
  - `fullName` - форматированно имя
  - `link` - ссылка на профиль
- `UF_RPA_...` - значения пользовательских полей

	▪ значения множественных п отдаются в виде массива
	▪ значение поля типа "файл" отдаются в виде списка
	  ▪ `id` - идентификатор
	  ▪ `url` - ссылка на файл портале
	  ▪ `urlMachine` - ссылка н файл для приложения

`rpa.item.list({typeId: number, order: ?{} = null, filter: ?{} = null, start: ?number = 0})`	Метод вернет массив элементов проц с идентификатором `typeId`. **Парамет** ▪ `typeId` - идентификатор процесс ▪ `order` - список для сортировки, ключ - поле, а значение - `ASC` ил `DESC` ▪ `filter` - список для фильтрации Ключи для фильтрации по пользовательским полям должны быть в `UPPER_CASE`, остальные - `camelCase`. Примеры фильтров н ▪ `start` - сдвиг для постраничной навигации В ответе будут только основные поля элементов, без данных о заданиях и пользователях элементов `{ "items": [ {}, {} ] }` **Примеры фильтра** 1. **Найти элементы, на которых есть невыполненные задания текущего пользователя**

```
{
    "filter": {
        "tasks": "has_tasks"
    }
}
```

Чтобы найти элементы, на котор
нет заданий пользователя, надо
передать значение `no_tasks`

2. **Найти элементы, обновленны
пользователем с
идентификатором 4**

```
{
    "filter": {
        "=updatedBy": "4"
    }
}
```

3. **Найти элементы, обновленны
или сдвинутые пользователе
идентификатором 4**

```
{
    "filter": {
        "logic": "or",
        "0": {
            "=updatedBy": "4
        },
        "1": {
            "=movedBy": "4"
        }
    }
}
```

4. **Найти элементы, у которых
заполнено пользовательское
поле с кодом `UF_RPA_1_STRING`**

```
{
    "filter": {
        "!=UF_RPA_1_STRING":
    }
}
```

5. **Найти элементы, которые бы
созданы, изменены и сдвину
период с 19.03 по 22.03**

```
{
    "filter": {
        ">createdTime":"202(
19T02:00:00+02:00",
        ">movedTime":"2020-(
19T02:00:00+02:00",
        ">updatedTime":"202(
19T02:00:00+02:00",
        "<createdTime":"202(
22T02:00:00+02:00",
        "<movedTime":"2020-(
22T02:00:00+02:00",
        "<updatedTime":"202(
22T02:00:00+02:00"
    }
}
```

6. **Найти элементы, которые бы
или созданы, или изменены и
сдвинуты в период с 19.03 по
22.03**

```
{
    "filter": {
        "logic":"OR",
        "0":{

">createdTime":"2020-03-
19T02:00:00+02:00",
               "
<createdTime":"2020-03-
22T02:00:00+02:00"
        },
        "1":{

">movedTime":"2020-03-
19T02:00:00+02:00",
               "
<movedTime":"2020-03-
22T02:00:00+02:00"
        },
        "2":{

">updatedTime":"2020-03-
19T02:00:00+02:00",
               "
<updatedTime":"2020-03-
22T02:00:00+02:00"
        }
```

	`}` `}`
`rpa.item.add({typeId: number, fields: ?{})`	Метод создает новый элемент процес идентификатором `typeId`. **Параметр** ▪ `typeId` - идентификатор процесс ▪ `fields` - значения пользователь полей элемента. Все остальные будут проигнорированы. Не обязательный параметр **После создания элемента роботы будут запущены автоматически** Метод вернет результат аналогичный вызову метода `rpa.item.get` на толь что созданном элементе Чтобы загрузить файл, в качестве значения пользовательского поля необходимо передать массив, где пер элемент - это имя файла, а второй - э закодированный в base64 контент фа

`rpa.item.update({typeId: number, id: number, fields: {})`	Метод обновляет элемент с
	идентификатором `id` процессса с
	идентификатором `typeId`. **Параметр**
	
	▪ `typeId` - идентификатор процесс
	▪ `id` - идентификатор элемента
	▪ `fields` - значения пользователь
	полей элемента. Обязательный
	параметр
	
	  ▪ `fields[stageId]` -
	  идентификатор стадии
	  ▪ `fields[UF_RPA_...]` - знач
	  пользовательских полей
	
	**Загрузить новый файл вместо
	старого (не множественное поле)**

Чтобы заменить файл в не множественном поле, просто загрузит новый файл. Старый будет удален автоматически

```
{
    "fields": {
        "UF_RPA_1_1585069397": [
            "myfile.pdf",
"...base64_encoded_file_content.
        ]
    }
}
```

Удалить значение пользовательск поля типа файл

Для этого достаточно передать пустую строку (`''`) вместо значения

Оставить значение не множественного поля типа файл изменений

Самый простой вариант - не добавлят `fields` ключ с этим полем. Но если на и передать, и не изменить, то в качес значения надо передать список, где п ключу `id` будет идентификатор файла

```
{
    "fields": {
        "UF_RPA_1_1585069397": {
            "id": 433
        }
    }
}
```

Если в `id` передать отличное от текущего значения, то значение п обнулится и файл будет стерт

Работа с множественным полем т файл

Значение множественного поля - это массив. Каждый элемент массива

	подчинается тем же правилам, что и , не множественных значений. **Частичная перезапись значения множественного поля типа файл** Например, сейчас в множественном п типа файл находится значение `[12, 44]` Необходимо оставить файла `12` и `44`, вместо `255` загрузить новый Запрос должен выглядеть следующим образом: `{ "fields": { "UF_RPA_1_1585069397": [ { "id": 12 }, { "id": 44 }, [ "myNewFile.pd "...base64_encoded_file_content. ] ] } }`
`rpa.item.delete({typeId: number, id: number)`	Метод удалит элемент. **Параметры**: ▪ `typeId` - идентификатор процесс ▪ `id` - идентификатор элемента.
`rpa.item.getTasks({{typeId: number, id: number})`	Метод вернет данные о текущих зада элемента с идентификатором `id` проц с идентификатором `typeId`. **Парамет** ▪ `typeId` - идентификатор процесс ▪ `id` - идентификатор элемента

Пример ответа


```
{
    "tasks": [
        {
            "id": "93",
            "title": "asdf",
            "description": "",
            "userId": 1,
            "data": {
                "participantJoin
"or",
                "isMine": true,
                "controls": {
                    "BUTTONS": [
                        {
                            "TYP
"submit",

"TARGET_USER_STATUS": 3,
                            "NAM
"complete",
                            "VAL
"Y",
                            "TEX
"Сохранить",
                            "COL
"3bc8f5"
                        }
                    ]
                },
                "type":
"RpaRequestActivity",
                "url":
"/rpa/task/id/93/",
                "fieldsToShow":
null,
                "fieldsToSet": [
                    "Название"
                ],
                "users": [
                    {
                        "id": 1,
                        "status"
                    }
                ]
            },
            "itemClassName":
"BX.Rpa.Timeline.Task",
```

```
            "users": {
                "1": {
                    "id": "1",
                    "name": "Ant
                    "secondName"
"",
                    "lastName":
"Gorbylev",
                    "title": nul
                    "workPositio
"",
                    "fullName":
"Anton Gorbylev",
                    "link":
"/company/personal/user/1/"
                }
            }
        }
    ]
}
```

Задания

Набор методов для работы с заданиями.

rpa.task.*

Метод	Описание	С версии
`rpa.task.delete({typeId: number, stageId: number, robotName: string})`	Метод удалит робота с именем `robotName` с процесса с идентификатором `typeId` со стадии с идентификатором `stageId`. **Параметры**: ▪ `typeId` - идентификатор процесса ▪ `stageId` - идентификатор стадии ▪ `robotName` - имя робота	
`rpa.task.addUser({typeId: number, stageId: number, robotName: string, userValue: string})`	Метод добавит пользователя к существующему заданию.	

	Параметры: ▪ `typeId` - идентификатор процесса ▪ `stageId` - идентификатор стадии ▪ `robotName` - имя робота ▪ `userValue` - строка с пользователем формата "Имя Фамилия [ид пользователя]"	
`rpa.task.do({})`	Метод выполнения задания. Набор параметров зависит от задания.	

Настройки видимости полей

Набор методов для работы с настройками видимости полей

Больше о настройках можно прочитать в документации по rpa

Кроме пользовательских полей, можно управлять видимостью системных полей в карточке канбана.

Системные поля имеют следующие коды:

- `id` - id элемента
- `createdBy` - кто создал
- `updatedBy` - кто изменил
- `movedBy` - кто сменил стадию
- `createdTime` - время создания
- `updatedTime` - время изменения
- `movedTime` - время смены стадии

rpa.fields.*

Метод	Описание
`rpa.fields.getSettings({typeId: number, stageId: ?number})`	Метод вернет полный набор настроек видимости полей на стадии с идентификатором

	`stageId` процесса с идентификатором `typeId`. **Параметры**: - `typeId` - идентификатор процесса - `stageId` - идентификатор стадии. Необязательное поле, по умолчанию 0 (общие настройк Пример ответа `{ "fields": { "kanban": "id": true, "UF_RPA_1_NAME": t }, "create": "UF_RPA_1_NAME": t } } }`
`rpa.fields.setSettings({typeId: number, fields: ?[], stageId: ?number})`	Метод устанавливает полный набор настро видимости полей на стадии с идентификатором `stageId` процесса с идентификатором `typeId`. **Параметры**: - `typeId` - идентификатор процесса - `stageId` - идентификатор

	стадии. Необязательное поле, по умолчанию 0 (общие настройк ▪ `fields` - массив настройками видимости полей Если передать пустой `fields`, то все настро будут стерты Пример запроса `{ "typeId": 1, "fields": { "kanban": "createdBy", "UF_RPA_1_NAME" ] } }`
`rpa.fields.setSettings({typeId: number, fields: ?[], stageId: ?number})`	Метод устанавливает полный набор настро видимости полей на стадии с идентификатором `stageId` процесса с идентификатором `typeId`. **Параметры**: ▪ `typeId` - идентификатор процесса ▪ `stageId` - идентификатор стадии. Необязательное поле, по умолчанию 0 (общие настройк

	▪ `fields` - массив настройками видимости полей Если передать пустой `fields`, то все настро будут стерты Пример запроса `{ "typeId": 1, "fields": { "kanban": "createdBy", "UF_RPA_1_NAME" ] } }` Метод вернет результ аналогичный запросу `rpa.fields.getSetti`
`rpa.fields.setVisibilitySettings({typeId: number, visibility: string, fields: ?[], stageId: ?number})`	Метод изменяет настройки видимости `visibility` полей `fields` для процесса идентификатором `typeId` на стадии с идентификатором `stageId`. Остальные настройки при этом н изменяются. **Параметры**: ▪ `typeId` - идентификатор процесса ▪ `visibility` - идентификатор видимости, для

которого меняют
настройки

- `stageId` - идентификатор стадии. Необязательное поле, по умолчанию 0 (общие настройк
- `fields` - массив полями, для которых необходимо поменять настро

Метод необходимо использовать, когда надо изменить настройки видимости только одного типа

Пример запроса

```
{
    "typeId": 1,
    "visibility":
"kanban",
    "fields": [
        "createdBy

"UF_RPA_1_NAME"
    ]
}
```

Метод вернет результ
аналогичный запросу
`rpa.fields.getSetti`

Записи таймлайна

Набор методов для работы с записями таймлайна

Подробнее о структуре таблицы читайте в документации по rpa.

rpa.timeline.*

Метод	Опис
`rpa.timeline.listForItem({typeId: number, itemId: number, start: ? number = 0})`	Метод возвращает массив запи идентификатором `itemId` проц `itemId`, отсортированные по у (сверху самые новые). **Парам** ▪ `typeId` - идентификатор г ▪ `itemId` - идентификатор з ▪ `start` - сдвиг для постран Пример ответа `{ "timeline": [ { "id": 321, "typeId": 24, "itemId": 10, "createdTime": 26T20:28:57+02:00", "userId": 1, "title": "Выпо "description": "action": "tas "isFixed": fal`

```
            "data": {
                "item": {
                    "name"
                },
                "scope": "
                "stageFrom
                    "id":
                    "name"
                },
                "stageTo":
                    "id":
                    "name"
                },
                "fields":
                    {
                        "n
                        "t
                    }
                ],
                "task": {
                    "ID":
                    "USER_
                    "WORKF
"5e7cf3e91ef413.27314358",
                    "ACTIV
                    "ACTIV
"A79985_79846_49104_50661"
                    "NAME"
                    "DESCR
                    "PARAM
                        "D


"Bitrix\\Rpa\\Integration\

                        ],
                        "T
"/rpa/task/id/#ID#/",
                        "A





                        ],
                        "F


"UF_RPA_24_STRING_MANDATOR
                        ],
                        "R
```

```
                        "A
                        "F

                        ]
                    },
                    "USERS
                        1
                    ],
                    "INCOM

                    ]
                }
            },
            "createdTimest
            "users": {
                "1": {
                    "id":
                    "name"
                    "secon
                    "lastN
                    "title
                    "workP
                    "fullN
                    "link"
"/company/personal/user/1/
                }
            }
        },
    ]
}
```


- `id` - идентификатор запис
- `typeId` - идентификатор г
- `itemId` - идентификатор э
- `createdTime` - время созд
- `userId` - идентификатор г инициировал действие
- `title` - заголовок записи
- `description` - текстовое о
- `action` - код типа действи
- `isFixed` - флаг прикрепле
- `data` - сериализованные д действии и связанных сущ записи. В зависимости от содержать разный набор это:

	▪ `item` - данные об эле ▪ `item[name]` - на выполнения дей ▪ `scope` - код источник одно из следующих з ▪ `manual` - вручну ▪ `task` - при выпо ▪ `automation` - ро ▪ `rest` - приложе ▪ `stageFrom` - данные выполнения действи ▪ `id` - идентифик ▪ `name` - название ▪ `stageTo` - данные о н изменена при выпол ▪ `fields` - массив данн которых были измен ▪ `name` - код поля ▪ `title` - заголов ▪ `task` - данные о зада выполнено при выпо ▪ `users` - данные о пользов отношение к действию

`rpa.timeline.updateIsFixed({id: number, isFixed: string})`	Метод обновляет флаг прикреп
	▪ `id` - идентификатор запис
	▪ `isFixed` - флаг прикрепле будет прикреплена, иначе
	
	Метод вернет данные об обнов
	
	`{`
	`"timeline": {`

```
        "id": 322,
        ...
    }
}
```

`rpa.timeline.add({typeId: number, itemId: number, fields: {}})`

Метод создаст новую запись та
идентификатором `itemId` проц
`typeId`. **Параметры**:

- `typeId` - идентификатор г
- `itemId` - идентификатор з
- `fields` - поля записи.
  - `title` - заголовок за
  - `description` - описа
    использовать html)

Этот метод позволяет изменять
`description`

Пример выполнения

```
{
    "timeline": {
        "id": 325,
        "typeId": 24,
        "itemId": 10,
        "createdTime": "20
        "userId": 1,
        "title": "rest upd
        "description": "<h
        "action": false,
        "isFixed": false,
        "data": {
            "scope": "rest
        },
        "createdTimestamp"
        "users": {
            "1": {
                "id": "1",
                "name": "A
                "secondNam
                "lastName"
                "title": n
                "workPosit
                "fullName"
```

	`"link": "/`
	`}`
	`}`
	`}`
	`}`

`rpa.timeline.update({id: number, fields: {}})`	Метод обновит запись таймлай
	Параметры:
	▪ `id` - идентификатор запис
	▪ `fields` - поля записи.
	  ▪ `title` - заголовок за
	  ▪ `description` - описа использовать html)
	
	Этот метод позволяет изменять `description`
	
	Этот метод позволяет измен которые были созданы этим помощью приложения.

`rpa.timeline.delete({id: number})`	Метод удалит запись таймлайн **Параметры**: ▪ `id` - идентификатор запис **Этот метод позволяет удаля которые были созданы этим помощью приложения.**

Комментарии

Набор методов для работы с комментариями в таймлайне элементов.

По факту комментарии - это те же записи таймлайна, но с другим отображением и возможностью редактирования пользователем.

Данные о комментариях можно получить методом `rpa.timeline.listForItem` - этот метод возвращает все записи, в том числе комментарии.

rpa.comment.*

Метод	Описание

`rpa.comment.add({typeId: number, itemId: number, fields: {}})`	Метод создаст новый комментарий в таймлайне элемента с идентификатором `itemId` процесса с идентификатором `typeId`. **Параметры**:
	▪ `typeId` - идентификатор процесса
	▪ `itemId` - идентификатор элемента
	▪ `fields` - поля комментария.
	  ▪ `description` - описание записи (можно использовать html и BB-code)
	  ▪ `files` - массив прикрепленных файлов, где каждый элемент - это массив с именем и закодированным в base64 содержимым

Пример запроса

```
{
    "typeId": 24,
    "itemId": 10,
    "fields": {
        "description": "Упоминание
пользователя с ид 1
[USER=1]Anton[/USER]",
        "files": [
            [
                "document.pdf",
"...base64_decoded_content..."
            ]
        ]
    }
}
```

Результат

```
{
    "comment": {
        "id": 350,
        "createdTime": "2020-03-
27T16:00:59+02:00",
        "isFixed": false,
        "typeId": 24,
        "itemId": 10,
        "action": "comment",
        "description": "Упоминание
пользователя с ид 1
[USER=1]Anton[/USER]",
        "userId": 1,
        "title": "Комментарий",
        "data": {
            "files": [
                15
            ]
        },
        "createdTimestamp":
1585317659000,
        "htmlDescription":
"Упоминание пользователя с ид 1 <a
class=\"blog-p-user-name\"
id=\"bp_K6r6vvp7\"
href=\"/company/personal/user/1/\"
bx-tooltip-user-id=\"1\">Anton
Gorbylev</a>  ",
        "textDescription":
```

	"Упоминание пользователя с ид 1 Anton",
	"users": {
	"1": {
	"id": "1",
	"name": "Anton",
	"secondName": "",
	"lastName": "",
	"title": null,
	"workPosition": "",
	"fullName": "Anton",
	"link": "/company/personal/user/1/"
	}
	}
	}
	}

`rpa.comment.update({id: number, fields: {}})`	Метод обновит запись таймлайна с идентификатором `id`. **Параметры**:
	- `id` - идентификатор комментария
	- `fields` - поля записи.
	  - `description` - описание записи (можно использовать html и BB-code)
	  - `files` - массив прикрепленных файлов, где каждый элемент - это массив с именем и закодированным в base64 содержимым
	
	Этот метод позволяет изменять только поля `title` и `description`
	
	Этот метод позволяет изменить только те комментарии, которые были добавлены этим же пользователем.
	
	Чтобы дописать новый файл, в качестве записи о старом файле необходимо передать список, где по ключу `id` будет

	идентификатор прикрепленного к этому комментарию файла. Для загрузки новых надо также передать массив с именем и содержимым файла в base64 Пример `{ "typeId": 24, "itemId": 10, "fields": { "description": "Упоминание пользователя с ид 1 [USER=1]Anton[/USER]", "files": [ { "id": 15 }, [ "another_document.pdf", "...base64_decoded_content..." ] ] } }`
`rpa.comment.delete({id: number})`	Метод удалит комментарий с идентификатором `id`. **Параметры**: - `id` - идентификатор записи **Этот метод позволяет удалять только те комментарии, которые были добавлены этим же пользователем**

Введение

Чтобы создать полноценный сайт через REST или внести изменения в существующий, вы должны понимать, что REST копирует логику работы пользователя. То есть, например, чтобы начать менять блок, нужно его сначала добавить, если его нет на странице, а чтобы изменения увидели свет, нужно опубликовать страницу. Но давайте коротко по пунктам.

Итак, "чтобы создать сайт" нужно всего ничего:

1. Создать сайт, или выбрать один из существующих. На выходе вы будете так или иначе иметь идентификатор сайта, с которым работаете. (Методы для работы с сайтом.)
2. Теперь дело за страницей. Аналогичным образом создаем страницу или выбираем из существующих. (Методы для работы со страницей)
3. Блоки. Блоки это молекулы сайтов (ноды – атомы). Вы должны хорошо понимать, что такое блок, и что такое его манифест. Вы можете работать с блоками в понятиях страницы (добавлять, перемещать, удалять) с помощью данных методов. А вот работать с конкретным блоком с помощью данных методов.
4. Не забывайте, после всех действий страницу нужно опубликовать.
5. Если вам не хватает блоков, вы всегда можете зарегистрировать новые.

Это – необходимый вам костяк для работы с блоками. Методов конечно существует намного больше и они достаточно точечны, чтобы охватить максимум ваших кейсов.

Успехов!

© «Битрикс», 2001-2008, «1С-Битрикс» 2008-2022

ORM запросов

В некоторых методах (как правило, они носят название ***.getList**) даны лишь куцые примеры применения ORM, но не дана глобальная расшифровка допустимых операндов. Чтобы лучше понимать такие запросы, а самое главное, научиться составлять новые, важно понимать, что под капотом данных методов работает ORM Битрикс, который подробно объяснен в Курсе разработчика Bitrix Framework и в другой документации. Но чтобы не пугать читателя абсолютно новым языком формирования запросов, достаточно глянуть сводку на данной странице, из которой станут понятны принципы применения тех или иных ключей и операндов.

© «Битрикс», 2001-2008, «1С-

Работа с типами сайтов, скоупы

Типы сайтов

Сайты могут быть следующих типов.

- Основные:
  - PAGE (от Home Page) - обычные сайты.
  - STORE - магазины.
  - SMN - сайты, использующиеся в разделе Сайты24 в административном разделе в БУС.
- Дополнительные:
  - KNOWLEDGE –– базы знаний.
  - GROUP –– базы знаний групп соц.сети.

На данный момент не поддерживается расширение типов.

Скоупы

Помимо разделительной функции на уровне компонентов существует также разграничение по правам, получившее название скоупов.

Если вы работаете с основными типами, ничего делать не нужно. Если с дополнительными, перед работой нужно установить скоуп. В случае rest сделать это можно, передав дополнительным параметром **scope**.

Пример

В примере дан метод получения списка страниц, но правило распространяется на любой другой метод, в том числе на работу с правами и изменениями сущностей.

```
BX24.callMethod(
    'landing.landing.getList',
    {
        params: {
            select: [
                'ID', 'TITLE'
            ],
            filter: {
                TITLE: '%услуги%',
                SITE_ID: 205
            },
            order: {
                ID: 'DESC'
            }
        },
        scope: 'knowledge'
    },
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

Полезные ссылки

Несколько ссылок, которые могут быть полезны разработчикам при работе с функционалом Сайты

1. Презентация функционала Сайты24
   - Презентация
   - Видео
2. Функционал Сайтов с точки зрения пользователя.
   - Клиентская документация
3. Не стесняйтесь предлагать свои идеи, мы вас внимательно читаем.
   - Сайт Идея!
4. Обсуждение функционала.
   - Группа по последним новостям с разработки модуля (для доступа требуется регистрация)
5. На сайте поставщика шаблонов вы можете выбрать понравившийся вам код для использования в новых блоках или полностью новых шаблонах для сайтов и страниц.

© «Битрикс», 2001-2008, «1С-

Поля сущности Сайт

Поле	Описание	Чтение	Запись
ID	Идентификатор сайта. Создается автоматически и уникален в рамках БД.	Да	Нет
CODE	Уникальный символьный код сайта. В облачном Б24 никак не фигурирует. **Обязательное поле**.	Да	Да
ACTIVE	Активность сайта: Y / N.	Да	Нет
TYPE	Тип сайта. (PAGE – обычный сайт, STORE – магазин). Тип сайта. Обязательно ознакомьтесь с понятием скоупов	Да	Да
DELETED	Флаг [dw]удаленной страницы[/dw] [di]Маркированные как удаленные сущности не фигурируют ни в одном запросе.	Да	Да

	Система их как бы не видит. Через REST вы сможете добраться до таких сущностей только явно указав в фильтрации DELETED=Y.[/di]: Y / N.		
TITLE	Название сайта. **Обязательное поле**.	Да	Да
XML_ID	Внешний ключ для нужд разработчика. Не используется сервисом	Да	Да
DESCRIPTION	Произвольное описание сайта. Выводится в списке сайтов.	Да	Да
DOMAIN_ID	Если поле передано и оно пустое, то создастся случайный домен в зоне bitrix24.site. Иначе будет зарегистрирован новый указанный портал в зоне bitrix24.site, если он не занят. По умолчанию создается случайный домен.	Нет	Да
DOMAIN_NAME	Домен сайта.	Да	Да
LANDING_ID_INDEX	ID страницы, которая указана как главная сайта.	Да	Да

LANDING_ID_404	ID страницы, которая указана как ошибка 404 сайта.	Да	Да
TPL_ID	Идентификатор шаблона представления.	Да	Да
LANG	Поле актуально только для облачного Битрикс24, в нем содержится идентификатор языка, по которому будет отдаваться сайт (изменяться некоторые языковые фразы, например, в оформлении заказа).	Да	Да
CREATED_BY_ID	Идентификатор создавшего пользователя.	Да	Нет
MODIFIED_BY_ID	Идентификатор изменившего пользователя.	Да	Нет
DATE_CREATE	Дата создания.	Да	Нет
DATE_MODIFY	Дата изменения.	Да	Нет
PUBLIC_URL	Публичный URL сайта.	Да	Нет
PREVIEW_PICTURE	Превью-картинка сайта (его главной страницы).	Да	Нет

Дополнительные поля сущности Сайт

На чтение поля получаются через метод landing.site.getadditionalfields.

Обратите внимание! Ниже указаны коды полей, которые для записи в сущность необходимо указывать в массиве с ключом ADDITIONAL. Например, `ADDITIONAL_FIELDS: {UP_SHOW: 'Y'}`.

Поле	Описание
THEME_CODE	Цветовая палитра. Описание тем
THEME_CODE_TYPO	Настройки шрифтов.
B24BUTTON_CODE	Идентификатор виджета на сайт. Передает JS-путь до виджета. Например, https://cdn.bitrix24.com/crm/loader_2_ibikv
B24BUTTON_COLOR	Цвет виджета, может принимать значения (использовать основной цвет сайта), butto (использовать цвет из настроек виджета)
UP_SHOW	Показывать ли кнопку Вверх: Y / N.
	Фоновая картинка
BACKGROUND_USE	Использовать функционал: Y / N
BACKGROUND_PICTURE	Путь до изображения.

BACKGROUND_POSITION	Позиционирование: center (растянуть), re (замостить).
BACKGROUND_COLOR	Цвет фона.
	Аналитика
YACOUNTER_USE	Использовать Яндекс.Метрику: Y / N.
YACOUNTER_COUNTER	Код счетчика Яндекс.Метрики.
GACOUNTER_USE	Использовать Google Analytics: Y / N.
GACOUNTER_COUNTER	Код счетчика Google Analytics.
GACOUNTER_SEND_CLICK	Отправлять данные о кликах по кнопкам и ссылкам в Google Analytics.
GACOUNTER_SEND_SHOW	Отправлять данные о просмотре блоков страницы в Google Analytics.
GTM_USE	Использовать Google Tag Manager.
GTM_COUNTER	Код Google Tag Manager.
	Карты
GMAP_USE	Использовать Google Карты: Y / N.
GMAP_CODE	Код Google Карты.
	Представление сайта
VIEW_USE	Использовать представление: Y / N.
VIEW_TYPE	Тип представления: no (без представлени ltr (отступ сверху и по бокам), all (отступ с всех сторон).
	Robots.txt

ROBOTS_USE	Показывать свой Robots.txt: Y / N.
ROBOTS_CONTENT	Контент robots.txt
Пользовательский HTML	
HEADBLOCK_USE	Использовать: Y / N.
HEADBLOCK_CODE	Блок HEAD, произвольный html.
Пользовательский CSS	
CSSBLOCK_USE	Использовать: Y / N.
CSSBLOCK_CODE	Произвольный CSS-код.
CSSBLOCK_FILE	Ссылка на CSS-файл.

landing.site.add

```
landing.site.add(fields)
```

Метод для добавления Сайта. Возвращает ID созданного сайта или ошибку.

Параметры

Параметр	Описание	С версии
fields	Поля сущности	

Пример

```
BX24.callMethod(
        'landing.site.add',
        {
                fields: {
                        TITLE: 'My first
site!',
                        CODE: 'firstsite',
                        DOMAIN_ID:
```

```
'my.bitrix24.site'
                }
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.site.addFolder

```
landing.site.addFolder(
    siteId,
    fields
)
```

Метод добавляет папку в сайт.

Параметры

Параметр	Описание	С версии
siteId	Идентификатор сайта. **Внимание:** Требуются права на запись в указанный сайт.	
fields	Поля папки: ▪ ACTIVE – активность папки (Y/N). По умолчанию создается не активной;	

	▪ TITLE – заголовок (наименование) папки; ▪ CODE – символьный код папки (часть URL страницы папки). По умолчанию транслитерируется из названия папки.	

Пример

```
BX24.callMethod(
    'landing.site.addFolder',
    {
        siteId: 1817,
        fields: {
            TITLE: 'Новая папка'
        }
    },
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.site.delete

```
landing.site.delete(id)
```

Метод для удаления Сайта. Возвращает *true* в случае успеха, или ошибку.

Параметры

Метод	Описание	С версии
id	id сущности.	

Пример

```
BX24.callMethod(
        'landing.site.delete',
        {
                id: 206
        },
        function(result)
        {
                if(result.error())
```

```
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.site.fullExport

```
landing.site.fullExport(
    id,
    params
)
```

Метод экспортирует сайт и всего его страницы в специальный массив, который требуется для работы метода landing.demos.register.

Параметры

Параметр	Описание	С версии
id	Идентификатор сайта.	
params	Опциональный массив, ключи которого описаны в примере ниже.	

Пример

```
BX24.callMethod(
   'landing.site.fullExport',
   {
      id: 326,
      params: {
         edit_mode: 'Y',
         //scope: 'knowledge',//передаем
scope, если требуется (подробнее)
         hooks_disable:
['B24BUTTON_CODE'],//коды доп.полей, которые
не надо экспртировать
         code: 'myfirstsite',//симв.код
сайта
         name: 'Сайт автомастерской',//имя
сайта (страницы)
         description: 'Сайт для вашего
автосервиса. Под капотом все самое
нужное.',//описание сайта
         preview:
'http://site.ru/preview.jpg',//основная
превью-картинка для списка шаблонов (реком.
280x115)
         preview2x:
'http://site.ru/preview.jpg',//увеличенная
превью-картинка (рекомен. 560x230)
         preview3x:
'http://site.ru/preview.jpg'//ретина-размер
превью картинки (рекомен. 845x345)
      }
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
```

```
        {
            console.info(result.data());
        }
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

landing.site.getadditionalfields

```
landing.site.getadditionalfields(id)
```

Метод для получения дополнительных полей сайта. Возвращает дополнительные поля сайта или ошибку.

Параметры

Метод	Описание	С версии
id	идентификатор сайта	

Пример

```
BX24.callMethod(
        'landing.site.getadditionalfields',
        {
                id: 205
        },
        function(result)
        {
                if(result.error())
```

```
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.site.getFolders

```
landing.site.getFolders(
    siteId,
    filter
)
```

Метод получает папки сайта.

Параметры

Параметр	Описание	С версии
siteId	Идентификатор сайта. **Внимание:** Требуются права на запись в указанный сайт.	
filter	Опциональный фильтр. Может принимать поля: ▪ ACTIVE – активность папки (Y/N). По умолчанию создается не активной;	

	▪ DELETED – папка удалена (Y/N). По умолчанию возвращаются не удаленные папки; ▪ PARENT_ID – идентификатор родительской папки; ▪ TITLE – заголовок папки; ▪ INDEX_ID – идентификатор индексной страницы папки; ▪ CODE – символьный код папки; ▪ CREATED_BY_ID – идентификатор пользователя, создавшего папку; ▪ MODIFIED_BY_ID – идентификатор пользователя, изменившего папку;	

Пример

```
BX24.callMethod(
    'landing.site.getFolders',
    {
        siteId: 1817,
        filter: {
            TITLE: 'Измененная папка'
        }
    },
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
```

```
            }
            else
            {
                console.info(result.data());
            }
        }
);
```

landing.site.getList

```
landing.site.getList(params)
```

Метод для получения списка сайтов.

Обратите внимание, помеченные как удаленные страницы не фигурируют в выборках. Чтобы получить их явно, необходимо при фильтрации указать ключ `DELETED` со значением Y или N.

Параметры

Метод	Описание	С версии
params	Опциональный массив, с опциональными ключами: **select**, **filter**, **order**, **group**, которые содержат значения таблицы основных полей сущности.	

Пример

```
BX24.callMethod(
        'landing.site.getList',
        {
                params: {
                        select: [
                                'ID',
'TITLE', 'DOMAIN.DOMAIN'
                        ],
                        filter: {
                                TITLE:
'%услуги%'
                        },
                        order: {
                                ID: 'DESC'
                        }
                }
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

© «Битрикс», 2001-2008, «1С-

landing.site.getPreview

```
landing.site.getPreview(
    id
)
```

Метод возвращает URL изображения-превью сайта (превью индексной страницы). Сайт должен быть доступен на чтение.

Параметры

Параметр	Описание	Версия
id	Идентификатор сайта.	

Пример

```
BX24.callMethod(
    'landing.landing.getPreview',
    {
        id: 1817
    },
    function(result)
```

```
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.site.getPublicUrl

```
landing.site.getPublicUrl(
    $id
)
```

Метод возвращает полный URL сайта (сайтов).

Параметры

Параметр	Описание	С версии
id	Идентификатор сайта. Может быть также массивом идентификаторов, в таком случае в ответе будет массив URL сайтов.	

Пример

```
BX24.callMethod(
    'landing.site.getPublicUrl',
```

```
    {
       id: [752, 751]
    },
    function(result)
    {
       if(result.error())
       {
          console.error(result.error());
       }
       else
       {
          console.info(result.data());
       }
    }
);
```

landing.site.markDelete

```
landing.site.markDelete(
   id
)
```

Метод помечает сайт как удаленный.

Параметры

Параметр	Описание	Версия
id	Идентификатор сайта.	

Пример

```
BX24.callMethod(
   'landing.site.markDelete',
   {
      id: 1688
   },
   function(result)
   {
      if(result.error())
```

```
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.site.markFolderDelete

```
landing.site.markFolderDelete(
   id
)
```

Метод помечает папку как удаленную (помещенную в корзину).

Параметры

Параметр	Описание	Версия
id	Идентификатор папки. Права на доступ к сайту папки должны быть на удаление.	

Пример

```
BX24.callMethod(
    'landing.site.markFolderDelete',
    {
        id: 737
    },
```

```
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.site.markFolderUnDele

```
landing.site.markFolderUnDelete(
   id
)
```

Метод помечает папку как не удаленную (возвращает из корзины).

Параметры

Параметр	Описание	Версия
id	Идентификатор папки. Права на доступ к сайту папки должны быть на удаление.	

Пример

```
BX24.callMethod(
    'landing.site.markFolderUnDelete',
    {
        id: 737
    },
```

```
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.site.markUnDelete

```
landing.site.markUnDelete(
   $id
)
```

Метод помечает сайт как не удаленный.

Параметры

Параметр	Описание	Версия
ID	Идентификатор страницы	

Пример

```
BX24.callMethod(
   'landing.site.markUnDelete',
   {
      id: 1688
   },
   function(result)
   {
      if(result.error())
```

```
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.site.publication

```
landing.site.publication(
   id
)
```

Метод публикует сайт (и все его страницы).

Параметры

Параметр	Описание	Версия
$id	Идентификатор сайта.	

```
BX24.callMethod(
   'landing.site.publication',
   {
      id: 1688
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
```

```
        else
        {
            console.info(result.data());
        }
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

landing.site.publicationFolder

```
landing.site.publicationFolder(
    folderId
)
```

Метод публикует папку сайта. Должны быть права на публикацию сайта папки.

Параметры

Параметр	Описание	Версия
folderId	Идентификатор папки.	

Пример

```
BX24.callMethod(
    'landing.site.publicationFolder',
    {
        id: 737
    },
    function(result)
```

```
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.site.unpublic

```
landing.site.unpublic(
   $id
)
```

Метод снимает с публикации сайт (и все его страницы).

Параметры

Параметр	Описание	Версия
$id	Идентификатор сайта.	

```
BX24.callMethod(
   'landing.site.unpublic',
   {
      id: 1688
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
```

```
        else
        {
            console.info(result.data());
        }
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

landing.site.unPublicFolder

```
landing.site.unPublicFolder(
    folderId
)
```

Метод снимает с публикации папку сайта. Должны быть права на публикацию сайта папки.

Параметры

Параметр	Описание	Версия
folderId	Идентификатор папки.	

Пример

```
BX24.callMethod(
    'landing.site.unPublicFolder',
    {
        id: 737
    },
    function(result)
```

```
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.site.update

```
landing.site.update(id, fields)
```

Метод для изменения сайта. Возвращает *true* в случае успеха, или ошибку.

Параметры

Метод	Описание	С версии
id	Идентификатор сущности	
fields	Изменяемые поля сущности.	

Пример

```
BX24.callMethod(
        'landing.site.update',
        {
                id: 206,
                fields: {
                        TITLE: 'My second
```

```
site!'
                }
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.site.updateFolder

```
landing.site.updateFolder(
    siteId,
    folderId,
    fields
)
```

Метод изменяет папку в сайте.

Параметры

Параметр	Описание	С версии
siteId	Идентификатор сайта. **Внимание:** Требуются права на запись в указанный сайт.	
folderId	Идентификатор папки в сайте.	
fields	Поля папки:	

	▪ ACTIVE – активность папки (Y/N). По умолчанию создается не активной; ▪ TITLE – заголовок (наименование) папки; ▪ INDEX_ID –– идентификатор страницы внутри папки, которую требуется сделать индексной страницей папки; ▪ CODE – символьный код папки (часть URL страницы папки). По умолчанию транслитерируется из названия папки.	

Пример

```
BX24.callMethod(
    'landing.site.updateFolder',
    {
        siteId: 1817,
        folderId: 736,
        fields: {
            TITLE: 'Измененная папка'
        }
    },
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
```

```
                console.info(result.data());
            }
        }
);
```

Поля сущности Страница

Поля	Описание	Чтение	Запись
ID	Идентификатор страницы. Создается автоматически и уникален в рамках БД.	Да	Нет
CODE	Уникальный символьный код страницы. Добавляется к адресу сайта, если это не главная страница. **Обязательное поле**.	Да	Да
RULE	Регулярное выражение для вывода страницы по маске. Например, правило `section/([\d]+)` для страницы в корне сайта будет отвечать всем страницам вида `/section/<n>/`, где <n> - любое число..	Да	Нет
ACTIVE	Активность	Да	Нет

	страницы: Y / N.		
DELETED	Флаг [dw]удаленной страницы[/dw] [di]Маркированные как удаленные сущности не фигурируют ни в одном запросе. Система их как бы не видит. Через REST вы сможете добраться до таких сущностей только явно указав в фильтрации DELETED=Y.[/di]: Y / N.	Да	Да
TITLE	Название страницы. **Обязательное поле**.	Да	Да
XML_ID	Внешний ключ для нужд разработчика. Не используется сервисом.	Да	Да
DESCRIPTION	Произвольное описание страницы. Выводится в списке страниц.	Да	Да
SITE_ID	Идентификатор сайта, к которому привязана страница. **Обязательное поле**.	Да	Да
CREATED_BY_ID	Идентификатор пользователя	Да	Нет

	создавшего страницу.		
MODIFIED_BY_ID	Идентификатор пользователя изменившего страницу.	Да	Нет
DATE_CREATE	Дата создания.	Да	Нет
DATE_MODIFY	Дата изменения.	Да	Нет
SITEMAP	Страница присутствует в карте сайта (`/sitemap.xml`), Y / N.	Да	Да
FOLDER_ID	Идентификатор папки, где содержится страница.	Да	Да
TPL_ID	Идентификатор шаблона представления.	Да	Да
TPL_CODE	Идентификатор шаблона партнерского решения, на основе которого был создан сайт. Например, bitrix.eshop.	Да	Нет

Дополнительные поля сущности Страница

Одинаковые поля с Сайтом у страницы имеют более высокий приоритет.

На чтение поля получаются через метод landing.landing.getadditionalfields.

Обратите внимание! Ниже указаны коды полей, которые для записи в сущность необходимо указывать в массиве с ключом ADDITIONAL. Например, `ADDITIONAL_FIELDS => [METAROBOTS_INDEX => Y]`.

Поле	Описание	Чтение	Запи
THEME_CODE	Цветовая палитра. Описание тем	Да	Да
THEME_CODE_TYPO	Настройки шрифтов.	Да	Да
Предпросмотр в социальных сетях			
METAOG_TITLE	Заголовок. тег og:title.	Да	Да
METAOG_DESCRIPTION	Описание, тег og:description.	Да	Да
METAOG_IMAGE	Изображение, тег og:image.	Да	Да

Мета-теги			
METAMAIN_USE	Задать мета-теги: Y / N.	Да	Да
METAMAIN_TITLE	Заголовок, тег title.	Да	Да
METAMAIN_DESCRIPTION	Описание, тег description.	Да	Да
METAMAIN_KEYWORDS	Ключевые слова, тег keywords.	Да	Да
Фоновая картинка			
BACKGROUND_USE	Использовать функционал: Y / N	Да	Да
BACKGROUND_PICTURE	Путь до изображения.	Да	Да
BACKGROUND_POSITION	Позиционирование: center (растянуть), repeat (замостить).	Да	Да
BACKGROUND_COLOR	Цвет фона.	Да	Да
Аналитика			
YACOUNTER_USE	Использовать Яндекс.Метрику: Y / N.	Да	Да
YACOUNTER_COUNTER	Код счетчика Яндекс.Метрики.	Да	Да
GACOUNTER_USE	Использовать Google Analytics: Y / N.	Да	Да
GACOUNTER_COUNTER	Код счетчика Google Analytics.	Да	Да

GACOUNTER_SEND_CLICK	Отправлять данные о кликах по кнопкам и ссылкам в Google Analytics.	Да	Да
GACOUNTER_SEND_SHOW	Отправлять данные о просмотре блоков страницы в Google Analytics.	Да	Да
GTM_USE	Использовать Google Tag Manager.	Да	Да
GTM_COUNTER	Код Google Tag Manager.	Да	Да
Пользовательский HTML			
HEADBLOCK_USE	Использовать: Y / N.	Да	Да
HEADBLOCK_CODE	Блок HEAD, произвольный html.	Да	Да
Пользовательский CSS			
CSSBLOCK_USE	Использовать: Y / N.	Да	Да
CSSBLOCK_CODE	Произвольный CSS-код.	Да	Да
CSSBLOCK_FILE	Ссылка на CSS-файл.	Да	Да
Представление страницы			
VIEW_USE	Использовать представление: Y / N.	Да	Да
VIEW_TYPE	Тип	Да	Да

	представления: no (без представления), ltr (отступ сверху и по бокам), all (отступ со всех сторон).		
Прочее			
METAROBOTS_INDEX	Индексировать страницу в поисковых системах: Y / N.	Да	Да

Специальные страницы

Описание

При создании страницы через мастер в манифесте могут быть прописаны специальные страницы. Например, страница корзины. Для таких страниц предусмотрены специальные метки, чтобы в теле страницы можно было ссылаться на определенную системную, не зная ее полного адреса и даже ID.

Если для данного маркера существует соответствующая страница в рамках текущего сайта, адрес будет подставлен вместо макера, в ином случае маркер будет проигнорирован.

Пример

```
<a href="#system_cart">Корзина</a>
//обратите внимание, маркер намеренно
ставится без закрывающегося символа решетки
#
```

Существующие на данный момент маркеры спец.страниц:

- #system_mainpage – главная страница
- #system_catalog – главная страница каталога
- #system_personal – персональный раздел
- #system_cart – корзина
- #system_order – оформление заказа
- #system_payment – страница оплаты (непосредственно процессинг оплаты)

- #system_compare – страница сравнения

Методы

Метод	Описание

landing.syspage.set	Метод устанавливает для сайта специальную страницу. Если парам не передан, то соответствующий ти страницы, не сама страница, будет удален. Метод не возвращает резул
	`BX24.callMethod(`
	`  'landing.syspage.set',`
	`  {`
	`  id: 1390,// ИД сайта`
	`  type: 'personal',// тип страницы`
	`  lid: 8593// ИД страницы, которая в рамках сайта будет считаться данного типа`
	`  },`
	`  function(result)`
	`  {`
	`  if(result.error())`
	`  {`
	`  console.error(result.error());`
	`  }`
	`  else`
	`  {`
	`  console.info(result.data());`
	`  }`
	`  }`
	`)`

landing.syspage.get	Метод возвращает список страниц с которые установлены как специалы
	`BX24.callMethod(`
	`  'landing.syspage.get',`

```
   {
      id: 1390,// ИД сайта
      active: true// Если true,
вернутся только активные страни
сайта (по умолчанию все)
   },
   function(result)
   {
      if(result.error())
      {

console.error(result.error());
      }
      else
      {

console.info(result.data());
      }
   }
);
```

landing.syspage.getSpecialPage	Метод возвращает адрес специальн страницы сайта. В примере показан получить ссылку на страницу персонального раздела сайта.

```
BX24.callMethod(
   'landing.syspage.getSpecialP
   {
      siteId: 1391,// ИД сайта
      type: 'personal',// тип
специальной страницы
      additional: {// необязате
массив доп. параметров, которые
будут добавлены к URL
         SECTION: 'private'
      }
   },
   function(result)
   {
      if(result.error())
      {

console.error(result.error());
      }
      else
      {

console.info(result.data());
```

	`}`
	`}`
	`);`

landing.syspage.deleteForLanding	Метод удаляет все упоминания стра
	как специальной.
	`BX24.callMethod(`
	`'landing.syspage.deleteForLandi`
	`{`
	`id: 8613 // ИД страницы`
	`},`
	`function(result)`
	`{`
	`if(result.error())`
	`{`
	`console.error(result.error());`
	`}`
	`else`
	`{`
	`console.info(result.data());`
	`}`
	`}`
	`);`

landing.syspage.deleteForSite	Метод удаляет все специальные
	страницы сайта.
	`BX24.callMethod(`
	`'landing.syspage.deleteForSi`
	`{`
	`id: 1391 // ИД сайта`
	`},`
	`function(result)`
	`{`
	`if(result.error())`
	`{`
	`console.error(result.error());`
	`}`
	`else`
	`{`
	`console.info(result.data());`
	`}`

```
    }
);
```

Цветовые темы

В настоящий момент в системе присутствуют цветовые темы, представленные ниже. Стоит учитывать, что в шаблоне присутствует как тема визуального отображения (формы, цвета), так и тема шрифтов. Сайт по шаблону всегда создается в определенной цветовой теме и в определенной шрифтовой теме. Как правило, они совпадают (шрифтовая тема это небольшой дополнительный файл). Затем, когда пользователь меняет визуальную тему, шрифтовая тема внутри остается неизменной и может меняться только программно разработчиком. Это намеренное поведение системы.

Код темы	Название темы
1construction	Янтарный
2business	Фиолетово-синий
3corporate	Малибу
accounting	Желто-зеленый
agency	Пастельный красный
app	Умеренный бирюзовый
architecture	Закат солнца
charity	Желтый
consulting	Светло-зелёное море
courses	Умеренный аквамариновый
event	Амарантовый

gym	Индиго Крайола
lawyer	Карминно-розовый
music	Дикий арбуз
photography	Нефтяной
real-estate	Оранжевый закат
restaurant	Малиновый
shipping	Красный
spa	Цитрусовый
travel	Киноварь
wedding	Клюквенный

Примечание. Названия тем с цифрами в начале - не описка. Часть тем именно так и называется.

landing.landing.add

```
landing.landing.add(fields)
```

Метод для добавления Страницы. Возвращает LID созданной страницы или ошибку.

Параметры

Параметр	Описание	С версии
fields	Поля сущности	

Пример

```
BX24.callMethod(
   'landing.landing.add',
   {
      fields: {
         TITLE: 'My first page!',
         CODE: 'firstpage',
         SITE_ID: 292,
```

```
            ADDITIONAL_FIELDS: {
               THEME_CODE: 'wedding'
            }
         }
      },
      function(result)
      {
         if(result.error())
         {
            console.error(result.error());
         }
         else
         {
            console.info(result.data());
         }
      }
   );
```

Обратите внимание, при создании страницы передается код темы страницы (`THEME_CODE: 'wedding'`). Это необходимо, чтобы страница была в соответствующей цветовой схеме. Если этого не сделать, страница будет в теме по-умолчанию.

landing.landing.addByTemplate

Метод для добавления Страницы по шаблону (список шаблонов, который видит пользователь перед созданием страницы). Возвращает ID созданной страницы или ошибку.

Вы не можете влиять на поля создаваемой страницы, для этого вам поможет landing.landing.add.

Параметры

Параметр	Описание	С версии
siteId	ID сайта, в котором требуется создать страницу.	
code	Идентификатор шаблона для создания. Список шаблонов вы можете получить методом landing.demos.getPageList.	
fields	Необязательный. Можно передать массив полей для создаваемой страницы. Пока поддерживается только ключ TITLE и DESCRIPTION.	

Пример

```
BX24.callMethod(
   'landing.landing.addByTemplate',
   {
      siteId: 870,
      code: 'agency',
      fields: {
         TITLE: 'Заголовок страницы',
         DESCRIPTION: 'Описание страницы'
      }
   },
   function(result)
   {
      if(result.error())
         console.error(result.error());
      else
         console.info(result.data());
   }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

landing.landing.copy

```
landing.landing.copy (lid, toSiteId)
```

Метод копирует указанную страницу. Возвращает идентификатор новой страницы.

Параметры

Параметры	Описание	С версии
lid	Идентификатор страницы.	
toSiteId	Не обязательный параметр, идентификатор сайта. Если указан, копирование произойдет в указанный сайт.	
toFolderId	Не обязательный параметр, идентификатор папки. Если указан, то копирование произойдет в указанную папку (при ее существовании и наличии туда доступа). В ином случае копирование произойдет в те же папку, в которой находится страница,	18.7.500

	либо в корень, в случае если источник также в корне.	

Пример

```
BX24.callMethod(
    'landing.landing.copy',
    {
        lid: 1688
    },
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.landing.delete

```
landing.landing.delete(lid)
```

Метод для удаления Страницы. Возвращает *true* в случае успеха, или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор сущности.	

Пример

```
BX24.callMethod(
        'landing.landing.delete',
        {
                lid: 350
        },
        function(result)
        {
```

```
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.landing.getadditionalfi

```
landing.landing.getadditionalfields(lid)
```

Метод для получения дополнительных полей страницы.
Возвращает дополнительные поля сайта или ошибку.

Параметры

Метод	Описание	С версии
lid	идентификатор страницы	

Пример

```
BX24.callMethod(

'landing.landing.getadditionalfields',
        {
                lid: 349
        },
        function(result)
```

```
        {
                if(result.error())
                {
console.error(result.error());
                }
                else
                {
console.info(result.data());
                }
        }
);
```

landing.landing.getlist

```
landing.landing.getlist(params)
```

Метод для получения списка страниц.

Обратите внимание, помеченные как удаленные страницы не фигурируют в выборках. Чтобы получить их явно, необходимо при фильтрации указать ключ `DELETED` со значением Y или N.

Параметры

Параметр	Описание	С версии
params	Опциональный массив, с опциональными ключами: **select**, **filter**, **order**, **group**, которые содержат значения таблицы основных полей сущности. Дополнительно можно передать флаги `get_preview = 1` (вернуть превью страниц), `get_urls = 1` (вернуть публичные адреса страниц), `check_area` (вернуть флаг	

	IS_AREA является ли страница включаемой областью).	

Пример

```
BX24.callMethod(
        'landing.landing.getList',
        {
                params: {
                        select: [
                                'ID',
'TITLE'
                        ],
                        filter: {
                                TITLE:
'%услуги%',
                                SITE_ID: 205
                        },
                        order: {
                                ID: 'DESC'
                        }
                }
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
```

```
                        }
            }
);
```

landing.landing.getpreview

```
landing.landing.getpreview (lid)
```

Метод возвращает путь до превью страницы или ошибку.

Параметры

Метод	Описание	С версии
lid	идентификатор страницы	

Пример

```
BX24.callMethod(
        'landing.landing.getpreview',
        {
                lid: 351
        },
        function(result)
        {
```

```
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.landing.getpublicurl

```
landing.landing.getpublicurl(lid)
```

Метод возвращает веб-адрес страницы или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	

Пример

```
BX24.callMethod(
        'landing.landing.getpublicurl',
        {
                lid: 351
        },
        function(result)
        {
```

```
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.landing.markDelete

```
landing.landing.markDelete(
    $lid
)
```

Метод помечает страницу как удаленную.

Параметры

Параметр	Описание	Версия
$lid	Идентификатор страницы.	

Пример

```
BX24.callMethod(
    'landing.landing.markDelete',
    {
        lid: 1688
    },
    function(result)
```

```
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.landing.markUnDelete

```
landing.landing.markUnDelete(
  $lid
)
```

Метод помечает страницу как не удаленную.

Параметры

Параметр	Описание	Версия
$lid	Идентификатор страницы.	

Пример

```
BX24.callMethod(
   'landing.landing.markUnDelete',
   {
      lid: 1688
   },
   function(result)
```

```
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.landing.move

```
landing.landing.move (
    lid,
    toSiteId,
    toFolderId
)
```

Метод перемещает страницу в другой сайт и/или папку.

Параметры

Параметры	Описание	С версии
lid	Идентификатор страницы, которую надо переместить.	
toSiteId	Идентификатор сайта, куда надо переместить страницу. Должны быть права на запись в данный сайт.	
toFolderId	Идентификатор папки сайта, куда надо переместить страницу. Папка должна находиться в указанном сайте. Для перемещения в корень	

	сайта параметр следует опустить. (для перемещения в текущем сайте - опустить и параметр toSiteId).	

Пример

```
BX24.callMethod(
    'landing.landing.move',
    {
        lid: 11262,
        toSiteId: 1817,
        toFolderId: 737
    },
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.landing.publication

```
landing.landing.publication(lid)
```

Метод для публикации страницы. Возвращает *true* или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	

Пример

```
BX24.callMethod(
        'landing.landing.publication',
        {
                lid: 351
        },
        function(result)
        {
```

```
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.landing.removeEntities

```
landing.landing.removeEntities(lid, data)
```

Метод удаляет связанные сущности лендинга - блоки, и картинки блоков.

Обратите внимание: при удалении блоков связанные с ними картинки удаляются в любом случае. Но может возникнуть ситуация, когда для очистки мусора, необходимо удалить картинки независимо от блока. Используйте данный метод в этом случае.

Параметры

Параметры	Описание	С версии
lid	Идентификатор лендинга	
data	Ассоциативный массив, где в ключе **blocks** содержатся блоки на удаление, а в ключе **images** пары блок-картинка, в которых требуется удалить изображения (блоки в таком случае не удаляются).	

Пример

```
BX24.callMethod(
   'landing.landing.removeEntities',
   {
      lid: 648,
      data: {
         blocks: [12167, 123],
         images: [
            {
               block: 12269,
               image: 6866
            },
            {
               block: 12268,
               image: 6861
            }
         ]
      }
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
// В примере мы удаляем блоки с ID 12167,
123, а также картинку 6866 (из блока 12269)
и картинку 6861 (из блока 12268).
// Все сущности лежат в лендинге 648.
```

landing.landing.resolveIdByPu

```
landing.site.resolveIdByPublicUrl(
    landingUrl,
    siteId
)
```

Метод по переданному относительному URL страницы возвращает идентификатор страницы.

Параметры

Параметр	Описание	Версия
landingUrl	Относительный URL страницы.	
siteId	Идентификатор сайта.	

Пример

```
BX24.callMethod(
    'landing.landing.resolveIdByPublicUrl',
```

```
    {
        landingUrl:
'/folder/sub/folder/page/',
        siteId: 1817
    },
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.landing.unpublic

```
landing.landing.unpublic(lid)
```

Метод для снятия с публикации страницы. Возвращает true или ошибку.

Параметры

Параметр	Описание	С версии
lid	Идентификатор страницы	

Пример

```
BX24.callMethod(
    'landing.landing.unpublic',
    {
        lid: 351
    },
    function(result)
    {
```

```
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.landing.update

```
landing.landing.update(lid, fields)
```

Метод для изменения страницы. Возвращает *true* в случае успеха, или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор сущности	
fields	Изменяемые поля сущности	

Пример

```
BX24.callMethod(
        'landing.landing.update',
        {
                lid: 349,
                fields: {
```

```
                                TITLE: 'My second
page!'
                        }
                },
                function(result)
                {
                        if(result.error())
                        {

console.error(result.error());
                        }
                        else
                        {

console.info(result.data());
                        }
                }
);
```

Перед работой с блоками

Добавление блока происходит в режиме редактирования страницы. В данном режиме иные идентификаторы блоков, чем у опубликованной страницы. После работы с блоками страницу необходимо опубликовать.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

landing.landing.addblock

```
landing.landing.addblock(lid, fields)
```

Метод для добавление нового блока на страницу. Возвращает идентификатор нового блока или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
fields	Массив полей блока, где из поддерживаемого пока только следующие значения: ▪ **CODE** - символьный код блока. Код блока можно получить из метода landing.block.getrepository. Если добавляется блок, который был зарегистрирован партнером через landing.repo.register, то необходимо передавать для CODE значение `repo_<ID>`, где <ID> -	

	идентификатор такого блока. ▪ **AFTER_ID** - после какого блока (его ID) надо добавить новый блок (если не указано, блок добавится в начало) ▪ **ACTIVE** - активность блока (Y / N) ▪ **CONTENT** - полностью иное содержимое блока (см. замечания для метода landing.block.updatecontent)	

Пример

```
BX24.callMethod(
        'landing.landing.addblock',
        {
                lid: 351,
                fields: {
                        CODE: '15.social'
                }
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
```

```
                        }
            }
);
```

landing.landing.copyblock

```
landing.landing.copyblock(lid, block, params)
```

Метод для копирования блока со страницы на страницу. Возвращает идентификатор нового блока.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы, куда надо скопировать блок.	
block	Идентификатор блока, который может быть на другой странице (операция только в этом случае имеет смысл)	
params	массив параметров, где поддерживается пока один ключ - AFTER_ID - после какого блока вставить новый.	

Пример

```
BX24.callMethod(
        'landing.landing.copyblock',
        {
                lid: 349,
                block: 6428
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.landing.deleteblock

```
landing.landing.deleteblock(lid, block)
```

Метод для полного удаления блока со страницы. Возвращает *true* или ошибку. Для временного полного скрытия блока рекомендуется воспользоваться методом landing.landing.markdeletedblock.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	

Пример

```
BX24.callMethod(
        'landing.landing.deleteblock',
        {
```

```
                lid: 351,
                block: 6483
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.landing.downblock

```
landing.landing.downblock(lid, block)
```

Метод для опускания блока на одну позицию вниз на странице. Возвращает *true* или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	

Пример

```
BX24.callMethod(
        'landing.landing.downblock',
        {
                lid: 351,
                block: 6428
```

```
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.landing.favoriteBlock

```
landing.landing.favoriteBlock(
    lid,
    block,
    meta
)
```

Метод сохраняет имеющийся на странице блок в "Мои блоки". Возвращает идентификатор нового сохраненного блока.

Примечание: Метод может пригодиться при удалении блока из сохраненных.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы.	
block	Идентификатор блока.	
meta	Объект информации для сохранения блока. Содержит	

	поля: ▪ name – название блока; ▪ section – массив категорий, куда сохранить блок; ▪ preview – изображение блока.	

Пример

```
BX24.callMethod(
    'landing.landing.favoriteBlock',
    {
        lid: 11262,
        block: 81827,
        meta: {
           name: 'Мой блок',
           section: ['text', 'text_image'],
           preview:
'https://mycdn.com/pic/1.jpg'
        }
    },
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.landing.hideblock

```
landing.landing.hideblock(lid, block)
```

Метод скрывает блок со странице. Возвращает *true* или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	

Пример

```
BX24.callMethod(
        'landing.landing.hideblock',
        {
                lid: 351,
                block: 6428
        },
```

```
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.landing.markdeletedbl

```
landing.landing.markdeletedblock(lid, block)
```

Метод помечает блок как удаленный, но не удаляет его физически.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	

Пример

```
BX24.callMethod(
   'landing.landing.markdeletedblock',
   {
      lid: 627,
      block: 11923
   },
```

```
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.landing.markundeleted

```
landing.landing.markundeletedblock(lid,
block)
```

Метод восстанавливает блока из помеченных как удаленный

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	

Пример

```
BX24.callMethod(
   'landing.landing.markundeletedblock',
   {
      lid: 627,
      block: 11923
```

```
    },
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.landing.moveblock

```
landing.landing.moveblock(lid, block, params)
```

Метод для переноса блока со страницы на страницу. Возвращает *true* или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы, куда надо скопировать блок.	
block	Идентификатор блока, который может быть на другой странице (операция только в этом случае имеет смысл)	
params	массив параметров, где поддерживается пока один ключ - AFTER_ID - после какого блока вставить новый.	

Пример

```
BX24.callMethod(
        'landing.landing.moveblock',
        {
                lid: 349,
                block: 6428
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.landing.showblock

```
landing.landing.showblock(lid, block)
```

Метод для показа блока со странице. Возвращает *true* или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	

Пример

```
BX24.callMethod(
        'landing.landing.showblock',
        {
                lid: 351,
                block: 6428
        },
```

```
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.landing.unFavoriteBloc

```
landing.landing.unFavoriteBlock(
    blockId
)
```

Метод удаляет блок, который был сохранен в "Мои блоки".

Параметры

Метод	Описание	С версии
blockId	Идентификатор блока.	

Пример

```
BX24.callMethod(
    'landing.landing.unFavoriteBlock',
    {
        blockId: 81827
```

```
    },
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.landing.upblock

```
landing.landing.upblock(lid, block)
```

Метод для поднятия блока на одну позицию вверх на странице. Возвращает *true* или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	

Пример

```
BX24.callMethod(
        'landing.landing.upblock',
        {
                lid: 351,
                block: 6428
```

```
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

Сущность Блоки

Блок это html-код, который сопровожден файлом манифеста. Вот пример такого простейшего блока:

```
<section class="landing-block">
    <div class="text-center g-color-gray-dark-v3 g-pa-10">
        <div class="g-width-600 mx-auto">
            <div class="landing-block-node-text g-font-size-12 ">
                <p>© 2017 All right reserved. Developed by
                <a href="#" class="landing-block-node-link g-color-primary">Bitrix24</p>
            </div>
        </div>
    </div>
</section>
```

Стоит помнить, что в момент вывода блоков (как в режиме редактирования, так и в режиме просмотра) он обрамляется в спец.обертку `<div id="anchor" class="block-wrapper block-code">...</div>`, где:

- **anchor** – якорь блока, если не изменен пользователем, то вида block123, где 123 id блока;
- **block-wrapper** – одинаковый класс для всех блоков;
- **block-code** – класс, зависящий от кода блока, где **code** является непосредственно кодом блока (приведенный к безопасному виду).

Внимание! В режиме редактирования создаются копии всех блоков до их публикации. И обращение по ID к блокам надо делать ссылаясь на блоки именно чернового варианта.

Верстку новых блоков вы можете заказать у специалиста, или воспользоваться предложенными дополнительными блоками вендора.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Файл манифеста

Описание

Файл манифеста сопровождает каждый блок и описывает редактируемые части блока, а также содержит его название, описание, файлы JS/CSS.

Примеры

Важно! Результат работы блока должен обязательно содержать минимум один тег, даже если это техническая информация.
Например: `<div>Данный блок работает только после публикации</div>`.

Пример блока:

```
    <section class="landing-block container-fluid px-0 g-theme-business-bg-blue-dark-v1">
        <div class="row no-gutters align-items-start">

            <div class="landing-block-card js-animation fadeIn col-md-6 col-lg-6 g-flex-centered">
                <div class="landing-block-node-card-container text-center g-pa-30 w-100">
                    <div class="landing-
```

```
block-node-card-header text-uppercase u-
heading-v2-4--bottom g-brd-primary g-mb-40">
                        <h2 class="landing-
block-node-title h1 u-heading-v2__title g-
line-height-1_3 g-font-weight-600 g-font-
size-40 g-color-white g-mb-minus-10">Help
make <br> money</h2>
                    </div>

                                        <div
class="landing-block-node-text g-color-gray-
light-v2">

<p>Sed feugiat porttitor nunc, non dignissim
ipsum vestibulum in. Donec in blandit dolor.
Vivamus a fringilla lorem, vel faucibus
ante. Nunc
ullamcorper, justo a iaculis elementum, enim
orciviverra eros, fringilla porttitor lorem
eros vel odio. Praesent egestas ac arcu ac
convallis. Donec ut diam
risus purus.</p>

</div>
                </div>
            </div>

            <div class="landing-block-card
js-animation fadeIn col-md-6 col-lg-6 g-
flex-centered">
                <div class="landing-block-
node-card-container text-center g-pa-30 w-
100">
                    <div class="landing-
block-node-card-header text-uppercase u-
heading-v2-4--bottom g-brd-primary g-mb-40">
                        <h2 class="landing-
```

```
block-node-title h1 u-heading-v2__title g-
line-height-1_3 g-font-weight-600 g-font-
size-40 g-color-white g-mb-minus-10">Help
make <br> money</h2>
                    </div>

                                        <div
class="landing-block-node-text g-color-gray-
light-v2">

<p>Sed feugiat porttitor nunc, non dignissim
ipsum vestibulum in. Donec in blandit dolor.
Vivamus a fringilla lorem, vel faucibus
ante. Nunc
ullamcorper, justo a iaculis elementum, enim
orciviverra eros, fringilla porttitor lorem
eros vel odio. Praesent egestas ac arcu ac
convallis. Donec ut diam
risus purus.</p>

</div>
                </div>
            </div>

        </div>
    </section>
```

Примечание: Соблюдать уникальность разметки в манифесте важно только в рамках одного блока. Между блоками одинаковые селекторы могут иметь даже абсолютно разный смысл.

Вот так мог бы выглядеть файл манифеста для блока выше:

```
<?php
if (!defined('B_PROLOG_INCLUDED') ||
B_PROLOG_INCLUDED !== true)
{
        die();
}

use \Bitrix\Main\Localization\Loc;

return array(
        'block' => array(
                'name' => "Текст в две
колонки",
                'section' =>
array('columns', 'text'),
        ),
        'cards' => array(
                '.landing-block-card' =>
array(
                        'name' => "Колонка",
                        'label' => array(
                                '.landing-
block-node-subtitle',
                                '.landing-
block-node-title',
                        ),
                ),
        ),
        'nodes' => array(
                '.landing-block-node-title'
=> array(
                        'name' =>
"Заголовок",
                        'type' => 'text',
                ),
                '.landing-block-node-text'
=> array(
```

```
                        'name' => "Текст",
                        'type' => 'text',
                ),
        ),
        'style' => array(
                'block' => array(
                        'block-default',
                ),
                'nodes' => array(
                        '.landing-block-
card' => array(
                                'name' =>
"Колонка",
                                'type' =>
array('columns', 'animation'),
                        ),
                        '.landing-block-
node-title' => array(
                                'name' =>
"Заголовок",
                                'type' =>
'typo',
                        ),
                        '.landing-block-
node-text' => array(
                                'name' =>
"Текст",
                                'type' =>
'typo',
                        ),
                        '.landing-block-
node-card-header' => array(
                                'name' =>
"Заголовок",
                                'type' =>
'border-color',
                        ),
```

```
                ),
        ),
        'attrs' => array(
                '.landing-block-node-text'
=> array(
                        'name' => 'Настройка
текст',
                        'type' =>
'dropdown',
                        'attribute' =>
'data-copy',
                        'items' => array(
                                'val1' =>
'Значение 1',
                                'val2' =>
'Значение 2',
                        ),
                ),
        ),

        'assets' => array(
                'css' => array(

'https://site.com/aaa.css',
                ),
                'js' => array(

'https://site.com/aaa.js',
                ),
                'ext' => array(
                        'landing_form',
                ),
        ),
);
```

Поля манифеста блока

Поля манифеста блока:

- Ключ block
- Ключ assets
- Ключ menu
- Ключ nodes
- Ключ style
- Ключ cards
- Ключ attrs

Ключ block

Ключ **block** содержит название и категорию блока (или массив категорий). В системе есть некоторое количество категорий, вот они:

```
array(
    'cover' => Обложка,
    'about' => О проекте,
    'title' => Заголовок,
    'text' => Текстовый блок,
    'image' => Изображение,
    'gallery' => Галерея,
    'phrase' => Цитата,
    'benefits' => Преимущества,
    'columns' => Колонки,
    'separator' => Разделитель,
    'menu' => Меню,
    'footer' => Подвал сайта,
    'pages' => Список страниц,
    'tiles' => Плитка и ссылка,
    'forms' => CRM-форма,
    'team' => Команда,
```

```
    'feedback' => Отзывы,
    'schedule' => Расписание,
    'steps' => Этапы,
    'contacts' => Контакты,
    'social' => Социальные сети,
    'tariffs' => Тарифы,
    'partners' => Партнеры,
    'other' => Другое,
    'popular' => Популярные,
    'sidebar' => Сайдбар,
    'video' => Видео,
    'text_image' => Текст с картинками,
    'countdowns' => Таймеры для акций,
    'separator' => Переходы и разделители,
    'news' => Новостная лента,
);
```

Если нужной категории нет в списке, просто напишите ее текстом в манифесте, категория будет добавлена.

Помимо этого в данном ключе могут содержаться следующие настройки:

- **subtype** – тип спец.блока, подробнее почитать можно здесь, допускает одиночное значение, или множественное в виде массива.
- **type** - может содержать тип сайта, на страницах которого блок может работать (по умолчанию блок показывается везде). На данный момент поддерживаются:
    - page: обычные сайты
    - store: магазины
    - knowledge: базы знаний
    - group: базы знаний групп соцсети

Ключ menu

Меню на сайте вполне может быть обычным html-блоком, и под такие блоки отведен целый раздел в каталоге блоков. Ссылки пунктов меню просто редактируются как обычные карточки. Но как построить многоуровневое меню? Вы могли его видеть [dw]в Базе Знаний[/dw][di][/di].

Для этого предназначена отдельная запись в манифесте с ключом menu. Вот как выглядит такая запись:

```
'menu' => [
   '.landing-block-node-menu' => [
      'item' => '.landing-block-node-menu-
item',
      'name' =>
Loc::getMessage('LANDING_BLOCK_MENU_22-
NAVBAR'),
      'root' => [
         'ulClassName' => 'landing-block-
node-menu navbar-nav',
         'liClassName' => 'landing-block-
node-menu-item nav-item',
         'aClassName' => 'landing-block-
node-menu-link nav-link',
      ],
      'children' => [
         'ulClassName' => 'landing-block-
node-menu navbar-nav',
         'liClassName' => 'landing-block-
node-menu-item nav-item',
         'aClassName' => 'landing-block-
node-menu-link nav-link',
      ],
      'nodes' => [
         '.landing-block-node-menu-link' =>
[
            'name' =>
Loc::getMessage('LANDING_BLOCK_MENU_22-
LINK'),
```

```
            'type' => 'link',
        ],
    ],
],
```

Запись описывает данный html:

```
<ul class="landing-block-node-menu navbar-
nav">
   <li class="landing-block-node-menu-item
nav-item">
      <a href="#" target="_self"
class="landing-block-node-link nav-
link">Bitrix24 Knowledge Base</a>
   </li>
   <li class="landing-block-node-menu-item">
      <a href="#" target="_self"
class="landing-block-node-link nav-
link">Tasks and Projects</a>
      <ul class="landing-block-node-menu">
         <li class="landing-block-node-menu-
item nav-item">
            <a href="#" target="_self"
class="landing-block-node-link nav-
link">Article example</a>
         </li>
      </ul>
   </li>
</ul>
```

Рассмотрим ключи блока menu:

Корень массива (.landing-block-node-menu в данном случае) указывает селектор тега `<ul>`, который должен определяться как

меню. Как вы уже поняли, в рамках одного манифеста может быть несколько многоуровневых меню.

Значение **item** описывает каждый `<li>` внутри списка `<ul>`. То есть те теги `<li>`, которые будут определяться как пункты меню.

Значение **name** описывает название меню.

Значение **root** описывает корневой `<ul>`. А **children** все корневые `<ul>`. ulClassName описывает классы тега `<ul>` (их в каждом значении может быть несколько, разделенных пробелами), liClassName описывает классы тега `<li>`, ну а aClassName описывает классы тега `<a>`. В данном случае родительские и дочерние элементы по структуре совпадают. Каких-то дополнительных тегов структура меню не предполагает на данный момент.

Значение **nodes** описывает структуру внутренности `<li>`. На данный момент это просто ссылка. Внимательный читатель может догадаться, что с помощью данного пункта можно описать сколь угодно сложную структуру каждого пункта. Верно, но пока это не поддерживается в полной мере.

Ключ assets

Ключ **assets** содержит в себе JS и CSS, которые необходимо подключить при добавлении блока на страницу. Если несколько блоков используют один и тот же файл JS/CSS, не страшно, каждый файл подключится только один раз.

Ключ **ext** это библиотеки JS ядра Битрикс. В облачном варианте вы не можете подключить любое расширение ядра. Пока разрешено подключать только библиотеки, упомянутые в специальных блоках и в интерактивных блоках.

Если вы используете какие-либо сторонние библиотеки в своем коде, которые уже подключены к основному ядру (например, jQuery), рекомендуется вывод команд в вашем скрипте оборачивать в системный метод. Тогда ваш код успешно инициализируется после всех системных подключений:

```
BX.ready(function()
{
    console.log($(window));
});
```

Ключ nodes

Ключ **nodes** содержит блоки, содержимое которых разрешено к изменению контента. Здесь и далее применяется идеология css-селекторов для указания конечных нод. Вы должны это понимать, чтобы хорошо разобраться в дальнейшем API блоков.

В качестве селектора рекомендуется выбирать говорящее имя класса. Чтобы отличать обычные классы, от структурных классов-селекторов, рекомендуется давать имени говорящий префикс. Например, `"landing-block-node-"`. Один и тот же селектор допускается использовать в разных блоках. Селектор ноды **не совпадает** с селекторами карточек данного блока.

Таким образом, в ключе nodes перечислены ключами селекторы, контент которых можно править, а также указано название ноды и ее тип. Подробнее прочитать про типы нод.

Как вы уже, наверное, догадались, в зависимости от типа ноды и ее указания в данном блоке, она станет доступна к редактированию, и появится в форме редактирования контента блока.

Помимо типа, названия и специфических ключей того или иного типа есть также общие параметры ноды любого типа:

allowInlineEdit – если передать значение ключа равное *false*, то данные ноды будут запрещены к редактированию инлайн, но в то же время будут доступны к редактированию через форму редактирования блока.

useInDesigner - если передать *false*, элемент будет проигнорирован в дизайнере блока.

group – группировка нод, если указать одинаковое значение данного ключа для нескольких нод (в рамках одного блока), то при клике на любую из нод откроется интерфейс редактирования всей группы нод.

Ключ style

Ключ **style**. Ключ очень похож на ключ nodes за исключением того, что в данном ключе размечается допустимый дизайн: каким нодам разрешено менять внешний вид, и какого они типа.

Типы style могут быть следующими (в том числе их компоновка в виде массива):

- **box** – блочные элементы
- **button** – ссылки в виде кнопок
- **typo** – вся типографика
- **typo-simple** – упрощенная типографика
- **typo-link** – типографика ссылок
- **paddings** – все отступы
- **navbar** – навигационные блоки
- **navbar-full** – навигационные блоки
- и некоторые другие...

Полный список допустимых типов и какие css-стили он объединяет:

Ключ cards

Содержит так называемые карточки блока. Это повторяемый контент. Например, перечень услуг, фотографии сотрудников, и так далее. Разметив таким образом блок, в интерфейсе появится функционал клонирования, удаления, изменения карточек.

В простейшем виде он выглядит так, как указано в манифесте. Если вы хотите, вы можете изучить также расширенное управление карточками.

В качестве селектора рекомендуется выбирать говорящее имя класса. Чтобы отличать обычные классы, от структурных классов-

селекторов, рекомендуется давать имени говорящий префикс. Например, `"landing-block-card-"`. Один и тот же селектор вы можете использовать в разных блоках. Селектор карточки **не совпадает** с селекторами нод данного блока.

> Карточки одного селектора не должны быть с карточками другого селектора в одном общем родителе. Для случая общего родителя пользуйтесь расширенными карточками.

Ключ attrs

Описание атрибутов смотрите на отдельной странице.

Идеология изменения стилей блока

При изменении внешнего вида блоков, не происходит изменение атрибута **style** нод. Происходит изменение только классов. Например, если вы хотите поменять font-size, с 12 до 16, система изменит не 'font-size', а условный класс g-fontsize12 на g-fontsize16.

Для работы анимации

В штатном механизме для работы анимации нужно:

- чтобы у ноды был класс js-animation;
- чтобы в манифесте для данной ноды была настройка в разделе style - 'type' => 'animation';
- чтобы анимация работала сразу по-умолчанию, ещё нужно добавить какой-то из классов анимации (в нашем примере блока - fadeIn), но не обязательно, настройка заработает и так.

© «Битрикс», 2001-2008, «1С-

Атрибуты

Описание

С помощью ключа **attrs** в манифесте блока указывается список атрибутов для хранения данных, привязанных к определенным нодам. Применяется это повсеместно – от дефолтных значений полей, счетчиков до настройки карты, видео, и много чего еще. Как правило, в пару к набору атрибутов идет определенный скрипт, который умеет со всем этим работать. Либо атрибуты могут участвовать в стилизации блоков, путем указания в CSS, что карточка с определенным атрибутом имеет другой цвет (например).

Каждый атрибут описывается:

- названием,
- кодом,
- типом,
- ключом **items** (в случае списочного типа).

Места размещения

Ключ **attrs** в манифесте может размещаться в следующих местах:

1. Непосредственно в корне, как указано в примере манифеста.

```
'attrs':
{
        '.landing-block-node-text':
        {
            'name': 'Настройка текст',
            'type': 'dropdown',
            'attribute': 'data-copy'
```

```
        }
 },
```

2. В ключе style, в этом случае атрибут выводится в форме настроек дизайна.

```
'style':
   {
      '.landing-block-node-card-button':
         {
            'name': 'Button',
            'type': ['border-color',
'button', 'animation'],
            'additional': {
               'attrs': [
                  [
                     'type': 'text',
                     'name': 'Text
field',
                     'attribute': 'data-
test-card-attr'
                  ]
               ]
            }
         },
```

3. В описании карточки. В таком случае атрибут применяется непосредственно к каждой карточке отдельно

```
'cards':
   {
      '.landing-block-node-card-button':
         {
            'name': 'Card',
            'additional': {
```

```
                'attrs': [
                   [
                      'type': 'text',
                      'name': 'Text
field',
                      'attribute': 'data-
test-card-attr'
                   ]
                ]
             }
          },
```

Группировка атрибутов

Если требуется часть атрибутов группировать, то делается это следующим образом:

```
// корневое размещение
'attrs' => array(
   '' => array(
      array(
         'name' => 'Test group',
         'attrs' => array(
            array(
               "type" => "checkbox",
               // Переопределение селектора
(если нужно)
               "selector" =>
"bitrix:catalog.section",
               "name" => "",
               "items" => array(
                  array("name" =>
"Отображение товаров", "value" => "1"),
                  array("name" =>
```

```
"Отображение товаров 2", "value" => "2"),
                  array("name" =>
"Отображение товаров 3", "value" => "3"),
               ),
               "attribute" => "data-
checkbox"
            ),
            array(
               "type" => "checkbox",
               "name" => "",
               "items" => array(
                  array("name" =>
"Отображение товаров 22", "value" => "1")
               ),
               "compact" => true,
               "attribute" => "data-
checkbox2"
            )
         )
      ),
      array(
         "type" => "checkbox",
         "name" => "",
         "items" => array(
            array("name" => "Отображение
товаров 33", "value" => "1")
         ),
         "attribute" => "data-checkbox3"
      )
   )
)
// блок style (обратите внимание, в случае
style поддерживается только либо без групп,
либо группировка в рамках одного селектора)
'additional' =>
      array(
         array(
```

```
            'name' => 'Test group',
            'attrs' => array(
               array(
                  "type" => "text",
                  "name" => "Test",
                  "attribute" => "data-text"
               ),
               array(
                  "type" => "text",
                  "name" => "Test 2",
                  "attribute" => "data-
text2"
               )
            )
         )
      )
```

Отличающиеся селекторы

Если вы хотите, чтобы значения атрибутов сохранялись в иной селектор, то просто укажите у конкретного атрибута другой селектор. (Это может быть полезно, чтобы не добавлять лишние ноды для визуального изменения):

```
array(
   'name' => 'Текстовое поле',
   'type' => 'text',
   'attribute' => 'data-text-field',
   'selector' => '.demo-another-selector'
)
```

Типы атрибутов

Атрибуты - это условное хранение hidden-значений. Например, стартовые координаты карты. Естественно, атрибуты имеет смысл вводить только вкупе с неким JS-кодом, который эти атрибуты

умеет использовать. Атрибуты необходимо зарегистрировать в манифесте в ключе **attrs**.

На данный момент поддерживаются следующие типы атрибутов:

- **text** - обычная текстовая строка.
- **html** - многострочное текстовое поле
- **images** - картинка со стандартными контролами - выбор с компьютера или поиск в библиотеках.
- **icon** - иконка.
- **dropdown** - выпадающий список.
- **checkbox** - группа чекбококсов. Если вы хотите вывести одиночный чекбокс, просто укажите одно значение в items.
- **multiselect** - множественный список.
- **link** - ссылка со стандартными контролами.
- **url** - упрощенный вариант ссылки: выбор страницы/блока или произвольного URL.
- **slider** / **range-slider** - варианты слайдеров массива значений.
- **palette** - палитра.
- **sortable-list** - сортируемый список значений. Сортировка происходит посредством перетаскивания элементов мышкой.
- **position** - набор стрелок для указания положения элемента в блоке.
- **date** - выбор даты и времени.

Конкретные примеры с данными типами смотрите ниже. Там же вы сможете найти дополнительные опции вариативности.

Дополнительно

Помимо специфических свойств того или иного типа (смотрите пример ниже) каждый тип может обладать дополнительными свойствами:

- **hidden** - атрибут регистрируется, но не выводится на редактирование в карточке блока, удобно для регистрации блоков, когда санитайзер не пропускает не зарегистрированные атрибуты.

Пример

```
<?php
$attrs = array(
   ".landing-node" => array(
      array(
         "type" => "text",
         "name" => "Test attr field",
         "placeholder" => "Type your text",
         "value" => "default_value",
         "attribute" => "data-test-text"
         "textOnly" => false//если в true,
то при редактировании не будет подключаться
редактор
      ),
      ),
      array(
         "type" => "image",
         "name" => "Test attr image field",
         "value" => array(
            "src" =>
"http://bitrix24.io/bitrix/images/landing/ap
p-store-badge.svg",
            "alt" => "test alt"
         ),
         "attribute" => "data-test-image"
      ),
      array(
         "type" => "icon",
         "name" => "Test attr icon field",
         "value" => array(
            "classList" => array("fa", "fa-
address-card")
         ),
         "attribute" => "data-test-icon"
      ),
      array(
         "type" => "dropdown",
         "name" => "Test attr dropdown
```

```
field",
         "items" => array(
            array("name" => "#1", "value" =>
1),
            array("name" => "#2", "value" =>
2),
            array("name" => "#3", "value" =>
3),
            array("name" => "#4", "value" =>
4)
         ),
         "value" => 3,
         "attribute" => "data-test-dropdown"
      ),
      array(
         'name' => 'Checkbox field',
         'type' => 'checkbox',
         'attribute' => 'data-test-
checkbox',
         'items' => array(
            array('name' => 'Разрешить
указание количества товара', 'value' => '1',
'checked' => true),
            array('name' => 'Разрешить
оповещения для отсутствующих товаров',
'value' => '2', 'checked' => true),
            array('name' => 'Показывать
процент скидки', 'value' => '3', 'checked'
=> true),
            array('name' => 'Показыать
старую цену', 'value' => '4', 'checked' =>
true),
            array('name' => 'Разрешить
сравнение товаров', 'value' => '5',
'checked' => true)
         )
      ),
```

```
      array(
         'name' => 'Multi select field',
         'type' => 'multiselect',
         'attribute' => 'data-test-
multiselect',
         'items' => array(
            array('name' => 'Разрешить
указание количества товара', 'value' => '1',
'selected' => true),
            array('name' => 'Разрешить
оповещения для отсутствующих товаров',
'value' => '2', 'selected' => true),
            array('name' => 'Показывать
процент скидки', 'value' => '3'),
            array('name' => 'Показыать
старую цену', 'value' => '4', 'items' =>
array(
               array('name' => 'Разрешить
сравнение товаров', 'value' => '41',
'selected' => true),
               array('name' => 'Разрешить
указание количества товара', 'value' =>
'42', 'selected' => true),
               array('name' => 'Разрешить
оповещения для отсутствующих товаров',
'value' => '43', 'selected' => true),
               array('name' => 'Показывать
процент скидки', 'value' => '44', 'selected'
=> true)
            )),
            array('name' => 'Разрешить
сравнение товаров', 'value' => '5'),
            array('name' => 'Разрешить
указание количества товара', 'value' =>
'6'),
            array('name' => 'Разрешить
оповещения для отсутствующих товаров',
```

```
'value' => '7', 'selected' => true)
         )
      ),
      array(
         "type" => "link",
         "name" => "Test attr link field",
         "value" => array(
            "text" => "Link anchor",
            "href" => "/test",
            "target" => "_popup"
         ),
         "attribute" => "data-test-link"
      ),
      array(
         "type" => "slider",
         "name" => "Test attr slider field",
         "items" => array(
            array("name" => "1", "value" =>
1),
            array("name" => "2", "value" =>
2),
            array("name" => "3", "value" =>
3),
            array("name" => "4", "value" =>
4),
            array("name" => "5", "value" =>
5)
         ),
         "value" => 2,
         "attribute" => "data-test-slider"
      ),
      array(
         "type" => "range-slider",
         "name" => "Test attr range slider
field",
         "items" => array(
            array("name" => "1", "value" =>
```

```
1),
            array("name" => "2", "value" =>
2),
            array("name" => "3", "value" =>
3),
            array("name" => "4", "value" =>
4),
            array("name" => "5", "value" =>
5)
         ),
         "value" => array(
            "from" => 3,
            "to" => 5
         ),
         "attribute" => "data-test-range-
slider"
      ),
      array(
         "type" => "palette",
         "name" => "Test attr palette
field",
         "items" => array(
            array('name' => 'g-bg-
lightblue', 'value' => 'g-bg-lightblue'),
            array('name' => 'g-bg-lightblue-
opacity-0_1', 'value' => 'g-bg-lightblue-
opacity-0_1'),
            array('name' => 'g-bg-lightblue-
v1', 'value' => 'g-bg-lightblue-v1'),
            array('name' => 'g-bg-lightblue-
v1-opacity-0_1', 'value' => 'g-bg-lightblue-
v1-opacity-0_1'),
            array('name' => 'g-bg-darkblue',
'value' => 'g-bg-darkblue'),
            array('name' => 'g-bg-darkblue-
opacity-0_1', 'value' => 'g-bg-darkblue-
opacity-0_1'),
```

```
            array('name' => 'g-bg-indigo',
'value' => 'g-bg-indigo'),
            array('name' => 'g-bg-indigo-
opacity-0_1', 'value' => 'g-bg-indigo-
opacity-0_1'),
            array('name' => 'g-bg-red',
'value' => 'g-bg-red'),
            array('name' => 'g-bg-red-
opacity-0_1', 'value' => 'g-bg-red-opacity-
0_1'),
            array('name' => 'g-bg-red-
opacity-0_2', 'value' => 'g-bg-red-opacity-
0_2'),
            array('name' => 'g-bg-red-
opacity-0_5', 'value' => 'g-bg-red-opacity-
0_5'),
            array('name' => 'g-bg-red-
opacity-0_8', 'value' => 'g-bg-red-opacity-
0_8'),
            array('name' => 'g-bg-lightred',
'value' => 'g-bg-lightred'),
            array('name' => 'g-bg-lightred-
opacity-0_1', 'value' => 'g-bg-lightred-
opacity-0_1'),
            array('name' => 'g-bg-darkred',
'value' => 'g-bg-darkred'),
            array('name' => 'g-bg-darkred-
opacity-0_1', 'value' => 'g-bg-darkred-
opacity-0_1'),
            array('name' => 'g-bg-purple',
'value' => 'g-bg-purple')
         ),
         "property" => "background-color",
         "attribute" => "data-test-palette",

         // Set if tou need get color by
className from css
```

```
         // "stylePath" =>
"/path/to/stylesheet.css",

         // Set if you need get color from
styles for pseudo-element (::before,
::after)
         // "pseudo-element" => "::after",

         // Set if you need get color from
styles for pseudo-class (:hover, :active,
...)
         // "pseudo-class" => ":hover"
      ),
      array(
         "type" => "sortable-list",
         "name" => "Product blocks",
         "items" => array(
            array("name" => 'head', "value"
=> "1"),
            array("name" => "props", "value"
=> "2"),
            array("name" => "tp", "value" =>
"3"),
            array("name" => "qant", "value"
=> "4"),
            array("name" => "quant2",
"value" => "5"),
            array("name" => "action",
"value" => "6"),
            array("name" => "comp", "value"
=> "7")
         ),
         "value" => array("1", "2", "3",
"4", "5", "6", "7"),
         "attribute" => "data-catalog-prop-
sort"
      ),
```

```
      array(
         "type" => "position",
         "name" => "position",
         "items" => array(
            "top-left" => array("content" =>
"", "value" => "1"),
            "top-center" => array("content"
=> "", "value" => "2"),
            "top-right" => array("content"
=> "", "value" => "3"),
            "middle-left" => array("content"
=> "", "value" => "4"),
            "middle-center" =>
array("content" => "", "value" => "5"),
            "middle-right" =>
array("content" => "", "value" => "6"),
            "bottom-left" => array("content"
=> "", "value" => "7"),
            "bottom-right" =>
array("content" => "", "value" => "8")
         ),
         "value" => "top-right",
         "attribute" => "data-catalog-prop-
position"
      ),
      array(
         'name' => 'URL field',
         'type' => 'url',
         'value' => '#landing166',
         'attribute' => 'data-test-url',
         'disableBlocks' => true, //
Отключает выбор блоков
         'disableCustomURL' => false //
Отключает возможность ввести урл руками
      ),
      array(
         'name' => 'Datetime',
```

```
            'type' => 'date',
            'time' => true,//давать возможность
выбора точного времени
            'format' => 'ms',//'ms'
(миллисекунды) / 's' (секунды)
            'value' => 1621584180000
        )
    )
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Типы нод

Описание

Как вы можете узнать из файла манифеста, существует несколько типов нод – контейнеров, в которых расположен тот или иной контент. Давайте рассмотрим какие они есть на данный момент. В примерах показано как регистрировать в манифесте в ключе **nodes**, и как это выглядит в верстке.

text

Обычный текстовый контент с минимум инлайнового html.

```
'.landing-block-node-card-title' => array(
   'name' => 'Описание',
   'type' => 'text',
),
```

```
<h2 class="landing-block-node-card-
title">Company24 video</h2>
```

img

Изображение. Допускается как отдельным тегом (`<img>`), так и фоновое (как правило для тега `<div>`). Для данного типа необходимо также указывать рекомендуемый размер. Для чего это сделано? Картинки могут быть разные по своему предназначению. Аватарка может быть очень маленькой, а фон напротив - большим. Вместе с тем, если в аватарку контент-редакторы будут загружать

огромные картинки, они очень быстро исчерпают свободное место портала, а страница будет виснуть в браузере. Поэтому авторам блока необходимо самим заботиться о декларировании данных нод. Если размер не указан, система будет приводить такие картинки к единому небольшому размеру.

```
    '.landing-block-node-card-bgimg' =>
array(
        'name' => 'Фоновая картинка',
        'type' => 'img',
        // изображение будет приведено к
данному размеру
        'dimensions' => array('width' =>
1920, 'height' => 1080),
        // система уменьшит изображение,
только если оно превысит размер
        'dimensions' => array('maxWidth' =>
1920, 'maxHeight' => 1080),
        // систему будет увеличивать
изображение, пока ширина
        // или высота не будет совпадать с
указанными
        'dimensions' => array('minWidth' =>
1920, 'minHeight' => 1080),
    ),
```

```
<div class="landing-block-node-card-bgimg">
</div>
```

Важно! У данного типа обязательно должен присутствовать атрибут с изображением. У тега `img` это атрибут **src** и, очевидно, он не должен быть пустым. У атрибута **style** (тег `div`) это **background-image**.

link

Ссылка, несет в себе текст ссылки, значение ссылки, тип ссылки (ссылка, телефон, ...), а также тип открытия (в текущем окне, новом, ...).

```
'.landing-block-node-card-button' => array(
   'name' => 'Кнопка',
   'type' => 'link',
),
```

```
<a href="/"
   class="landing-block-node-card-button
text-uppercase btn u-btn-outline-white btn-
md rounded-0">
   Read more
</a>
```

Если ссылка не является текстовой (например, несет в себе картинку, или что-либо еще), то следует добавить параметр skipContent, чтобы при сохранении он не изменялся.

```
'.landing-block-node-card-button' => array(
   'name' => 'Кнопка',
   'type' => 'link',
   'skipContent' => true
),
```

icon

Визуально похоже на изображение, так как конечный пользователь оперирует вариантами иконок, за тем лишь исключением, что не может загружать свои в данные ноды. Технически это класс

элемента, который отрисовывает ту или иную иконку через стандартные иконочные шрифты.

```
'.landing-block-node-list-item-icon' =>
array(
   'name' => 'Иконка',
   'type' => 'icon',
),
```

```
<i class="landing-block-node-list-item-icon
fa fa-instagram"></i> //класс fa-instagram
отвечает за вывод иконки Инстаграмм
```

embed

Мультимедиа. Например, фоновое видео. При данном типе меняются только два атрибута – `src` и `source`.

```
'.landing-block-node-card-videobg' => array(
   'name' => 'Фоновое видео',
   'type' => 'embed',
),
```

```
<iframe
      class="landing-block-node-card-videobg
bg-video__video"
      width="100%"

src="//www.youtube.com/embed/q4d8g9Dn3ww?
autoplay=1&controls=0&loop=1&mute=1&rel=0"
```

```
      data-
source="https://www.youtube.com/watch?
v=q4d8g9Dn3ww"
      frameborder="0"
      allowfullscreen=""></iframe>
```

map

Позволяет работать с селектором как с гео-картой. В данный момент поддерживаются только карты Google. В режиме редактирования позволяет легко управлять балунами.

```
'.landing-block-node-map' => array(
   'name' => 'Местоположение нашего офиса',
   'type' => 'map',
),
```

```
<div class="landing-block-node-map mx-auto
w-100 g-min-height-430 h-100"></div>
```

Расширенное описание карточек

Описание

Для понимания принципов формирования и работы с карточками рекомендуется сначала ознакомиться с манифестом.

Расширенные карточки были призваны решать такие задачи:

- Разный набор контактов (e-mail и телефон, только телефон, телефон и соцсеть).
- Один и тот же набор полей, но разное графическое представление (социальные кнопки).
- И так далее.

Все карточки свернуты в [dw]компактные панели[/dw][di][/di], с быстрым доступом к основным функциям: Сортировать, Редактировать, Удалить. По клику на панель или на кнопку “Редактировать” - панель разворачивается в форму редактирования карточки.

Возможно добавление как пустой карточки, так и карточки из пресета.

Поддерживаются составные и динамические заголовок карточек. Название карточки формируется из тех данных что указал пользователь. Заголовок можно составлять из текста, изображений, иконок и ссылок. Заголовки будут динамически обновляться по мере редактирования значений карточки.

Манифест расширенной карточки

```
'cards' => [
   '.landing-block-card' => [
      'name' =>
Loc::getMessage('LANDING_BLOCK_4_FEATURES_3_
COLS_...'),
      'label' => [
         '.landing-block-node-element-icon',
         '.landing-block-node-element-title'
      ],
      'presets' => [
         'telegram' => [...],
         'instagram' => [...],
         'reddit' => [...],
         'whatsapp' => [...],
         'skype' => [...]
      ]
   ]
]
```

Ключ name

Определяет заголовок группы карточек. По умолчанию используется для формирования заголовков карточек вида "Карточка 1", "Карточка 2"

Ключ label

Определяет правила формирования заголовка карточки. В качестве значения можно передавать селектор ноды, либо массив селекторов нод из которых необходимо брать значения для формирования заголовка.

Ключ presets

Определяет набор пресетов карточек текущего набора. Если значение свойства не пустое, то добавить новую карточку можно

будет только из пресета. В качестве значения принимает массив пресетов, в качестве ключей массива необходимо указывать идентификаторы пресета.

Пресет

```
'telegram' => [
   'name' => ' Telegram',
   'html' => '<html-код-пресета>',
   'values' => [
      '.landing-block-node-element-title' =>
'Telegram',
      '.landing-block-node-element-text' =>
'Any text ...',
      '.landing-block-node-element-icon' =>
[
         'type' => 'icon',
         'classList' => ['landing-block-
node-element-icon', 'fa', 'fa-telegram']
      ]
   ],
   'disallow' => [
      '.landing-block-node-element-icon'
   ]
]
```

Ключ name

Определяет заголовок пресета. Поддерживает html. Отображается в выпадающем списке пресетов.

Ключ html

Верстка карточки. Может может отличаться от верстки остальных карточек. При добавлении верстки в пресет, нужно учитывать, что редактироваться будут только те ноды, которые определены в nodes и не переопределены в disallow.

Ключ values

Определяет значения нод и полей карточки с которыми они будут инициализированы. Ключ - селектор ноды из nodes, значение - значение ноды в соответствии с типом ноды.

Ключ disallow

Определяет какие ноды пользователь не сможет редактировать. В значении ожидается массив селекторов.

Верстка пресета

Верстка одного пресета это повторяемый блок кода.

Пример:

```
<li class="landing-block-node-list-item col
g-min-width-65 list-inline-item g-mr-0"
   data-card-preset="telegram">
   <a class="landing-block-node-list-item-
link d-block g-py-15 g-px-30 g-bg-telegram--
hover g-bg-telegram g-color-white text-
center" href="#">
      <i class="landing-block-node-list-
item-icon fa fa-telegram"></i>
   </a>
```

В идеологическом отличии от обычной карточки такие повторяемые контенты могут изменяться. Например, в одном `<li>` в примере выше может не быть ссылки, или вовсе, присутствовать изображение.

> Обратите внимание, что содержимое карточек может отличаться, но внешний блок лучше оставлять одинаковым. Также обратите внимание, что обязательно надо указывать data-card-preset="<код пресета>" как в пресете, так и в верстке (смотрите пример выше).

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Локализация блока

Локализация блоков возможна с версии 18.5.6. Перевод возможен на любое количество языков. Для локализации:

1. Создайте блок под свой родной язык.
2. В манифесте добавьте два ключа:

```
   'lang' => [
      'en' => [
         'Заголовок с разделителем на
светлом фоне' => 'Title with a separator
on a light background (translated)'
      ],
      'de' => [
         'Заголовок с разделителем на
светлом фоне' => 'Überschrift mit einem
Trennzeichen auf einem hellen
Hintergrund'
      ]
   ],
   'lang_original' => 'ru'
```

Где:
lang_original – язык, на котором манифест блока сделан. Внимание: именно фразы манифеста.
lang – все иные языки, минимально хотя бы один, иначе какой смысл локализации. (Никто не запрещает создавать блоки на французов и русских только, например.)

Рассмотрим подробнее массив **lang**. В его ключах перечислены все возможные поля name в вашем манифесте. Это и название самого блока, и название нод, стилей, атрибутов и их значений.

Система производит обход всего манифеста, и если встречает ключ **name**, ищет ему подходящую замену в массиве языков (либо текущий язык, либо en, если язык портала отличается от родного языка блока) и заменяет.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Блоки-меню

Если в секции block манифеста блока добавить ключ **subtype** со значением `menu`, то при добавлении блока в магазине меню будет автоматически наполняться разделами текущего каталога.

Помимо subtype секция **block** должна содержать ключ **subtype_params** со следующими параметрами:

```
'block' => array(
   'name' => 'Меню с логотипом слева и пунктами меню справа',
   'section' => 'menu',
   'subtype' => 'menu',
   'subtype_params' => array(
      'selector' => '.landing-block-node-menu-list-item-link',
      'count' => 5,
      'source' => 'catalog',
   )
),
```

Где:

- **selector** - селектор непосредственно тега пункта меню (тег a).
- **count** - ограничение в количестве добавляемых динамических пунктов (обязательно, 5 по-умолчанию).
- **source** - источник данных (catalog: меню текущего каталога магазина; structure: структура текущих папок и страниц, работает только при наличии ключа menu в манифесте).

Обратите внимание, что если блок меню будет добавляться в обычном лендинге (не магазин) данный подтип будет игнорироваться.

Карты в блоках

Чтобы работать с картами (пока поддерживаются карты Google) и иметь весь функционал по их редактированию, необходимо:

1. Разместить два ключа в секции **block**: subtype и subtype_params (подробнее смотрите пример манифеста).
2. Указать расширение **landing_google_maps_new** (смотрите assets в примере манифеста).
3. Указать необходимой ноде (где будет карта) тип **map**.

Пример манифеста

```
return [
   'block' => [
      'name' => 'Карта Google',
      'section' => ['contacts'],
      'subtype' => 'map',
      'subtype_params' =>[
         'required' => 'google'
      ],
   ],
   'cards' => [],
   'nodes' => [
      '.landing-block-node-map' => [
         'name' => 'Map',
         'type' => 'map',
      ]
   ],
   'style' => [
```

```
        'block' => [
            'type' => ['block-default-wo-
background-vh-animation']
        ],
        'nodes' => [],
    ],
    'assets' => [
        'ext' => ['landing_google_maps_new'],
    ]
];
```

Пример блока для данного манифеста:

```
<section class="landing_block g-pt-0 g-pb-0
g-height-70vh">
    <div class="landing-block-node-map h-
100" data-map></div>
</section>
```

Примеры блоков данного типа вы можете посмотреть в нашем репозитории, воспользовавшись методами landing.block.getmanifestfile и landing.block.getrepository. Их коды:

- 16.3.two_cols_map_text_fix
- 16.4.three_cols_map
- 16.5.two_cols_map
- 16.6.two_cols_map_reverse
- 16.1.google_map
- 16.2.two_cols_text_map_fix
- и многие другие

Навигация и заголовок

Для данного типа блока не требуется указывать каких-то специальных параметров в манифесте. Любой блок может стать навигационной цепочкой или заголовком. Достаточно наличие маркеров в контенте блока:

Маркер: #title#

Заменяется на заголовок текущей страницы, который прежде всего зависит от названия страницы, но может быть переопределен компононентами внутри блоков страницы, или сторонним внешним решением.

Пример блока с заголовком:

```
<section class="landing-block g-pt-20 g-pb-20">
   <div class="landing-title-container container g-font-size-12">
      #title#
   </div>
</section>
```

Маркер: #breadcrumb#

Данный маркер заменяется на навигационную цепочку – последовательность ссылок от главной страницы к текущей. Как правило, реальная польза цепочки появляется только в случае динамического контента и разветвленной структуры.

Влиять на верстку непосредственно самой последовательности ссылок в облачной версии нет возможности.

Пример блока с навигационной цепочкой:

```
<section class="landing-block g-pt-20 g-pb-20">
   <div class="landing-breadcrumb-container container g-font-size-12">
      #breadcrumb#
   </div>
</section>
```

Результаты поиска

Данный блок является дочерним для блока Форма поиска. Все что требуется для работы такого блока это выбрать его страницу в настройках формы поиска, а в текущем блоке результатов включить динамические карточки и выбрать источник "Страницы сайта".

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Формы в блоках

Описание

В блоки достаточно легко встроить [dw]формы Битрикс24 (CRM)[/dw][di][/di]. Для этого сделайте следующее:

1. В секции **block** манифеста блока добавьте ключ **subtype** со значением `form`.
2. Разместите `div` с классом `bitrix24forms` внутри вашего блока. Там и будет выводиться форма.
3. Добавьте ключ ext (внутри assets) со значением landing_form, который подключит все необходимое для работы форм.

```
'assets' =>
        array (
            'ext' => array (
                    'landing_form'
            ),
    ),
```

Разметка

Нода, в которой появится форма, должна быть пустой. Её необходимо пометить классом **.bitrix24forms**. Так же можно добавить два необязательных параметра:

- **data-b24form-use-st yle="Y"** - использовать стилизацию блока (Y), или выводить оригинальный вид формы (N),

- **data-b24form-show-header="N"** - скрывать или показывать заголовок формы.

Эти параметры могут быть изменены в настройках блока.

Принципы разметки

Формы встраиваются в `<iframe>`, то есть формально они отображаются не в Сайтах, а на стороннем ресурсе. При такой схеме нет возможности повлиять на вид форм снаружи, со стороны блока. Однако при инициализации форм есть возможность передать им массив CSS-стилей, которые применятся к форме (внутри `<iframe>`). Именно таким образом работают блоки, изменяя внешний вид форм и идеально вписывая их в свой дизайн.

Постройте группу стилей типа "основной цвет, основной фон, дополнительный цвет, главный цвет рамок" и тому подобные. Для этого выберите в блоке ноду, имеющую заданный цвет и пометить её дата-атрибутом, данные атрибуты начинаются с префикса **data-form-style-**. Поскольку в блоке может не быть элементов с нужным, например, цветом, можно добавлять скрытые блоки с нужными параметрами. В общих чертах это выглядит так:

```
<div hidden>
<div class="g-bg-primary g-color-primary g-
brd-primary"
data-form-style-main-bg="1"
data-form-style-main-border-color="1"
data-form-style-main-font-color-hover="1"
>
</div>
</div>
```

В примере добавляется блок, имеющий primary-фоновый цвет, primary-цвет шрифта и primary-цвет рамок. А дата-атрибутами устанавливаются значения для главного цвета/фона/цвета рамок уже внутри формы.

Внимание, если нода, отмеченная **data-form-style-...** доступна для дизайна пользователям (например, заголовок

блока, который можно перекрашивать), то изменения цвета будут интерактивно проброшены в форму.

Всего таких атрибутов довольно много и описывать их назначение довольно сложно. Поэтому рекомендуется в качестве примера взять один из штатных блоков и менять в нём настройки, следя за изменениями.

Пример блока:

```
<section class="g-pos-rel landing-block
text-center g-pt-80 g-pb-80 g-bg-primary">
   <div class="container">
      <div class="landing-block-form-styles"
hidden>
         <div class="g-bg-transparent h1 g-
color-white g-brd-none g-pa-0"
             data-form-style-wrapper-
padding="1"
             data-form-style-bg="1"
             data-form-style-bg-content="1"
             data-form-style-bg-block="1"
             data-form-style-header-font-
size="1"
             data-form-style-main-font-
weight="1"
             data-form-style-border-
block="1"
         >
         </div>

         <div class="g-bg-white g-color-
primary g-brd-primary"
             data-form-style-main-bg="1"
             data-form-style-main-border-
color="1"
             data-form-style-main-font-
color-hover="1"
```

```
        >
        </div>
        <div class="g-bg-primary-dark-v2 u-
theme-restaurant-shadow-v1 g-brd-around g-
color-gray-dark-v2 rounded-0"
            data-form-style-input-bg="1"
            data-form-style-input-box-
shadow="1"
            data-form-style-input-select-
bg="1"
            data-form-style-input-
border="1"
            data-form-style-input-border-
radius="1"
            data-form-style-button-font-
color="1"
        >
        </div>
        <div class="g-brd-around g-brd-
gray-light-v2 g-brd-bottom g-bg-black-
opacity-0_7"
            data-form-style-input-border-
color="1"
            data-form-style-input-border-
hover="1"
        >
        </div>

        <p class="g-color-white-opacity-
0_7"
           data-form-style-second-font-
color="1"
           data-form-style-main-font-
family="1"
           data-form-style-main-font-
weight="1"
           data-form-style-header-text-
```

```
font-size="1">
        </p>

        <h3 class="g-font-size-11 g-color-
white"
           data-form-style-label-font-
weight="1"
           data-form-style-label-font-
size="1"
           data-form-style-main-font-
color="1"[DISK FILE ID=810652][DISK FILE
ID=810656]
        >
        </h3>

        <!--         for resource booking--
>
        <div class="g-bg-white"
            data-form-style-bg-as-text="1"
        >
        </div>

        <div class="g-bg-primary-dark-v2"
            data-form-style-input-bg-
light="1"
        >
        </div>

        <div class="g-bg-primary-dark-v3"
            data-form-style-input-bg-
light2="1"
        >
        </div>

        <div class="g-bg-primary u-shadow-
custom-v2"
            data-form-style-input-bg-
```

```
light3="1"
            data-form-style-gradient-box-
shadow="1"
        >
        </div>

        <div class="g-bg-primary-opacity-
0_4"
            data-form-style-main-bg-
light="1"
        >
        </div>
      </div>

      <div class="row">
        <div class="col-md-6 mx-auto">
           <div class="bitrix24forms g-brd-
white-opacity-0_6 u-form-alert-v3"
               data-b24form-use-st yle="Y"
               data-b24form-show-header="N"
           ></div>
        </div>
      </div>
   </div>
</section>
```

Пример

Примеры блоков данного типа вы можете посмотреть в нашем репозитории, воспользовавшись методами landing.block.getmanifestfile и landing.block.getrepository. Их коды:

- 33.1.form_1_transparent_black_left_text
- 33.10.form_2_light_left_text
- 33.23.form_2_themecolor_no_text
- и многие другие

Простой пример:

```
// пример регистрации простейшей формы
BX24.callMethod(
   'landing.repo.register',
   {
      "code":"test_form",
      "fields":{"NAME":"Test form",
         "SECTIONS":"other",

"PREVIEW":"https://restapi.bx24.net/booking/
cycles_b24.jpg",
         "CONTENT":"<div
class=\"bitrix24forms\"></div>"
      },
      "manifest": {
         "block":{"subtype":"form"},
         "assets":{"ext":["landing_form"]}
      }
   },
   function(result)
   {
      if(result.error())
         console.error(result.error());
      else
         console.info(result.data());
   }
);
```

Формы поиска

Такие блоки содержат форму со строкой поиска и кнопкой отправки данных на страницу результатов поиска.

Минимальные условия корректного функционирования такого блока:

1. Наличие тега `<form>`, строки ввода запроса `<input>`, кнопки отправки данных.
2. Наличие атрибута для селектора `<form>`.

```
'attrs' => [
   '.landing-block-node-form' => [
      'name' => 'Search result page',
      'attribute' => 'action',
      'type' => 'url',
      'allowedTypes' => [
         'landing',
      ],
      'disableCustomURL' => true,
      'disallowType' => true,
      'disableBlocks' => true
   ]
]
```

3. По желанию к описанию блока можно добавить **subtype** и **subtype_params**. В таком случае в атрибут **Search result page** (см. п. 2) будет подставлена страница при добавлении блока (что удобно для пользователя):

```
'block' => [
   'name' =>
Loc::getMessage('LANDING_BLOCK_59_2-
NAME'),
   'section' => array('sidebar',
'other'),
   'subtype' => 'search',
   'subtype_params' => [
      'type' => 'form',
      'resultPage' => 'result'
   ]
],
```

Где **result** код шаблона страницы результатов поиска. Если такая страница будет найдена на сайте, она добавится автоматически.

Введение

Инструкция по созданию интерактивных блоков.

В Битрикс.Сайтах есть возможность создавать интерактивные блоки нескольких типов. В общем случае для этого требуется:

- обеспечить соответствующую вёрстку;
- подключить js-расширение в манифесте блока;
- в зависимости от типа указать редактируемые ноды или подтип (subtype) блока.

В разделе описано подключение по каждому из типов интерактивных блоков.

Конечно, у вас есть возможность создать любой из такого типа блоков по своим правилам. В таком случае они не будут интерактивными в редакторе, и вам придется самим писать клиентские скрипты. В случае задокументированных возможностей система все сделает за вас.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Галереи

[dw]Галерея[/dw][di][/di] создаётся с использованием стандартных нод-картинок и карточки. В манифесте блока укажите расширение **landing_gallery_cards**.

```
'assets' => array(
   'ext' => array('landing_gallery_cards'),
),
```

В разметке обозначьте контейнер классом **.js-gallery-cards**, внутри него добавьте необходимое количество нод **<img>**. Каждому изображению добавьте атрибут **data-fancybox="gallery"**. Этот служебный параметр может иметь любое значение, кроме пустого.

Галерея имеет только одну версию изображения, а не миниатюру и полную, как обычно. Поэтому используйте картинки достаточного размера или масштабируйте их средствами браузера (ограничивать ширину/высоту). Скрипт галереи обернёт каждую картинку в ссылку и по клику будет открывать изображение, указанное в **src**.

Опционально допускается атрибут **data-link-classes="d-block g-pos-rel"**.

Оба класса добавляются к ссылке-обёртке вокруг изображения, они необходимы для вёрстки.

Галереи могут совмещаться с другими возможностями. Например, картинки могут быть карточками, чтобы клиент добавлял столько, сколько нужно. Или же это может быть слайдер, каждая из картинок которого может открываться в большом размере. Примеры можно увидеть в наших стандартных блоках.

Важно! При совмещении галереи и карусели (слайдера) нужно инициализировать ассеты в определённом порядке: сначала карусель, затем - галерею! Другие ассеты, при их наличии, могут идти в любой последовательности. Смотри код ниже.

```
'assets' => [
   'ext' => ['landing_carousel',
'landing_gallery_cards'],
],
```

Пример

Примеры блоков данного типа вы можете посмотреть в нашем репозитории, воспользовавшись методами landing.block.getmanifestfile и landing.block.getrepository. Их коды:

- 32.11.img_grid_4cols_4
- 32.7.img_grid_4cols_2
- 45.2.gallery_app_with_slider - со слайдером
- и многие другие

Пример простой галереи:

```
<div class="landing-block g-pt-80 g-pb-80">
   <div class="container">
      <div class="js-gallery-cards row">
         <div class="landing-block-node-card
js-animation slideInUp text-center col-lg-3
col-md-4 col-sm-6 g-mb-30">
            <div class="g-pos-rel d-inline-
block">
               <img class="landing-block-
node-card-img h-100 g-width-auto g-max-
width-100x g-max-height-350 g-max-height-
```

```
500--md"

src="https://cdn.bitrix24.site/bitrix/images
/landing/business/270x481/img1.jpg"
                  data-fancybox="gallery"
                  data-link-classes="d-
block g-pos-rel" alt=""/>
            </div>
         </div>

         <div class="landing-block-node-card
js-animation slideInUp text-center col-lg-3
col-md-4 col-sm-6 g-mb-30">
            <div class="g-pos-rel d-inline-
block">
               <img class="landing-block-
node-card-img h-100 g-width-auto g-max-
width-100x g-max-height-350 g-max-height-
500--md"

src="https://cdn.bitrix24.site/bitrix/images
/landing/business/270x481/img2.jpg"
                  data-fancybox="gallery"
                  data-link-classes="d-
block g-pos-rel" alt=""/>
            </div>
         </div>

         <div class="landing-block-node-card
js-animation slideInUp text-center col-lg-3
col-md-4 col-sm-6 g-mb-30">
            <div class="g-pos-rel d-inline-
block">
               <img class="landing-block-
node-card-img h-100 g-width-auto g-max-
width-100x g-max-height-350 g-max-height-
500--md"
```

```
src="https://cdn.bitrix24.site/bitrix/images
/landing/business/270x481/img3.jpg"
                   data-fancybox="gallery"
                   data-link-classes="d-
block g-pos-rel" alt=""/>
            </div>
         </div>

         <div class="landing-block-node-card
js-animation slideInUp text-center col-lg-3
col-md-4 col-sm-6 g-mb-30">
            <div class="g-pos-rel d-inline-
block">
               <img class="landing-block-
node-card-img h-100 g-width-auto g-max-
width-100x g-max-height-350 g-max-height-
500--md"

src="https://cdn.bitrix24.site/bitrix/images
/landing/business/270x481/img4.jpg"
                   data-fancybox="gallery"
                   data-link-classes="d-
block g-pos-rel" alt=""/>
            </div>
         </div>

      </div>
   </div>
</div>
```

Слайдеры

Описание

В манифесте блока подключите расширение **landing_carousel**.

```
'assets' => array(
   'ext' => array('landing_carousel'),
)
```

В разметке блока пометьте классами ноды:

- **.js-carousel** - корневой контейнер слайдера
- **.js-slide** - каждый слайд в отдельности

По умолчанию показывается по 1 слайду за раз, каждый слайд занимает всю ширину контейнера. Кнопки переключения и индикаторы отсутствуют. Автоматическая прокрутка выключена. Всё это изменяется настройками, которые задаются data-атрибутами. Атрибуты нужно добавлять к элементу **.js-carousel**.

[dw]Вид слайдера[/dw][di][/di]

Важно! При совмещении галереи и карусели (слайдера) нужно инициализировать ассеты в определённом порядке: сначала карусель, затем - галерею! Другие ассеты, при их наличии, могут идти в любой последовательности. Смотри код ниже.

```
'assets' => [
   'ext' => ['landing_carousel',
'landing_gallery_cards'],
],
```

Атрибуты

Кнопки перелистывания.

Атрибут добавляет кнопки перелистывания. Стили задаются как общие для обеих кнопок, а так же отдельно для левой и правой.

```
data-arrows-classes="u-arrow-v1 g-absolute-
centered--y g-width-45 g-height-45 g-color-
white g-bg-primary"
data-arrow-left-classes="fa fa-chevron-left
g-left-0"
data-arrow-right-classes="fa fa-chevron-
right g-right-0"
```

Индикаторы страницы (пагинация)

Атрибут добавляет элемент пагинации, задаёт его классы.

```
data-pagi-classes="u-carousel-indicators-v1
g-absolute-centered--x g-bottom-60 text-
center"
```

Количество слайдов на экране

```
data-slides-show="3"
```

Количество слайдов, меняющихся за одно перелистывание

```
data-slides-scroll="2"
```

Включение/выключение автопрокрутки

```
data-autoplay="true"
```

Скорость автопрокрутки в миллисекундах

```
data-speed="1000"
```

Остановить автопрокрутку при наведении мыши

```
data-pause-hover="true"
```

Эффект "появления" слайдов

Атрибут позволяет не перелистывать слайды, а менять их местами с изменением прозрачности.

```
data-fade="true"
```

Внимание! Корректно работает только с data-slides-show="1"

Вертикальный слайдер

```
data-vertical="true"
```

Будьте внимательны, кнопки и пагинация для вертикальных слайдеров должны отличаться от горизонтальных (располагаться по-другому). Примеры можно увидеть в штатных блоках. Рекомендуем выключать вертикальность на мобильных устройствах с помощью настройки **Адаптивность**. В противном случае, скролл пальцем по экрану будет не двигать страницу, а листать слайды.

Количество строк

```
data-rows="2"
```

В мультистроковом слайдере параметры **data-slides-show** и **data-slides-scroll** влияют не на количество слайдов, а на количество колонок.

Проигрывание по кругу

Если включено, то после последнего слайда снова будет показан первый. Если выключено, то проигрывание остановится. Для совместимости с редактором, эта настройка работает только в режиме Предпросмотра и Публикации. В редакторе зацикливание всегда выключено.

```
data-infinite="true"
```

Адаптивность

Слайдеры могут гибко менять свои настройки в зависимости от размера экрана. Адаптивности могут быть подвержены любые вышеперечисленные настройки, но чаще всего меняется **Количество слайдов на экране**.

В атрибуте необходимо передать массив объектов, каждый из которых должен содержать:

- breakpoint - размер экрана в пикселях. Правило применяется "вниз", то есть для экранов данного размера и меньше.
- settings - массив настроек, применяемых для данного правила. Имена настроек отличаются от имён дата-атрибутов. Список имён для ранее приведённых атрибутов:
    - **arrowsClasses**
    - **prevArrow**
    - **nextArrow**
    - **dotsClass**
    - **slidesToShow**
    - **slidesToScroll**
    - **autoplay**
    - **autoplaySpeed**
    - **pauseOnHover**
    - **fade**
    - **vertical**

```
data-responsive='[{
   "breakpoint": 1200,
   "settings": {
      "slidesToShow": 5
   }
}, {
   "breakpoint": 992,
   "settings": {
      "slidesToShow": 3
   }
}, {
   "breakpoint": 768,
   "settings": {
      "slidesToShow": 2
   }
}, {
   "breakpoint": 576,
```

```
    "settings": {
        "slidesToShow": 1
    }
}]'
```

Пример

Примеры блоков данного типа вы можете посмотреть в нашем репозитории, воспользовавшись методами landing.block.getmanifestfile и landing.block.getrepository. Их коды:

- 01.big_with_text
- 01.big_with_text_blocks
- 28.5.team_4_cols_slider
- 39.1.five_blocks_carousel
- 45.2.gallery_app_with_slider - с галереей
- и многие другие

Простой пример:

```
<div class="js-carousel"
    data-arrows-classes="u-arrow-v1 g-
absolute-centered--y g-width-45 g-height-45
g-color-white g-bg-primary"
    data-arrow-left-classes="fa fa-chevron-
left g-left-0"
    data-arrow-right-classes="fa fa-chevron-
right g-right-0"
    data-pagi-classes="u-carousel-
indicators-v1 g-absolute-centered--x g-
bottom-60 text-center"
    data-slides-show="3"
    data-slides-scroll="2"
    data-autoplay="true"
    data-speed="1000"
```

```
    data-pause-hover="true"
    data-responsive='[
      {
         "breakpoint": 768,
         "settings": {
            "slidesToShow": 2
         }
      }, {
         "breakpoint": 576,
         "settings": {
            "slidesToShow": 1
         }
      }
   ]'
>

   <div class="js-slide g-height-50vh g-brd-
gray-light-v3 g-brd-around g-bg-primary-
opacity-0_1">
      <div class="g-flex-centered w-100 h-
100">
         <h3>Slide 1</h3>
      </div>
   </div>

   <div class="js-slide g-height-50vh g-brd-
gray-light-v3 g-brd-around g-bg-primary-
opacity-0_1">
      <div class="g-flex-centered w-100 h-
100">
         <h3>Slide 2</h3>
      </div>
   </div>

   <!-- ... and other slides ... -->

</div>
```

Счетчики обратного отсчёта

Описание

[dw]Пример счётчика[/dw][di][/di]

В манифесте блока добавьте расширение **landing_countdown**.

```
'assets' => array(
   'ext' => array('landing_countdown'),
),
```

В разделе **block** манифеста добавьте ключ:

```
'version' => '18.5.0'
```

Параметр version необязателен, но он ограничит добавления данного блока в версии ранее указанной, когда ещё не существовало нужных asset'ов.

Добавьте атрибуты для ноды-контейнера счётчика к той ноде, которая соответствует описанию и должна выполнять роль счетчика. За подробностями смотри пример.

```
'attrs' => array(
   '.landing-block-node-date' => array(
```

```
        'name' =>
Loc::getMessage('LANDING_BLOCK_51_2_COUNTDOW
N_04--DATE'),
        'time' => true,
        'type' => 'date',
        'format' => 'ms',
        'attribute' => 'data-end-date',
    ),
),
```

Разметка

Таймер должен содержать 4 цифровых элемента, помеченных соответствующими классами:

- **js-cd-days** - дни
- **js-cd-hours** - часы
- **js-cd-minutes** - минуты
- **js-cd-seconds** - секунды

Возможность добавлять год отсутствует, мы считаем нецелесообразным создавать столь долгие таймеры на сайте.

Цифры оборачивайте в общий контейнер, помеченный классом **js-countdown**. Этому же контейнеру передавайте настройки посредством дата-атрибутов.

- **data-end-date="1586690955000"** - дата окончания таймера в формате Unix-time в миллисекундах. Т.е. полученную unix-дату нужно умножить на 1000.
- **data-days-format="%D"** - формат представления дней
- **data-hours-format="%H"** - формат представления часов
- **data-minutes-format="%M"** - формат представления минут
- **data-seconds-format="%S"** - формат представления секунд

Доступно два вариант формата:

"%S" - выводит число с лидирующими нулями "03", но "18",
"%-S" - выводит только значимые символы "3" или "18".

Вместо **"%H"** можно использовать **"%I"** или **"%-I"**. Это значение полного количества часов до окончания (то есть, 1 день 6 часов превращается в 30 часов). В этом случае нужно удалить data-days-format и ноду .js-cd-days.

Необязательный параметр:

```
data-days-expired-classes="u-countdown--days-expiried"
```

Когда количество дней станет равным нулю, таймер может добавить себе помечающий класс. Это поможет вам скрыть нулевое количество дней, либо как-то выделить их. Мы используем для этого класс **.u-countdown--days-hide**.

Пример

Примеры блоков данного типа вы можете посмотреть в нашем репозитории, воспользовавшись методами landing.block.getmanifestfile и landing.block.getrepository. Их коды:

- 51.2.countdown_04
- 51.3.countdown_08
- 51.3.countdown_08_wo_bg
- 51.4.countdown_music
- 51.5.countdown_event
- 51.7.countdown_13
- 51.1.countdown_01

Пример простого таймера

```
<section class="landing_block g-pt-30 g-pb-30 g-bg-orange g-color-white">
   <div class="landing-block-node-date mx-auto js-countdown text-center g-font-weight-300 g-line-height-1-2"
       data-end-date="1555249081000"
       data-days-format="%D"
```

```
       data-hours-format="%H"
       data-minutes-format="%M"
       data-seconds-format="%S"
       data-days-expired-classes="u-
countdown--days-expiried"
   >

      <div class="landing-block-node-number
u-countdown--days-hide d-inline-block g-mx-
20">
         <div class="landing-block-node-
number-number g-font-size-36 mb-0">
            <span class="js-cd-
days">12</span>
         </div>
      </div>

      <div class="landing-block-node-number-
delimiter u-countdown--days-hide d-inline-
block g-font-size-36">:</div>

      <div class="landing-block-node-number
d-inline-block g-mx-20">
         <div class="landing-block-node-
number-number g-font-size-36 mb-0">
           <span class="js-cd-
hours">01</span>
         </div>
      </div>

      <div class="landing-block-node-number-
delimiter d-inline-block g-font-size-36">:
</div>

      <div class="landing-block-node-number
d-inline-block g-mx-20">
         <div class="landing-block-node-
```

```
number-number g-font-size-36 mb-0">
            <span class="js-cd-
minutes">52</span>
         </div>
      </div>

      <div class="landing-block-node-number-
delimiter d-inline-block g-font-size-36">:
</div>

      <div class="landing-block-node-number
d-inline-block g-mx-20">
         <div class="landing-block-node-
number-number g-font-size-36 mb-0">
            <span class="js-cd-
seconds">52</span>
         </div>
      </div>

   </div>
</section>
```

landing.block.clonecard

```
landing.block.clonecard(lid, block,selector)
```

Метод для клонирования карточки блока. Возвращает *true* или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	
selector	Селектор карточки, взятый с манифеста, с добавленным идентификатором карточки. Например: `'.landing-block-card@0'`. 0 в конце означает, что воздействуем на первую по порядку карточку. Если знак @ и число за ним отсутствует, вставка будет произведена в начало родительского контейнера карточек.	

Обратите внимание, что как только вы склонировали карточку, их счетчики поменялись.

Смотри также

landing.block.addcard - Метод полностью повторяет работу landing.block.clonecard но дает возможность вставить карточку сразу с измененным контентом.

Примеры

```
BX24.callMethod(
        'landing.block.cloneCard',
        {
                lid: 311,
                block: 6057,
                selector: '.landing-block-
card@0'
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.block.removecard

```
landing.block.removecard (lid,
block,selector)
```

Метод для удаления карточки блока. Возвращает *true* или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	
selector	Селектор карточки, взятый с манифеста, с добавленным идентификатором карточки. Например: `'.landing-block-card@0'`. 0 в конце означает, что воздействуем на первую по порядку карточку.	

Обратите внимание, что как только вы удалили карточку, их счетчики поменялись.

Пример

```
BX24.callMethod(
        'landing.block.removecard',
        {
                lid: 311,
                block: 6057,
                selector: '.landing-block-
card@0'
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.block.updatenodes

Описание

```
landing.block.updatenodes(lid, block,data)
```

Метод для изменения контента блока. Возвращает *true* или ошибку. Также метод применяется для обновления параметров динамических блоков, таких как список товаров, детальный товар, и некоторых других.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	
data	Массив селекторов и новых значений. Например: `data: {'.landing-block-node-text@1': 'new text!'}`. Принцип тот же - селектор и его новое значение. Если вы уверены, что селектор в блоке один, то в данном случае вы можете отбросить счетчик **@1**.	

	Также data зависит от типов изменяемых нод. Подробнее смотрите пример ниже, за описанием типов обращайтесь к отдельной странице.	

Пример

```
BX24.callMethod(
    'landing.block.updatenodes',
    {
        lid: 311,
        block: 6058,
        data: {
           '.landing-block-node-text':
'Текст, c html',
           '.landing-block-node-img': {src:
'/some/path/picture.png', alt: 'Моя
картинка'},
           '.landing-block-node-link':
{text: 'Моя ссылка', href:
'https://bitrix24.com', target: '_blank'},
           '.landing-block-node-icon': ['fa-
telegram', 'fa-skype'],
           '.landing-block-node-embed':
{src: '//www.youtube.com/embed/q4d8g9Dn3ww?
autoplay=1&controls=0&loop=1&mute=1&rel=0',
source: 'https://www.youtube.com/watch?
v=q4d8g9Dn3ww'},
       },
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
```

```
        else
        {
            console.info(result.data());
        }
    }
);
```

Редактирование параметров динамических блоков

Есть ряд динамических блоков, параметры которых можно менять через REST. Например, количество товаров на странице. Сделать это можно следующим образом.

1. Посредством метода landing.block.getmanifest узнаем какие параметры у блока есть. Метод вернет массив манифеста, где нас интересует ключ attrs и параметры интересующего вас компонента динамического блока. В данном случае нам интересен bitrix:catalog.section.

```
attrs:
bitrix:catalog.section: Array(24)
0: {name: "ID раздела", style: false,
original_type: "component",
component_type: "STRING", attribute:
"SECTION_ID", …}
1: {name: "Недоступные товары", style:
false, original_type: "component",
component_type: "LIST", attribute:
"HIDE_NOT_AVAILABLE", …}
2: {name: "Недоступные торговые
предложения", style: false,
original_type: "component",
component_type: "LIST", attribute:
"HIDE_NOT_AVAILABLE_OFFERS", …}
3: {name: "По какому полю сортируем
```

```
элементы", style: false, original_type:
"component", component_type: "LIST",
attribute: "ELEMENT_SORT_FIELD", …}
4: {name: "Порядок сортировки элементов",
style: false, original_type: "component",
component_type: "LIST", attribute:
"ELEMENT_SORT_ORDER", …}
5: {name: "Валюта, в которую будут
сконвертированы цены", style: false,
original_type: "component",
component_type: "LIST", attribute:
"CURRENCY_ID", …}
6: {name: "Тип цены", style: false,
original_type: "component",
component_type: "LIST", attribute:
"PRICE_CODE", …}
...
```

2. Через метод landing.block.updatenodes изменяем необходимые параметры. Так сложилось исторически, что динамические параметры (атрибуты) изменяются именно через этот метод

```
BX24.callMethod(
   'landing.block.updatenodes',
   {
      lid: 5597,
      block: 44131,
      data: {
         'bitrix:catalog.section': {
            attrs: {
               'MESS_BTN_BUY': 'Add to my
cart'
            }
         }
      },
      function(result)
```

```
        {
            if (result.error())
            {

console.error(result.error());
            }
            else
            {
                console.info(result.data());
            }
        }
    }
);
```

landing.block.updateattrs

Описание

```
landing.block.updateattrs(lid, block,data)
```

Метод для изменения атрибутов ноды блока. Возвращает *true* или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	
data	Массив селектора и новых значений дата-атрибутов. Например, `data: {'.bitrix24forms': {'data-b24form': 'tratrata'}}`. Манифест должен содержать в себе атрибуты, которые вы хотите изменять таким образом.	

Если атрибут относится в карточке (то есть может иметь различное содержимое от карточки к карточке, селектор необходимо

передавать с разделителем @:

```
data: {
    '.container-fluid@1': {//влияние
произойдет на атрибут второй карточки
(отсчет от нуля)
        'data-test-checkbox': [1, 2, 3]
    }
}
```

Типы изменяемого контента

Каждый тип атрибута обладает тем или иным форматом сохранения. В примерах даны значения по-умолчанию для каждого типа. Передача нового значения происходит по аналогичному формату. Например, сохранение в атрибут типа **image**:

```
        data: {
            '.container-fluid': {
                'data-test-image': {src:
'https://i.img.com/images/i/291626458734-0-
1/s-l1000.jpg', alt: 666}
            }
        }
```

Отдельные пояснения для типа **checkbox** и **multiselect**: для сохранения нового значения необходимо отправлять значения выделенных элементов:

```
        data: {
            '.container-fluid': {
                'data-test-checkbox': [1, 2,
```

```
3]
            }
        }
```

Редактирование параметров динамических блоков производится через метод landing.block.updatenodes.

Пример

```
BX24.callMethod(
        'landing.block.updateattrs',
        {
                lid: 313,
                block: 6134,
                data: {
                        '.bitrix24forms': {
                                'data-
b24form': 'tratrata'
                        }
                }
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
```

```
            }
);
```

landing.block.addcard

```
landing.block.addcard(lid, block, selector,
content)
```

Метод полностью повторяет работу landing.block.clonecard но дает возможность вставить карточку сразу с измененным контентом.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	
selector	Селектор карточки, взятый с манифеста, с добавленным идентификатором карточки. Например: `'.landing-block-card@0'`. 0 в конце означает, что воздействуем на первую по порядку карточку.	
content	Содержимое новой карточки.	

Обратите внимание, что как только вы склонировали карточку, их счетчики поменялись.

Примеры

```
BX24.callMethod(
   'landing.block.addCard',
   {
      lid: 634,
      block: 12079,
      selector: '.landing-block-node-menu-
list-item@0',
      content: '
•  ' +
                 'New card item' +
            '
'
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

landing.block.changeAnchor

```
landing.block.changeAnchor(
   lid,
   block,
   data
)
```

Метод изменяет символьный код якоря. Штатно якорь выглядит следующим образом: `#block12345`, где 12345 – идентификатор блока.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока.	
data	Символьный код якоря.	

Пример

```
BX24.callMethod(
   'landing.block.changeAnchor',
   {
      lid: 3496,
      block: 29356,
      data: 'about'
   },
   function (result)
   {
      if (result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

landing.block.changeNodeNam

```
landing.block.changeNodeName(lid,
block,data)
```

Метод изменяет название тега. Например, тег h3 требуется поменять на тег h1. Вернет *true* в случае успеха, или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	
data	Массив селекторов и новых значений. Подробнее смотрите пример. Селектор может передаваться как без указания позиции (например, `.landing-block-node-text`), тогда будут изменены все карточки по данному селектору. Так и с указанием позиции (например, `.landing-block-node-text@2`), тогда будет изменена только	

	карточка на указанной позиции (отсчет с нуля).	

Пример

```
BX24.callMethod(
   'landing.block.changeNodeName',
   {
      lid: 2006,
      block: 20476,
      data: {
         '.landing-block-node-small-
title@0': 'i',
         '.landing-block-node-small-
title@1': 'u'
      }
   },
   function (result)
   {
      if (result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс»

landing.block.getcontent

```
landing.block.getcontent(
   lid,
   block,
   editMode,
   params
)
```

Метод для получения контента блока. Возвращает массив содержимого блока - html, файлы стилей и JS. Или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	
editMode	Режим редактирования (1) или нет (0), вернется разный набор блоков	
params	Массив дополнительных параметров. На данный момент поддерживает один ключ – **wrapper_show** – показывать	

	ли обрамляющий системный div (0, 1). По умолчанию - показывать.	

Пример

```
BX24.callMethod(
   'landing.block.getContent',
   {
      lid: 4858,
      block: 39556,
      editMode: 1,
      params: {
         wrapper_show: 0
      }
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

landing.block.getContentFromR

```
landing.block.getContentFromRepository(
   $code
)
```

Метод получает контент блока из репозитория "как есть" до добавления блока на какую-либо страницу.

Параметры

Параметр	Описание	С версии
code	код блока	

Пример

```
BX24.callMethod(
   'landing.block.getContentFromRepository',
   {
      code: '28.6.team_4_cols'
```

```
    },
    function(result)
    {
       if(result.error())
          console.error(result.error());
       else
          console.info(result.data());
    }
);
```

landing.block.updateCards

```
landing.block.updateCards(lid, block,data)
```

Метод для массового изменения карточек блока. Вернет *true* в случае успеха, или ошибку.

Внимание! 1. Метод полностью удалит текущие карточки блока.
2. Метод специфический и рекомендуется к применению только если ваши задачи не решает landing.block.updatenodes.

Параметры

Параметр	Описание
lid	Идентификатор страницы.
block	Идентификатор блока.
data	Массив для изменения. Для пояснения смотрите пример. Опционально можно передавать пресеты карточек. Обратите внимание, селекторы формируются по похожему методу формирования в *landing.block.updatenodes*.

Пример

```
BX24.callMethod(
   'landing.block.updateCards',
   {
      lid: 2856,
      block: 25458,
      data: {
         //воздействуем на данный селектор
карточки
         // (можно передавать и другие
селекторы одновременно)
         '.landing-block-card': {
            //останется только данное кол-во
карточек, у которых
            //будут изменены только
указанные ноды;
            //для клонирования будет браться
первая карточка
            'values': [
               {
                  '.landing-block-node-
title': 'New title 0'
               },
               {
                  '.landing-block-node-
title': 'New title 1'
               },
               {
                  '.landing-block-node-
title': 'New title 2'
               }
            ],
            //опционально можно применить
пресеты карточек (ключ - порядковый номер
карточки, начиная с 0)
```

```
                'presets': {
                   '1': 'preset_h2'
                }
            }
        }
    },
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.block.updatecontent

```
landing.block.updatecontent(lid, block, content)
```

Метод обновляет содержимое уже размещенного на странице блока на любой произвольный. Для изменения контентной части рекомендуется метод landing.block.updatenodes. Вернет *true* в случае успеха, или ошибку.

Важно! - Если новая разметка блока не будет согласовываться с его текущим манифестом, блок может оказаться не редактируемым.
- Контент пропускается через санитайзер, который может удалить некоторые подозрительные атрибуты и теги.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	
content	Новый контент блока	
designed	Необязательный, по умолчанию	

	false. Если передать *true*, то блок будет считаться как заблокированный к изменению штатным апдейтером системы.	

Атрибут **style** может вырезаться встроенным санитайзером. Чтобы это обойти используйте вместо него атрибут **bxstyle**. При добавлении система конвертирует его в штатный style.

Пример

```
BX24.callMethod(
   'landing.block.updatecontent',
   {
      lid: 625,
      block: 11883,
      content: '<h3>My super content</h3>'
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

landing.block.updateStyles

```
landing.block.updateStyles(lid, block,data)
```

Метод для изменения стилей блока. Возвращает *true* или ошибку.

Параметры

Параметр	Описание
lid	Идентификатор страницы
block	Идентификатор блока
data	В параметре передается массива ключ-значение, где ключом идет селектор, а каждым значением указывается два массива:: ▪ **classList** - какие классы добавить в изменяемый селектор. ▪ **affect** - передаются стили, которые надо обнулить у всех дочерних нод. Например, передаётся класс, который окрашивает элемент в цвет (color). Значит в affect надо передать массив [color], чтобы система обнулила все color у дочерних. Иначе будет такая ситуация - цвет родителя стоит красный, а текст внутри останется прежним.

	Селектор может передаваться как без указания позиции (например, .landing-block-node-text), тогда будут изменены все карточки по данному селектору. Так и с указанием позиции (например, .landing-block-node-text@2), тогда будет изменена только карточка на указанной позиции (отсчет с нуля). Селектор можно передавать в виде `#wrapper`, тогда влияние будет происходить на стили блока (его оболочки).

Пример

В примере используется text-right - это **класс, который выравнивает справа**. Поэтому в affect задаётся что все нижележащие стили text-align должны быть удалены.

Важно! Такие классы как landing-block-node-text являются системными в манифесте. Если вы их не передадите, класс потеряется, и нода не сможет меняться через визуальный интерфейс. Вы должны четко понимать, что делаете.

```
BX24.callMethod(
    'landing.block.updateStyles',
    {
        lid: 311,
        block: 6058,
        data: {
           '.landing-block-node-text': {
                  classList: ['landing-
block-node-text', 'g-color-gray-light-v2',
'text-right'],
                  affect: ['text-align']
            }
```

```
            }
        },
    function(result)
    {
        if(result.error())
        {
            console.error(result.error());
        }
        else
        {
            console.info(result.data());
        }
    }
);
```

landing.block.uploadfile

```
landing.block.uploadfile(block, picture)
```

Метод загружает картинку и привязывает ее к указанному блоку. В случае успеха возвращает пару: прямой путь до загруженного файла и id сохраненного файла. С этого момента картинка удалится только при полном удалении блока, страницы, содержащей блок, или через вызов метода landing.landing.removeEntities.

Параметры

Параметр	Описание	С версии
block	ID блока	
picture	Варианты: 1. Путь до картинки, размещенной по веб-адресу. 2. `document.getElementById('file')` в случае работы через JS API. 3. Массив имя + содержимое `{ 0: 'name.jpg', 1: 'base64-содержимое файла' }`	

Пример

```
BX24.callMethod(
   'landing.block.uploadfile',
   {
      block: 12294,
      picture: 'https://site.com/******.jpg'
//      picture:
document.getElementById('file')
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

landing.block.getlist

```
landing.block.getlist(lid, params)
```

Метод для получение списка блоков страницы. Возвращает массив блоков или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы, или массив идентификаторов.	
params	Параметры: **edit_mode** - Режим редактирования (1) или нет (0 - по-умолчанию), вернется разный набор блоков. Обратите внимание, если вы еще не публиковали страницу, в случае режима 0 не вернется ничего. **delited** - удаленные (1) или нет (0) блоки, по-умолчанию выводятся все не удаленные. В режиме `edit_mode=0` удаленных блоков быть не может.	

Пример

```
BX24.callMethod(
        'landing.block.getlist',
        {
                lid: 313,
                params: {
                        edit_mode: 0
                }
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.block.getbyid

```
landing.block.getbyid(block, params)
```

Метод для получения блока по его идентификатору. Возвращает блок или ошибку.

Параметры

Метод	Описание	С версии
block	Идентификатор блока	
params	Параметры: **edit_mode** - Режим редактирования (1) или нет (0 - по-умолчанию), вернется разный набор блоков.	

Пример

```
BX24.callMethod(
        'landing.block.getbyid',
        {
                block: 6102,
```

```
                params: {
                        edit_mode: 0
                }
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.block.getmanifest

```
landing.block.getmanifest(lid, block,params)
```

Метод для получения манифеста конкретного блока, уже размещенного на странице. Возвращает манифест блока или ошибку.

Параметры

Метод	Описание	С версии
lid	Идентификатор страницы	
block	Идентификатор блока	
params	Параметры: **edit_mode** - режим редактирования (1) или нет (0 - по-умолчанию)	

Пример

```
BX24.callMethod(
        'landing.block.getmanifest',
```

```
        {
                lid: 313,
                block: 6102,
                params: {
                        edit_mode: 0
                }
        },
        function(result)
        {
                if(result.error())
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.block.getmanifestfile

```
landing.block.getmanifestfile(code)
```

Метод для получения манифеста блока из репозитория. Вернет манифест блока или ошибку.

Параметры

Метод	Описание	С версии
code	код блока	

Пример

```
BX24.callMethod(
        'landing.block.getmanifestfile',
        {
                code: '01.big_with_text'
        },
        function(result)
        {
                if(result.error())
```

```
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

landing.block.getrepository

```
landing.block.getrepository(section)
```

Метод возвращает список блоков из репозитория или ошибку.

Параметры

Метод	Описание	С версии
section	Код секции репозитория (список дан выше)	

Пример

```
BX24.callMethod(
        'landing.block.getrepository',
        {
                section: 'about'
        },
        function(result)
        {
                if(result.error())
```

```
                {

console.error(result.error());
                }
                else
                {

console.info(result.data());
                }
        }
);
```

Введение

Шаблоны представления описывают представления конкретного сайта или страницы о том, как выглядит их разметка:

- С шапкой и подвалом
- С сайдбаром
- Их вариации

Причем привязка к странице сильнее, чем привязка к сайту (если и сайт и страница данного сайта привязаны к разным шаблонам, то вывод будет происходить по сетке шаблона страницы). Каждая область такого шаблона есть отдельная страница (с набором блоков и прочими аналогичными характеристиками).

На данный момент в системе предустановлено несколько шаблонов представления и расширять их у стороннего разработчика нет возможности.

Вот данный список:

Внешний код (XML_ID)	Название
empty	Пустой
sidebar_left	С сайдбаром слева
sidebar_right	С сайдбаром справа
header_footer	С шапкой и подвалом
without_left	Без левого сайдбара

without_right	Без правого сайдбара

Обратите внимание, здесь не указываются идентификаторы шаблонов, вместе с тем при передаче сайту или странице нужно указать именно идентификатор. Чтобы получить непосредственно идентификатор, воспользуйтесь методом landing.template.getlist. Вы получите список идентификаторов в разрезе именно вашего портала.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Поля сущности Шаблон

Поле	Описание	Чтение	Запись
ID	Идентификатор шаблона.	Да	Нет
ACTIVE	Активность шаблона: Y / N.	Да	Нет
AREA_COUNT	Количество областей помимо контента	Да	Нет
SORT	Сортировка	Да	Нет
TITLE	Название.	Да	Нет
XML_ID	Внешний код.	Да	Нет
CONTENT	Разметка шаблона.	Да	Нет
CREATED_BY_ID	Идентификатор пользователя создавшего шаблон.	Да	Нет
MODIFIED_BY_ID	Идентификатор пользователя изменившего шаблон.	Да	Нет
DATE_CREATE	Дата создания.	Да	Нет

DATE_MODIFY	Дата изменения.	Да	Нет

Включаемые области шаблона

У каждого шаблона есть включаемые области (за исключением пустого). Как правило, это шапка, подвал, сайдбар левый или правый. Каждая такая включаемая область это независимая страница. Для работы с такими областями есть ряд методов, описанных ниже.

Метод	Описание
landing.template.getSiteRef	Метод получает список включаемых областей для сайта.
landing.template.getLandingRef	Метод получает список включаемых областей для страницы.
landing.template.setSiteRef	Метод устанавливает включаемые области для сайта.
landing.template.setLandingRef	Метод устанавливает включаемые области для страницы.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

landing.template.getLandingRe

```
landing.template.getLandingRef(id)
```

Метод получает список включаемых областей для страницы. Ключами возвращаемого массива являются идентификаторы включаемых областей, а значениями - идентификаторы страниц.

Параметры

Метод	Описание
ID	Идентификатор страницы

Пример

```
BX24.callMethod(
   'landing.template.getLandingRef',
   {
      id: 557
   },
   function(result)
   {
```

```
        if(result.error())
        {
           console.error(result.error());
        }
        else
        {
           console.info(result.data());
        }
    }
);
```

landing.template.getlist

```
landing.template.getlist(params)
```

Метод для получения списка шаблонов

Параметры

Параметр	Описание	С версии
params	Опциональный массив, с опциональными ключами: **select**, **filter**, **order**, **group**, которые содержат значения таблицы основных полей сущности.	

Пример

```
BX24.callMethod(
   'landing.template.getlist',
   {
      params: {
```

```
            select: [
               'ID', 'TITLE'
            ],
            filter: {
               '>ID': 0
            },
            order: {
               ID: 'DESC'
            }
         }
      },
      function(result)
      {
         if(result.error())
         {
            console.error(result.error());
         }
         else
         {
            console.info(result.data());
         }
      }
   );
```

landing.template.getSiteRef

```
landing.template.getSiteRef(id)
```

Метод получает список включаемых областей для сайта. Ключами возвращаемого массива являются идентификаторы включаемых областей, а значениями - идентификаторы страниц.

Параметры

Параметр	Описание
id	Идентификатор сайта.

Пример

```
BX24.callMethod(
   'landing.template.getSiteRef',
   {
      id: 1
   },
   function(result)
   {
```

```
        if(result.error())
        {
           console.error(result.error());
        }
        else
        {
           console.info(result.data());
        }
    }
);
```

landing.template.setLandingRe

```
landing.template.setLandingRef(id, data)
```

Метод устанавливает включаемые области для страницы в рамках конкретного шаблона (страница должна быть уже привязана к шаблону через поле TPL_ID). Вернет *true* в случае успеха, или ошибку.

Параметры

Параметр	Описание
id	Идентификатор страницы.
data	Массив данных для установки (если массив пустой или не передан, установленные области сбросятся). Ключами массива являются идентификаторы областей, а значениями идентификаторы страниц, которые необходимо установить как область.

Пример

```
BX24.callMethod(
   'landing.template.setLandingRef',
   {
      id: 557,
      data: {
         1: 614,
         2: 615,
         3: 616
      }
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

landing.template.setSiteRef

```
landing.template.setSiteRef(id, data)
```

Метод устанавливает включаемые области для сайта в рамках конкретного шаблона (сайт или страница должны быть уже привязаны к шаблону через поле TPL_ID). Вернет *true* в случае успеха, или ошибку.

Параметры

Метод	Описание
id	Идентификатор сайта.
data	Массив данных для установки (если массив пустой или не передан, установленные области сбросятся). Ключами массива являются идентификаторы областей, а значениями идентификаторы страниц, которые необходимо установить как область.

Пример

```
BX24.callMethod(
   'landing.template.setSiteRef',
   {
      id: 557,
      data: {
         1: 614,
         2: 615,
         3: 616
      }
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Настройки

Место встраивания LANDING_SETTINGS позволяет добавить новый пункт в меню настроек (Страницы / Сайта) в режиме редактирования страницы.

Параметры

Для данного места встраивания доступны параметры:

Параметр	Описание	С версии
SITE_ID	идентификатор сайта.	
LID	идентификатор страницы.	

Получить параметры можно из PLACEMENT_OPTIONS:

```
$placement =
isset($_REQUEST['PLACEMENT_OPTIONS'])
         ?
json_decode($_REQUEST['PLACEMENT_OPTIONS'],
true)
         : [];
```

Пример

```
BX24.callMethod(
   'landing.repo.bind',
   {
      fields: {
         PLACEMENT: 'LANDING_SETTINGS',
         PLACEMENT_HANDLER:
'https://cpe/rest/settings.php',
         TITLE: 'My settings'
      }
   },
   function(result)
   {
      if(result.error())
         console.error(result.error());
      else
         console.info(result.data());
   }
);
```

Редактирование блока

Описание

На данный момент разработчики имеют возможность внедриться в пункты редактирования любого блока. В случае такой регистрации рядом с кнопками блока "Редактировать" и "Дизайн" появится кнопка вызова вашего приложения.

Код для встройки приложения зависит от кода блока и в общем виде выглядит так: `LANDING_BLOCK_<CODE>`. Если в случае системного блока все понятно, и вместо `<CODE>` нужно вставить код блока (примеры ниже), то в случае встраивания в зарегистрированный вами же блок нужно подставить его идентификатор.

Например:

1. Регистрируем блок. Метод вернет ID блока. Пусть он равен 1132, для примера.
2. При регистрации места встраивания указываем код: `LANDING_BLOCK_repo_1132` (или `LANDING_BLOCK_REPO_1132`, регистр не важен)

Параметры

Для данного места встраивания доступны параметры:

Параметр	Описание	С версии
ID	– идентификатор блока.	

CODE	– символьный код блока.	
LID	– идентификатор страницы.	

Получить параметры можно из PLACEMENT_OPTIONS:

```
$placement =
isset($_REQUEST['PLACEMENT_OPTIONS'])
         ?
json_decode($_REQUEST['PLACEMENT_OPTIONS'],
true)
         : [];
```

Пример

```
BX24.callMethod(
   'landing.repo.bind',
   {
      fields: {
         PLACEMENT:
'LANDING_BLOCK_04.1.one_col_fix_with_title',
         PLACEMENT_HANDLER:
'https://cpe/rt/placement.php',
         TITLE: 'My block'
      }
   },
   function(result)
   {
      if(result.error())
         console.error(result.error());
      else
         console.info(result.data());
   }
);
```

Если вы хотите встроить универсальное приложение для каждого блока, следует указывать код со *:

```
BX24.callMethod(
   'landing.repo.bind',
   {
      fields: {
         PLACEMENT: 'LANDING_BLOCK_*',
         PLACEMENT_HANDLER:
'https://cpe/rt/placement.php',
         TITLE: 'My block'
      }
   },
   function(result)
   {
      if(result.error())
         console.error(result.error());
      else
         console.info(result.data());
   }
)
```

Обновление блока из приложения

В открывающемся приложении доступна команда на обновление конкретного блока. Подразумевается, что после работы с блоком, из которого вызвано приложение, вам может потребоваться его обновить. Делается это через команду refreshBlock.

Пример

```
BX24.placement.call(
   'refreshBlock',
   {
      id: 123//блок с идентификатором 123
   },
```

```
    function()
    {
       console.log('Блок успешно обновлен');
       //закрываем слайдер
    }
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Удаление с места встраивания

Удаление происходит собственным методом модуля **landing.repo.unbind**, которому просто передаётся код места встраивания. Удалятся все места встраивания по этому коду. Если приложением зарегистрировано несколько мест с различными путями, то удалить конкретное можно, передав адрес места встраивания.

Пример

```
BX24.callMethod(
   'landing.repo.unbind',
   {
      code: 'LANDING_SETTINGS',
//      handler:
'https://site.ru/rt/placement.php?version=3'
   },
   function(result)
   {
      if(result.error())
         console.error(result.error());
      else
         console.info(result.data());
   }
);
```

Встраивание Базы знаний

Описание

В интерфейсе Битрикс24 встречаются [dw]места встройки Баз знаний самим пользователем[/dw][di][/di]. Давайте разберем как с этим работать с позиции REST. Обратим внимание, что это не места встройки в чистом виде? которые рассмотрены здесь. Это исключительно работа с базами знаний с позиции интерфейса.

Для начала вам нужно определиться с меню, в которое вы хотите встроиться, получить его код. Сделать это можно, открыв в интерфейсе выбор привязки к меню ("Выбрать Базу знаний") и посмотрев адрес открывшегося фрейма. Там будет параметр, например, `menuId=crm_switcher:deal`. Это и есть так называемый код меню. И вот с ним уже можно работать.

Встраивание в меню и удаление

Встраивание в меню

Метод **landing.site.bindingToMenu** привязывает Базу знаний в указанное меню. К Базе знаний должен быть доступ на чтение.

Параметры

Параметр	Описание	С версии
id	Идентификатор Базы знаний.	

menuCode	Символьный код меню, как мы определяли выше.	

Пример

```
BX24.callMethod(
   'landing.site.bindingToMenu',
   {
      id: 31,
      menuCode: 'crm_switcher:deal'
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

Удаление из меню

Метод **landing.site.unbindingFromMenu** удаляет привязку Базы знаний в меню. К Базе знаний должен быть доступ на чтение.

Параметры

Параметр	Описание	С версии
id	Идентификатор Базы знаний.	

menuCode	Символьный код меню, как мы определяли выше.	

Пример

```
BX24.callMethod(
   'landing.site.unbindingFromMenu',
   {
      id: 31,
      menuCode: 'crm_switcher:deal'
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

Привязки в меню

Получение списка привязок в меню

Метод **landing.site.getMenuBindings** возвращает список привязанных к меню (конкретному или всем) Баз знаний. Вернутся только привязки, к Базам знаний которых текущий пользователь имеет доступ на чтение.

Параметры

Параметр	Описание	С версии
menuCode	Символьный код меню, как мы определяли выше. Не обязательный, по умолчанию возвращаются все привязки.	

Пример

```
BX24.callMethod(
   'landing.site.getMenuBindings',
   {
      menuCode: 'crm_switcher:deal'
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

Привязка к группе Социальной сети

Метод **landing.site.bindingToGroup** привязывает конкретную Базу знаний к группе. Пользователь должен состоять в указанной группе, и у группы не должно быть привязанной Базы знаний.

Параметры

Параметр	Описание	С версии
id	Идентификатор Базы знаний.	
groupId	Идентификатор группы.	

Пример

```
BX24.callMethod(
   'landing.site.bindingToGroup',
   {
      id: 32,
      groupId: 174
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

Удаление привязки к группе социальной сети

Метод **landing.site.unbindingFromGroup** отвязывает конкретную Базу знаний от группы. Пользователь должен состоять в указанной групп.

Параметры

Параметр	Описание	С версии
id	Идентификатор Базы знаний.	
groupId	Идентификатор группы.	

Пример

```
BX24.callMethod(
   'landing.site.unbindingFromGroup',
   {
      id: 32,
      groupId: 174
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

Получение привязок к группам

Метод **landing.site.getGroupBindings** позволяет узнать, существует ли привязка к группе, или какие вообще есть привязки к группам. Вернутся только привязки, к Базам знаний которых текущий пользователь имеет доступ на чтение.

Параметры

Параметр	Описание	С версии
groupId	Идентификатор группы, для которой надо вернуть привязку. Не обязательный, по умолчанию возвращаются все привязки к любым группам.	

Пример

```
BX24.callMethod(
   'landing.site.getGroupBindings',
   {
      groupId: 174
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
```

```
    }
);
```

landing.repo.checkContent

```
landing.repo.checkContent(
   content,
   splitter
)
```

Метод проверяет контент на опасные подстроки. К таким относятся `onclick=""`, `<iframe>` и ряд других. При обычном кейсе использования варианты срабатывания минимальны. Метод используется исключительно для контроля содержимого при регистрации блока.

Параметры

Параметр	Описание	С версии
content	Содержимое для тестирования.	
splitter	Необязательный параметр для разделения опасных подстрок. По-умолчанию равен `#SANITIZE#`.	

Пример

```
BX24.callMethod(
   'landing.repo.checkContent',
   {
      content: '<div style="color: red"
onclick="alert(123)"><iframe
src="//evil.com"></iframe></div>',
      splitter: '#AAA#'
   },
   function(result)
   {
      if(result.error())
         console.error(result.error());
      else
         console.info(result.data());
   }
);
```

В ответе вернется:

```
content:"
"
is_bad:true
```

Собственно, метка `is_bad = true`, говорящая о том, что в содержимом есть опасные места, и сам текст, помеченный разделителями в опасных местах. Разработчику надлежит изменить такие места перед регистрацией.

landing.repo.getList

Описание

```
landing.repo.getList(params)
```

Метод для получения списка блоков текущего приложения.

Параметры

Параметр	Описание	С версии
params	Опциональный массив, с опциональными ключами: ▪ select, ▪ filter, ▪ order, ▪ group, которые содержат значения таблицы основных полей сущности. Таблица размещена ниже.	

Поля сущности

Поле	Описание

ID	Идентификатор записи
XML_ID	Уникальный код записи.
APP_CODE	Код текущего приложения.
ACTIVE	Активность (Y / N).
NAME	Название.
DESCRIPTION	Описание.
SECTIONS	Символьные коды категорий.
PREVIEW	Превью изображение.
MANIFEST	Манифест.
CONTENT	Контент.
CREATED_BY_ID	Идентификатор пользователя создавшего запись.
MODIFIED_BY_ID	Идентификатор пользователя изменившего запись.
DATE_CREATE	Дата создания.
DATE_MODIFY	Дата изменения.

Пример

```
BX24.callMethod(
    'landing.repo.getList',
    {
        params: {
            select: [
```

```
                'ID', 'NAME', 'MANIFEST'
            ],
            filter: {
                '>ID': '1'
            }
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.info(result.data());
    }
);
```

landing.repo.register

Описание

```
landing.repo.register(code, fields,
manifest)
```

Метод добавления блока в репозиторий. Возвращает ошибку или ID добавленного блока. Этот ID используется для добавления блока на создаваемые программно лендинги.

При добавлении происходит проверка. Если блок с данным кодом уже присутствует в системе, произойдет его удаление

Метод может вернуть ошибку об опасном содержимом блока. В этом случае требуется сначала проверить регистрируемое содержимое методом landing.repo.checkContent.

При разработке нового блока или изменении существующего может потребоваться быстрее увидеть изменение, чем это позволяет пере-добавление блока или флаг RESET. Рекомендуется для этих целей использовать метод landing.block.updatecontent. Метод передаёт в блок произвольный контент и отображает изменения практически "налету". После того как разработка закончена, разработчик может окончательно его зарегистрировать.

Метод подходит только для изменения контента. При изменении манифеста блок требуется перерегистрировать (без пере-добавления на страницу).

Параметры

Метод	Описание	С версии
code	Уникальный код вашего блока, по нему будет осуществляться удаление блока, если требуется.	
fields	Массив полей, описывающих ваш блок, состоит из ключей: ▪ NAME - название блока ▪ DESCRIPTION - описание блока ▪ SECTIONS - категории, в которых должен появиться блок, через запятую. Если нужной категории нет в списке, просто напишите ее текстом в манифесте, категория будет добавлена. Ключом новой категории становится значение `md5(strtolower($sectionName))`. ▪ PREVIEW - URL картинки, обложки блока ▪ CONTENT - html-содержимое блока ▪ ACTIVE - активность блока (Y / N) ▪ SITE_TEMPLATE_ID – привязка блока к определенному шаблона сайта главного модуля. **Только для коробочных версий!** Дополнительные параметры: ▪ RESET - если передать со значением Y, то система автоматически обновит все добавленные на страницы блоки на новую верстку. Обратите внимание, технология тестовая и требует отдельного тестирования на блоках, которые вы хотите изменить. Подробнее...	

manifest	Массив манифеста, которым описывается блок.	

Атрибут **style** может вырезаться встроенным санитайзером. Чтобы это обойти используйте вместо него атрибут **bxstyle**. При добавлении система конвертирует его в штатный style.

Пример

```
<?
//для наглядности, передадим PHP-массив на
исполнение JS
$data = array(
   'code' => 'myblockx',
   'fields' => array(
      'NAME' => 'Test block',
      'DESCRIPTION' => 'Just try!',
      'SECTIONS' => 'cover,about',
      'PREVIEW' =>
'https://www.bitrix24.ru/images/b24_screen.p
ng',
      'CONTENT' => '
<section class="landing-block">
   <div class="text-center g-color-gray-
dark-v3 g-pa-10">
      <div class="g-width-600 mx-auto">
         <div class="landing-block-node-text
g-font-size-12 ">
            <p>© 2017 All right reserved.
Developed by
            <a href="#" class="landing-
block-node-link g-color-
primary">Bitrix24</a></p>
         </div>
      </div>
```

```
    </div>
</section>'
    ),
    'manifest' => array(
        'assets' => array(
            'css' => array(
                'https://site.com/aaa.css'
            ),
            'js' => array(
                'https://site.com/aaa.js'
            )
        ),
        'nodes' =>
            array(
                '.landing-block-node-text' =>
                    array(
                        'name' => 'Text',
                        'type' => 'text',
                    ),
                '.landing-block-node-link' =>
                    array(
                        'name' => 'Link',
                        'type' => 'link',
                    ),
            ),
        'style' =>
            array(
                '.landing-block-node-text' =>
                    array(
                        'name' => 'Text',
                        'type' => 'typo',
                    ),
                '.landing-block-node-link' =>
                    array(
                        'name' => 'Link',
                        'type' => 'typo',
                    ),
```

```
         ),
      'attrs' =>
         array(
           '.landing-block-node-text' =>
              array(
                 'name' => 'Настройка
копирайта',
                 'type' => 'dropdown',
                 'attribute' => 'data-copy',
                 'items' => array(
                    'val1' => 'Значение 1',
                    'val2' => 'Значение 2'
                     )
                ),
         ),
   ),
);
?>
// обратите внимание! далее идет JS код.
BX24.callMethod(
   'landing.repo.register',
   //абстрактный метод, превращающий PHP-
массив в JS-объект
   <?= \CUtil::PhpToJSObject($data)?>,
   function(result)
   {
      if(result.error())
         console.error(result.error());
      else
         console.info(result.data());
   }
);
```

landing.repo.unregister

```
landing.repo.unregister(code)
```

Метод удаления блока. Возвращает *true* при удалении или *false*, если блок уже удален или его не было.

Пример

Метод	Описание	С версии
code	Уникальный код удаляемого блока.	

Пример

```
BX24.callMethod(
    'landing.repo.unregister',
    {code: 'myblockx'},
    function(result)
    {
        if(result.error())
            console.error(result.error());
```

```
        else
            console.info(result.data());
    }
);
```

Введение по шаблонам

Партнерские шаблоны дают возможность добавить свой шаблон в мастер создания сайта или страницы. Для разработчика не составит труда создать свой сайт и распространять его среди клиентов. Вот как это можно сделать.

1. В начале конечно же нужно создать сайт и страницы (или сайт из одной страницы), настроить ссылки как внутри страниц, так и между страницами. Вы можете создать с нуля или на основании одного из готовых шаблонов в штатной поставке продукта.

   > **Обратите внимание!** Ряд шаблонов собран еще на старой версии API и их экспорт может сработать неадекватно. Мы уже работаем над исправлением ошибок, но потребуется время.

2. У себя на портале вызвать метод landing.site.fullExport, результат работы которого сохранить в некоем хранилище. Например, json-файл.
3. В своем приложении при его установке вызвать landing.demos.register, которому передать данный массив. Формат массива не описывается, его изменение остается на совести разработчика.
4. Если вам не хватает блоков, вы можете заказать верстку у специалистов, или воспользоваться предложенными дополнительными блоками вендора.

Важные моменты:

- Если сайт содержит одну страницу, то при установке (п.3) шаблон добавится как в мастер создания сайта, так и в мастер создания страницы. В ином случае будет добавлен только шаблон в мастер создания сайта.
- Если вы экспортировали магазин, то и шаблон появится только в магазинах. Если сайт, то только в сайтах.

Исключение составляют одностраничные сайты, которые также появляются среди шаблонов мастера создания страницы в рамках магазина.

- Если на созданной вами странице размещен партнерский блок вы должны позаботиться заранее, что при вызове **landing.demos.register** необходимые партнерские блоки уже будут зарегистрированы в системе.
- При удалении приложения, шаблоны и блоки приложения также удаляются, за исключением страниц, которые были созданы на основании шаблонов.

URL предварительного просмотра

Данная ссылка показывается в предварительном просмотре шаблона, когда пользователь его выбирает. Получить ссылку для регистрации шаблона можно следующим образом:

1. Создать сайт/страницу в облачном Битрикс24 по данному шаблону. Страница может лишь минимально отличаться от регистрируемого вами шаблона, чтобы у пользователя, создающего сайты и страницы, не возникало вопросов.
2. Если это сайт многостраничный, достаточно главной страницы.
3. Опубликовать сайт и указать полученную ссылку в поле "URL предпросмотра".

Обратите внимание! Следите за активностью вашего портала, чтобы он не был автоматически удален, что повлечет за собой удаление и ссылок предпросмотра. А это в свою очередь деактивацию ваших решений.

Изображения

Изображения сохраняются в шаблоне как абсолютные ссылки на ваш адрес (тот, с которого вы экспортировали шаблон). И будут ссылаться на ваш адрес пока пользователь не поменяет их на свои изображения.

Цветовая палитра

Если сайт многостраничный, крайне рекомендуется создавать все страницы сайта в одной из тем.

landing.demos.getList

Описание

```
landing.demos.getList(
   params
)
```

Метод для получения списка доступных партнерских шаблонов текущего приложения.

Параметры

Параметр	Описание	С версии
params	Опциональный массив, с опциональными ключами: ▪ select ▪ filter ▪ order ▪ group которые содержат значения таблицы основных полей сущности. Таблица размещена ниже.	

Поля сущности

Поле	Описание
ID	Идентификатор записи.
XML_ID	Уникальный код записи.
APP_CODE	Код текущего приложения.
ACTIVE	Активность (Y / N).
TITLE	Название.
DESCRIPTION	Описание.
PREVIEW_URL	URL предварительного просмотра.
TYPE	Тип создаваемого сайта (STORE, PAGE).
TPL_TYPE	Размещается в мастере создания сайта / магазина (S) или страницы (P).
MANIFEST	Манифест.
SHOW_IN_LIST	Показывать ли в списке шаблонов.
PREVIEW / PREVIEW2X / PREVIEW3X	Разноразмерные превью.
CREATED_BY_ID	Идентификатор пользователя создавшего запись
MODIFIED_BY_ID	Идентификатор пользователя изменившего запись.
DATE_CREATE	Дата создания.
DATE_MODIFY	Дата изменения.

Пример

```
BX24.callMethod(
   'landing.demos.getList',
   {
      params: {
         select: [
            'ID', 'TITLE', 'MANIFEST'
         ],
         filter: {
            '>ID': '1'
         }
      }
   },
   function(result)
   {
      if(result.error())
         console.error(result.error());
      else
         console.info(result.data());
   }
);
```

landing.demos.getPageList

Метод для получения списка доступных шаблонов для создания страниц. Как партнерских, так и системных.

Параметры

Параметр	Описание	С версии
type	Тип шаблона (page: обычные сайты, store: магазины).	

Пример

```
BX24.callMethod(
   'landing.demos.getPageList',
   {
      type: 'page'
   },
   function(result)
   {
      if(result.error())
         console.error(result.error());
      else
         console.info(result.data());
```

```
    }
);
```

landing.demos.getSiteList

Метод для получения списка доступных шаблонов для создания сайтов. Как партнерских, так и системных.

Параметры

Параметр	Описание	С версии
type	Тип шаблона (page: обычные сайты, store: магазины)	

Пример

```
BX24.callMethod(
   'landing.demos.getSiteList',
   {
      type: 'page'
   },
   function(result)
   {
      if(result.error())
         console.error(result.error());
      else
         console.info(result.data());
```

```
    }
);
```

landing.demos.register

Описание

```
landing.demos.register(
   data
)
```

Метод регистрирует шаблон в мастере создания сайта и страницы. Возвращает массив идентификаторов для созданных шаблонов. Выполнение метода прерывается, когда происходит ошибка в массиве элементов и возвращается описание ошибки.

Для распространения созданного сайта, достаточно получить экспорт в файл на портале источнике, и распространить его приложение, вызвав данный метод при установке.

Параметры

Параметр	Описание	С версии
data	Результат метода landing.site.fullExport как есть.	
params	Может содержать следующие ключи (**только для коробочных версий**):	

	▪ **site_template_id** - привязка блока к определенному шаблону сайта (главного модуля).	

Локализация

Для получения разъяснений по локализациям шаблона, пожалуйста, см. здесь. Когда требуется локализация, раскомментируйте ключи **lang** и **lang_original**. Принцип, использованный здесь, аналогичен локализации блоков.

Имейте ввиду, что локализация применима только для основных фраз: названий страниц, описаний. Не перегружайте данный массив ненужной информацией.

Пример

Заметьте, что в примере использован результат метода **landing.site.fullExport**.

```
BX24.callMethod(
   'landing.site.fullExport',
   {
      id: 326,
      params: {
         edit_mode: 'Y',
         code: 'myfirstsite',//symbolic code
of site
         name: 'Сайт автомастерской',//
наименование сайта (страницы)
         description: 'Сайт для вашего
автосервиса. Под капотом все самое
нужное.',//описание сайта
         preview_url:
'http://sample.landing.mycompany.ru/',//url
предварительного просмотра
         preview:
```

```
'http://site.ru/preview.jpg',//основная
превью-картинка для списка шаблонов (реком.
280x115)
         preview2x:
'http://site.ru/preview.jpg',//увеличенная
превью-картинка (рекомен. 560x230)
         preview3x:
'http://site.ru/preview.jpg',//ретина-размер
превью картинки (рекомен. 845x345)
      }
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         var data = result.data();
         console.info(data);
         BX24.callMethod(
            'landing.demos.register',
            {
               data: data,
               params: {
                  site_template_id: '',//
передать значение шаблона, если вы
регистрируете для своего шаблона (только
коробка!)
                  //локализационный массив и
оригинальный язык
                  /*lang: {
                     en: {
                        'Фраза 1':
'Translate en 1',
                        'Фраза 2':
```

```
'Translate en 2'
                    },
                    de: {
                       'Фраза 1':
'Translate de 1',
                       'Фраза 2':
'Translate de 2'
                    }
                 },
                 lang_original: 'ru'*/
              }
           },
           function(result)
           {
              if(result.error())
              {

console.error(result.error());
              }
              else
              {

console.info(result.data());
              }
           }
        );
     }
   }
);
```

landing.demos.unregister

```
landing.demos.unregister(
   code
)
```

Метод удаляет зарегистрированный партнерский шаблон. Возвращает *true* или ошибку. В случае, если шаблон уже был удален, или не найден, вернет *false*.

Обратите внимание! Удаляется как шаблон сайта с данным кодом, так и все шаблоны страниц с данным кодом. Созданные сайты и страниц по этим шаблонам остаются нетронутыми.

Параметры

Параметр	Описание	С версии
code	Символьный код шаблона.	

Пример

```
BX24.callMethod(
   'landing.demos.unregister',
   {
      code: 'myfirstsite'
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

Локализация шаблона

Если вы предполагаете распространение шаблонов не только в сегменте родного языка, то непосредственно контент страниц должен быть на универсальном языке, английском. Названия и описания страниц вы можете давать уже на оригинальном языке, но при регистрации шаблона необходимо передать локализационный массив. Подробнее смотрите здесь.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Основные положения

Права выдаваться могут только администратором портала, и если тариф проекта платный (в случае облака). На данный момент права можно выдавать только для сайтов, и они распространяются на все вложенные страницы. Например, право "Изменение настроек", выданное для сайта, распространяется на страницу настроек сайта и страницы настроек любых его страниц.

Обратите внимание. Если портал переходит из категории платных в бесплатные, функционал прав отключается на нем автоматически и все закрытые сущности становятся доступными для всех.

© «Битрикс», 2001-2008, «1С-

Отличия

Поддерживается две модели прав: ролевая и расширенная.

Ролевая модель подразумевает установку прав по ролям в публичном разделе Сайты24 и Магазины24.
Расширенная модель подразумевает установку прав внутри настроек каждого сайта.

Модели полностью разделены по логике, как на этапе установки прав, так и на этапе контроля вывода сущностей. Это означает, что вы безболезненно можете переключаться между моделями и возвращаться обратно, прежние настройки будут сохранены.

Обратите внимание! При обращении к методам для различных моделей не происходит определение текущего режима, это должен определять разработчик самостоятельно с помощью соответствующих методов.

© «Битрикс», 2001-2008, «1С-

Переключение моделей

Внимание! Функционал доступен только администратору.

Переключение между режимами осуществляется следующим методом:

```
BX24.callMethod(
   'landing.role.enable',
   {
      mode: 1// 1 - для включения ролевой
модели, 0 - для выключения (включения
расширенной)
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

Определить, какая модель в данный момент включена на проекте, можно с помощью следующего метода:

```
BX24.callMethod(
   'landing.role.isEnabled',
   {
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         if (result.data())
         {
            console.log('Ролевая модель');
         }
         else
         {
            console.log('Расширенная
модель');
         }
      }
   }
);
```

landing.site.getRights

Метод вернет права текущего пользователя. В случае несуществующего сайта или отсутствия прав на него вернется одинаковое состояние – пустой массив. В ином случае массив, состоящий из возможных значений:

- **denied** - запрещено всё,
- **read** – чтение (право автоматически ставится системой дополнительно при указании любого другого отличного от denied),
- **edit** – изменение (содержимого страниц),
- **sett** – изменение настроек,
- **public** – публикация,
- **delete** – удаление (в корзину, и восстановление из корзины).

Параметры

Метод	Описание	С версии
id	Идентификатор сайта.	

Пример

```
BX24.callMethod(
   'landing.site.getRights',
```

```
    {
       id: 645
    },
    function(result)
    {
       if(result.error())
       {
          console.error(result.error());
       }
       else
       {
          console.info(result.data());
       }
    }
);
```

landing.site.setRights

Описание и пример

Устанавливает права доступа для сайта. Вернёт *true* или ошибку. Метод доступен только администратору портала, а в облаке в том числе только платным тарифам.

Пример

```
BX24.callMethod(
   'landing.site.setRights',
   {
      id: 645,
      rights: {
         'U3': [
            'edit', 'delete'
         ],
         'U1': [
            'edit', 'sett'
         ]
      }
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
```

```
        else
        {
            console.info(result.data());
        }
    }
);
```

Параметры

Параметр	Описание	С версии
id	Идентификатор сайта.	
rights	Объект с правами, ключами которого являются уникальные идентификаторы (пользователя, отдела, группы, ...), а значениями допустимые операции: ▪ **denied** – доступ закрыт (при установке данного права даже в составе других, другие права сбрасываются для конкретного объекта) ▪ **read** – чтение (право автоматически ставится системой дополнительно при указании любого другого отличного от denied) ▪ **edit** – изменение (содержимого страниц) ▪ **sett** – изменение настроек ▪ **public** – публикация ▪ **delete** – удаление (в корзину, и восстановление из корзины)	

Права независимы и могут даваться точечно. Например, пользователь может обладать только правом публикации без возможности любого изменения.

В качестве ключей можно использовать значения:

- **SG<X>** - рабочая группа, например SG1 - рабочая группа с идентификатором 2;
- **U<X>** - пользователь, например U45 - пользователь с идентификатором 45;
- **DR<X>** - отдел, включая подразделы, например DR23 - раздел с идентификатором 23;
- **UA** - все авторизованные пользователи.
- **G<X>** - группа пользователей, например G2 - группа пользователей с идентификатором 2.

landing.role.getList

Важно! Функционал доступен только администратору.

Метод позволяет получить список ролей. Вернет массив идентификаторов и названий всех ролей.

Параметры

Метод без параметров.

Пример

```
BX24.callMethod(
   'landing.role.getList',
   {
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
```

```
    }
);
```

landing.role.getRights

Важно! Функционал доступен только администратору.

Метод позволяет получить список сайтов, права на которые установлены в рамках роли. Метод вернет массив (см. пример), где ключами будут идентификаторы сайта, а значениями массив доступных операций (нулевой ключ означает доступ по-умолчанию для роли):

- **denied** - запрещено всё,
- **read** – чтение (право автоматически ставится системой дополнительно при указании любого другого отличного от denied),
- **edit** – изменение (содержимого страниц),
- **sett** – изменение настроек,
- **public** – публикация,
- **delete** – удаление (в корзину, и восстановление из корзины).

Параметры

Параметр	Описание	С версии
id	Идентификатор роли.	

Пример

```
BX24.callMethod(
   'landing.role.getRights',
   {
      id: 11
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

landing.role.setAccessCodes

Важно! Функционал доступен только администратору.

Метод устанавливает для роли коды доступа, для которых будет действовать данная роль (и ее ограничения на сайты). На вход методу передается идентификатор роли и массив кодов доступа. Если массив будет пуст, коды для роли будут сброшены.

Права независимы и могут даваться точечно. Например, пользователь может обладать только правом публикации без возможности любого изменения.

В качестве ключей можно использовать значения:

- **SG<X>** - рабочая группа, например SG1 - рабочая группа с идентификатором 2;
- **U<X>** - пользователь, например U45 - пользователь с идентификатором 45;
- **DR<X>** - отдел, включая подразделы, например DR23 - раздел с идентификатором 23;
- **UA** - все авторизованные пользователи.
- **G<X>** - группа пользователей, например G2 - группа пользователей с идентификатором 2.

Параметры

Параметр	Описание	Версия
id	Идентификатор роли.	

codes	Массив кодов доступа.	

Примеры

```
BX24.callMethod(
   'landing.role.setAccessCodes',
   {
      id: 11,
      codes: [
         'SG3_A', 'G4'
      ]
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
   }
);
```

landing.role.setRights

Важно! Функционал доступен только администратору.

Метод устанавливает необходимые права в рамках роли для списков сайта. Все иные сайты, не указанные во входящем массиве считаются отвязанными от роли.

Ключами массива идут идентификаторы сайта, а значениями массив доступных операций (нулевой ключ означает доступ по-умолчанию для роли):

- **denied** - запрещено всё,
- **read** – чтение (право автоматически ставится системой дополнительно при указании любого другого отличного от denied),
- **edit** – изменение (содержимого страниц),
- **sett** – изменение настроек,
- **public** – публикация,
- **delete** – удаление (в корзину, и восстановление из корзины).

Параметры

Параметры	Описание	С версии
id	Идентификатор роли.	
rights	Массив сайтов для привязки прав. См. пример.	

additional	Опционально может быть передан массив с дополнительными правами, кому разрешено в рамках роли: ▪ **menu24** – показывать ли для данной роли пункт меню "Сайты" / "Магазины" в облачном Битрикс24 ▪ **create** – разрешать ли в рамках роли создавать сайты	

Пример

```
BX24.callMethod(
   'landing.role.setRights',
   {
      id: 11,
      rights: {
         '0': ['read'],
         '66': ['read','edit','sett']
      },
      additional: ['menu24', 'create']
   },
   function(result)
   {
      if(result.error())
      {
         console.error(result.error());
      }
      else
      {
         console.info(result.data());
      }
```

```
    }
);
```

messageservice.message.statu

messageservice.message.status.update - метод для управления статусами SMS для приложений.

Параметры

Параметр	Описание
CODE	Код провайдера. Обязательный.
MESSAGE_ID	Идентификатор сообщения. Обязательный.
STATUS	Статус сообщения. Обязательный. Допустимые статусы: ▪ **delivered** - доставлено ▪ **undelivered** - не доставлено ▪ **failed** - ошибка доставки

Пример

```
BX24.callMethod(
    'messageservice.message.status.update',
    {
        CODE: 'provider1',
```

```
        message_id: 1,
        status: 'delivered'
    },
    function(result)
    {
        if(result.error())
            alert("Ошибка: " +
result.error());
        else
            alert("Успешно: " +
result.data());
    }
);
```

messageservice.sender.add

messageservice.sender.add - регистрирует новый SMS-провайдер. Успешное выполнение метода возможно только в рамках созданного приложения.

Параметры

Параметр	Описание
CODE	Внутренний идентификатор провайдера. Допустимые символы a-z, A-Z, 0-9, точка, дефис и нижнее подчеркивание.
TYPE	Тип провайдера.
HANDLER	URL приложения, на который будут отправлены данные.
NAME	Название провайдера. Может быть строкой или ассоциативным массивом локализированных строк.
DESCRIPTION	Описание провайдера. Может быть строкой или ассоциативным массивом локализированных строк.

На HANDLER приходят данные:

- **module_id** - модуль-инициатор. **crm** - значит сообщение отправлено из карточки (в будущем могут быть и другие варианты), **bizproc** - отправлено из Бизнес Процессов или робота.

- **bindings** - параметр, актуальный только для module_id = crm. В нем содержится массив привязок сообщения к сущностям CRM (к чему привяжется дело).
- **workflow_id**, **document_id**, **document_type** - параметры, актуальные только для module_id = bizproc. Параметры приходят не всегда: если отправляем из карточки, то их не будет)
- **message_id** - уникальный идентификатор сообщения. По нему можно обращаться к messageservice.message.status.update.
- **message_to** - номер получателя сообщения
- **message_body** - текст сообщения

Пример

```
                var params = {
                        CODE: 'provider1',
                        TYPE: 'SMS',
                        HANDLER: 'http:///',
                        NAME: 'Провайдер
***.ru',
                        DESCRIPTION:
'Провайдер ***.ru'
                };

                BX24.callMethod(

'messageservice.sender.add',
                        params,
                        function(result)
                        {

if(result.error())

alert("Error: " + result.error());
                                else
```

```
alert("Успешно: " + result.data());
                        }
                );
```

Отправление из карточки CRM

```
Array
(
    [module_id] => crm
    [bindings] => Array
        (
            [0] => Array
                (
                    [OWNER_TYPE_ID] => 1
                    [OWNER_ID] => 98
                )

        )

    [properties] => Array
        (
            [phone_number] => +7*********
            [message_text] => test message
        )

    [type] => SMS
    [code] => example
    [message_id] =>
72dd742c8270db0ddbbab92f98877537
    [message_to] => +7**********
    [message_body] => test message
    [ts] => 1506687055
    [auth] => /*auth*/

)
```

Отправление из Бизнес-процесса или робота.

```
Array
(
    [module_id] => bizproc
    [workflow_id] => 59ce38567ff2a5.26351167
    [document_id] => Array
        (
            [0] => crm
            [1] => CCrmDocumentLead
            [2] => LEAD_98
        )

    [document_type] => Array
        (
            [0] => crm
            [1] => CCrmDocumentLead
            [2] => LEAD
        )

    [properties] => Array
        (
            [phone_number] => +7*********
            [message_text] => test message
        )

    [type] => SMS
    [code] => example
    [message_id] =>
8b3fc6cd0cb4a7b91f6632889cdf46e0
    [message_to] => +7*********
    [message_body] => test message
    [ts] => 1506687103
    [auth] => /*auth*/
```

```
)
```

messageservice.sender.delete

messageservice.sender.delete - метод удаляет зарегистрированного SMS-провайдера.

Параметры

Параметр	Описание
CODE	Идентификатор провайдера.

Пример

```
        function uninstallProvider(provider)
        {
                BX24.callMethod(

'messageservice.sender.delete',
                        {
                                'CODE':
provider
                        },
                        function(result)
                        {
```

```
if(result.error())

alert('Error: ' + result.error());
                                else

alert("Успешно: " + result.data());
                        }
                );

        }
```

messageservice.sender.list

messageservice.sender.list - возвращает список зарегистрированных приложением SMS-провайдеров.

Пример

```
BX24.callMethod(
'messageservice.sender.list',
{},
function(result)
{
if(result.error())
alert("Error: " + result.error());
else
alert("Успешно: " + result.data().join(', '));
}
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Справочник кодов и типов

Таблица кодов вызова (CALL_FAILED_CODE)

Код	Описание
$MESS["VI_STATUS_200"]	Успешный звонок.
$MESS["VI_STATUS_304"]	Пропущенный звонок.
$MESS["VI_STATUS_603"]	Отклонено.
$MESS["VI_STATUS_603-S"]	Вызов отменен.
$MESS["VI_STATUS_403"]	Запрещено.
$MESS["VI_STATUS_404"]	Неверный номер.
$MESS["VI_STATUS_486"]	Занято.
$MESS["VI_STATUS_484"]	Данное направление не доступно.
$MESS["VI_STATUS_503"]	Данное направление не доступно.
$MESS["VI_STATUS_480"]	Временно не доступен.
$MESS["VI_STATUS_402"]	Недостаточно средств на счету.
$MESS["VI_STATUS_423"]	Заблокировано
$MESS["VI_STATUS_OTHER"]	Не определен.

Таблица типов звонка (CALL_TYPE)

Тип	Описание
1	Исходящий.
2	Входящий.
3	Входящий с перенаправлением (на мобильный или стационарный телефон).
4	Обратный звонок.

Таблица типов АТС

Тип	Описание
cloud	Облачная АТС.
office	Офисная АТС.

Таблица состояний SIP-регистрации

Состояние	Описание
success	Успешная регистрация.
error	Неудачная регистрация.
in_progress	В процессе регистрации.
wait	Ожидает начала регистрации.

Смотрите также

- Полный перечень допустимых статусов

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

telephony.call.attachTranscript

Описание

Метод добавляет расшифровку записи к звонку.

Параметры

Параметр	Описание	Тип
CALL_ID	Идентификатор звонка	string
COST	Стоимость расшифровки	float
COST_CURRENCY	валюта стоимости расшифровки	string
MESSAGES	расшифровка звонка. Массив объектов типа **TranscriptMessage**.	

Поля класса TranscriptMessage

Поля	Описание	Тип
SIDE	Участник разговора. Варианты значения: **User** - пользователь портала, **Client** - внешний участник.	string
START_TIME	Время начала фразы в секундах, считая от начала разговора.	int

STOP_TIME	Время окончания фразы в секундах, считая от начала разговора.	int
MESSAGE	Текст фразы	string

Пример

```
var callId = '';
var messages = [
    {
        SIDE: "User",
        START_TIME: 1,
        STOP_TIME: 3,
        MESSAGE: "Добрый день, чем могу
помочь"
    },
    {
        SIDE: "Client",
        START_TIME: 4,
        STOP_TIME: 8,
        MESSAGE: "Здравствуйте, вы продаете
пылесосы?"
    },
    {
        SIDE: "User",
        START_TIME: 9,
        STOP_TIME: 11,
        MESSAGE: "К сожалению, нет"
    },
    {
        SIDE: "Client",
        START_TIME: 11,
        STOP_TIME: 13,
        MESSAGE: "Понятно, до свидания"
    },
```

```
    {
        SIDE: "User",
        START_TIME: 13,
        STOP_TIME: 15,
        MESSAGE: "До свидания"
    },
];

BX24.callMethod(
    "telephony.call.attachTranscription",
    {
         CALL_ID: callId,
         MESSAGES: messages
    },
    function(response)
    {
        console.log(response.data())
    }
);
```

telephony.externalCall.attachR

Метод прикрепляет запись к завершенному звонку и к Делу звонка. (Должен вызываться после telephony.external.finish, если запись на момент вызова finish еще не готова.) Запись прикрепляется только одна. Если вызвать метод несколько раз, то следующий вызов перетрет предыдущую запись.

Параметры функции

Параметр	Описание	Тип значения
CALL_ID	Идентификатор звонка из метода telephony.externalcall.register. Обязательный.	string
FILENAME	Имя файла, обязательный. Имя файла должно заканчиваться на wav или mp3.	string
FILE_CONTENT	base64-кодированное содержимое файла. Необязательный. Если параметр не указать, метод вернет параметр `uploadUrl` - урл, на который можно за'upload'ить содержимое файла.	string
RECORD_URL	Ссылка на запись на сервере клиента. Если указано, то	string

	будет осуществлена попытка скачать запись по указанному адресу, вместо ожидания загрузки записи на портал клиента. Во время выполнения метода, портал осуществит одну попытку скачать запись по указанному адресу. В случае неудачи, метод вернет ошибку. Так как возможность скачивания зависит от множества независящих от портала факторов, использование данного параметра не рекомендуется.	

telephony.externalcall.finish

Метод завершает звонок, фиксирует его в статистике, скрывает у пользователя карточку звонка.

Возвращаемое значение

Метод возвращает массив, аналогичный методу voximplant.statistic.get.

Параметры функции

Параметр	Описание	Тип значен
CALL_ID	Идентификатор звонка из метода telephony.externalcall.register. Обязательный.	string
USER_ID	Идентификатор пользователя. Обязательный. Ответственный в лиде будет изменен на переданный USER_ID. Ответственный будет меняться только для лида, созданного автоматически при вызове метода telephony.externalcall.register. Для существующего лида ответственный не меняется.	int
DURATION	Длительность в сек. Обязательный.	int

COST	Стоимость звонка.	double
COST_CURRENCY	Валюта, в которой указана стоимость звонка. Список валют можно получить методом crm.currency.list.	string
STATUS_CODE	Код вызова (см. таблицу кодов вызова).	string
FAILED_REASON	Причина несостоявшегося звонка.	string
RECORD_URL	URL файла (желательно mp3) с записью звонка. Портал осуществит две попытки скачать запись по указанному адресу. В случае неудачи, запись приложена не будет и никакого уведомления об ошибке отправлено не будет. Данный параметр является устаревшим и оставлен исключительно для обратной совместимости. Использовать его крайне не рекомендуется. Для загрузки записи звонка используйте метод telephony.externalCall.attachRecord.	string
VOTE	Оценка звонка пользователем (если АТС поддерживает функционал оценки разговора).	int
ADD_TO_CHAT	Добавлять ли сообщение о совершенном звонке сотруднику Б24 в чат (Возможные значения 0 или 1, по умолчанию 1).	int

Вместо USER_ID также может принимать USER_PHONE_INNER.

telephony.externalcall.hide

Метод скрывает карточку звонка у пользователя.

Параметры функции

Параметр	Описание	Тип значения
CALL_ID	Идентификатор звонка из метода telephony.externalcall.register. Обязательное.	string
USER_ID	Идентификатор, либо массив идентификаторов пользователей, у которых надо скрыть звонок, если карточка показывается не у одного пользователя. Обязательный.	int

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

telephony.externalcall.register

Метод регистрирует звонок в Битрикс24, для чего ищет в CRM соответствующий номеру объект. Если находит, то добавляет звонок в привязке к найденному объекту. Если не находит, то может автоматически создать лид.

При использовании *telephony.externalCall.register* ответственным за новый лид будет автоматически назначен первый ответственный за данного клиента ранее. Сменить такого ответственного можно в дальнейшем через telephony.externalcall.finish.

Одновременно с регистрацией звонка метод опционально может показать пользователю карточку звонка. Пользователь, которому показывается карточка, идентифицируется либо по USER_ID, либо по USER_PHONE_INNER. (То есть, поля помечены как обязательные, но фактически, нужно только одно из двух.)

Не нужно повторно вызывать этот метод для звонков, полученных на событии OnExternalCallStart. Эти звонки уже зарегистрированы в системе и для них надо вызывать только *telephony.externalcall.finish* в конце звонка.

> **Внимание!** Повторный вызов *telephony.externalcall.register* с теми же параметрами, без закрытия предыдущего звонка методом *telephony.externalcall.finish*, выдает тот же CALL_ID в течение 30 минут.

Для создания дела "звонок" необходимо также вызывать метод telephony.externalcall.finish/

Возвращаемое значение

Метод возвращает массив:

Параметр	Описание	Тип значения
CALL_ID	Идентификатор звонка внутри Битрикс24.	string
CRM_CREATED_LEAD	Идентификатор созданного лида (создается, если в CRM не найден объект по входящему номеру)	int
CRM_ENTITY_ID	Идентификатор найденного в CRM объекта.	int
CRM_ENTITY_TYPE	Тип найденного в CRM объекта по входящему номеру CONTACT | COMPANY | LEAD.	string
CRM_CREATED_ENTITIES	Массив автоматически созданных в CRM сущностей при регистрации звонка. Формат: ▪ ENTITY_TYPE - тип созданной сущности ▪ ENTITY_ID - идентификатор созданной сущности	array
LEAD_CREATION_ERROR	Текст ошибки, возникшей при попытке создания лида в CRM.	string

Параметры функции

Параметр	Описание	Тип значения
USER_PHONE_INNER	Внутренний номер пользователя. Обязательный.	string
USER_ID	Идентификатор пользователя. Обязательный.	int
PHONE_NUMBER	Номер телефона. Обязательный	string
CALL_START_DATE	Дата/время звонка в формате iso8601. Обратите внимание, что в дате необходимо передавать часовой пояс, для избежания искажения времени звонка. Пример: `2021-02-03T18:25:10+03:00` .	string
CRM_CREATE	[0/1] - Автоматическое создание в CRM сущности, связанной со звонком. При необходимости, создает в CRM лид или сделку, в зависимости от [dw]настроек и режима работы CRM[/dw] [di]Существует: - простой режим (без лидов) - при котором будет создаваться сделка, а не лид; - режим повторных продаж, при котором будет создавать сделка/лид даже если сущность в crm найдена. (Но не будет	int

	создаваться если есть активная сделка/лид или номер внесен в черный список crm).[/di]. Обратите внимание, что дело звонка создается при любом значении этого параметра, если создание возможно.	
CRM_SOURCE	STATUS_ID источника из справочника источников. Получить список источников можно методом crm.status.list с фильтром по `"ENTITY_ID": "SOURCE"`.	string
CRM_ENTITY_TYPE	Тип объекта CRM, из карточки которого совершается звонок - CONTACT | COMPANY | LEAD.	string
CRM_ENTITY_ID	Идентификатор объекта CRM, тип которого указан в CRM_ENTITY_TYPE	int
SHOW	[0/1] Показывать ли карточку звонка (по умолчанию 1).	int
CALL_LIST_ID	Идентификатор списка обзвона, к которому должен быть привязан звонок.	int
LINE_NUMBER	Номер внешней линии, через который совершался звонок (см. telephony.externalLine.add). Необязательный. **Важно**! Значения из этого параметра используются в	string

	сценариях сквозной аналитики Битрикс24. Поэтому решения по интеграции с телефонией для каталога Приложения24 в обязательном порядке должны передавать здесь номер, на который был совершён регистрируемый входящий звонок.	
TYPE	Обязательный. Тип звонка: 1 - исходящий 2 - входящий 3 - входящий с перенаправлением 4 - обратный	integer

telephony.externalCall.searchC

Метод позволяет получить одним запросом по номеру телефона информацию о клиенте из CRM. Информация позволяет принять решение кому из сотрудников перенаправить входящий звонок на пользователя прямо в этот момент. Метод возвращает подходящий список CRM-объектов с сортировкой по внутренним приоритетам. Если за сущности связанные с номером отвечают разные сотрудники (за лид отвечает один сотрудник, за компанию - другой), то рекомендуется брать тот объект, который метод вернул первым в списке. Если интеграция предполагает собственную логику, то возможен выбор, так как отдаются все объекты.

В списке объектов CRM сразу сообщается вся информация об ответственном сотруднике за каждый объект (чтобы не приходилось получать эти данных дополнительными запросами REST). Возвращаются все заданные у пользователя контактные телефоны: внутренний телефон сотрудника, мобильный, рабочий и т.д..

Возвращается и статус рабочего дня сотрудника (если в Битрикс24 включен учет рабочего времени). Интеграция может проверить, находится ли сотрудник на рабочем месте (или у него перерыв), и либо перенаправить входящий звонок в очередь, либо направить звонок на мобильник сотрудника и т.д.

Рекомендуется вызывать метод до вызова telephony.externalcall.register.

Параметры

Параметр	Описание	С версии

PHONE_NUMBER	Номер клиента. Обязательный.	

Пример

Пример возвращаемых данных

```
Array
(
    [0] => Array
       (
           [CRM_ENTITY_TYPE] => CONTACT
           [CRM_ENTITY_ID] => 1
           [ASSIGNED_BY_ID] => 1
           [ASSIGNED_BY] => Array
               (
                   [ID] => 1
                   [TIMEMAN_STATUS] =>
CLOSED
                   [USER_PHONE_INNER] => 102
                   [WORK_PHONE] =>
                   [PERSONAL_PHONE] =>
                   [PERSONAL_MOBILE] =>
79062195047
               )

       )

   [1] => Array
       (
           [CRM_ENTITY_TYPE] => COMPANY
           [CRM_ENTITY_ID] => 4
           [ASSIGNED_BY_ID] => 1
           [ASSIGNED_BY] => Array
               (
                   [ID] => 1
```

```
                        [TIMEMAN_STATUS] =>
CLOSED
                        [USER_PHONE_INNER] => 102
                        [WORK_PHONE] =>
                        [PERSONAL_PHONE] =>
                        [PERSONAL_MOBILE] =>
79062195047
                )

        )

)
```

telephony.externalcall.show

Метод показывает карточку звонка пользователю.

Параметры функции

Параметр	Описание
CALL_ID	Идентификатор звонка из метода telephony.externalcall.register. Обязательный.
USER_ID	Идентификатор, либо [dw]массив идентификаторов пользователей[/dw][di]В стандартном для PHP формате: `USER_ID[0]=11&USER_ID[1]=24&USER_ID[2]=31`[/di].

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

telephony.externalLine.add

Метод добавляет внешнюю линию

Параметры

Метод	Описание	С версии
NUMBER	Номер внешней линии (Например: 79117654321)	string
NAME	Название внешней линии. Необязательный.	string

После добавления, внешнюю линию можно указать при регистрации звонка, в поле LINE_NUMBER (см telephony.externalcall.register).

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

telephony.externalLine.delete

Метод для удаления внешней линии.

Параметры

Параметр	Описание	Тип
NUMBER	Номер внешней линии	string

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

telephony.externalLine.get

Метод позволяет получить список внешних линий приложения.

Параметры

Без параметров.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

telephony.externalLine.update

Метод позволяет изменить название внешней линии

Параметры

Метод	Описание	С версии
NUMBER	Номер внешней линии	string
NAME	Название внешней линии	string

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Основные пользовательские сценарии и пример

1. Входящий звонок на АТС на внутренний номер конкретного пользователя должен показывать карточку звонка у сотрудника в Битрикс24 (Метод telephony.externalcall.register).
2. Входящий звонок с неизвестного номера (не зарегистрированного в CRM), должен попадать в очередь обработки (список пользователей, которые должны отвечать на входящие звонки):

    - Одновременная очередь: всем сотрудникам, которые не отвечают на другие звонки в данный момент, одновременно показывается карточка звонка, когда кто-то из них начинает отвечать на звонок, у остальных карточка пропадает (сначала **telephony.externalcall.register** для первого в очереди, затем telephony.externalcall.show для остальных).
    - Последовательная очередь: каждому из сотрудников очереди, которые не отвечают на другие звонки в данный момент, показывается карточка звонка на какое-то время (3-5-7 секунд), если сотрудник не начинает отвечать на звонок, карточка у него пропадает, и звонок переводится на следующего в очереди (сначала **telephony.externalcall.register** для первого в очереди, затем telephony.externalcall.hide и **telephony.externalcall.show** для следующего).

3. Входящий звонок с известного номера в виде карточки звонка отображается в Битрикс24 у менеджера, ответственного за соответствующий объект CRM. (Сначала **telephony.externalcall.register** с SHOW = 0, который вернет либо CREATED_LEAD в случае, если телефон не был найден в CRM и был создан новый лид, либо пару CRM_ENTITY_TYPE и

CRM_ENTITY_ID с указанием найденного существующего клиента.
Одновременно возвращается CRM_ACTIVITY_ID с идентификатором нового дела в CRM, в котором будет зафиксирован звонок в дальнейшем. Зная идентификатор объекта в CRM, можно при помощи методов REST по работе с CRM получить идентификатор менеджера, который отвечает за клиента, перевести звонок на него и показать ему карточку звонка **telephony.externalcall.show**)

4. Сотрудник в Битрикс24 нажимает на номер телефона в интерфейсе CRM. Приложение инициирует исходящий звонок на указанный номер на стороне АТС (событие onexternalcallstart, сотруднику показывается карточка звонка **telephony.externalcall.register**).

5. Звонок завершен (входящий или исходящий). Факт звонка и запись фиксируется в привязке к объекту CRM (telephony.externalcall.finish. Если на момент завершения звонка в АТС еще не готова запись разговора, то вместо **telephony.externalcall.finish** сначала просто скрываем карточку звонка **telephony.externalcall.hide**, а уже потом, когда запись готова, все-таки вызываем **telephony.externalcall.finish**).

6. На стороне АТС произошел входящий звонок в тот момент, когда связи между АТС и Битрикс24 по каким-то причинам нет. После восстановления связи информация о произошедшем звонке фиксируется в Битрикс24 (см. 1-3, но без показа карточки звонка – последовательный вызов методов **telephony.externalcall.register** с параметром SHOW = 0 и **telephony.externalcall.finish**.)

Примечание. Чтобы запись добавлялась к звонку при сценарии обзвона, приложения должны передвать CALL_LIST_ID который им придет в событии начала звонка.

Пример:

```
<?php
/**
 * Created by PhpStorm.
 * User: sv
 * Date: 01.11.16
 * Time: 10:44
 */

// ini_set('display_errors','Off');

// формируем url нашего скрипта для
использования в ajax-запросах из интерфейса
приложения
$script_url = ($_SERVER['SERVER_PORT'] ==
443 ? 'https' : 'http') . '://' .
$_SERVER['SERVER_NAME'] .
(in_array($_SERVER['SERVER_PORT'],
        array(80, 443)) ? '' : ':' .
$_SERVER['SERVER_PORT']) .
$_SERVER['SCRIPT_NAME'];

$appsConfig = array();
$b24domain = $_REQUEST['DOMAIN'];

// если нам пришло событие исходящего
звонка, то авторизация передается через узел
auth в массиве request
// но нам оттуда нужен только домен,
авторизацию мы уже сохранили к этому моменту
if (!empty($_REQUEST['auth'])) {
    $b24domain = $_REQUEST['auth']
['domain'];
}

$configFileName = '/config_' .
trim(str_replace('.', '_', $b24domain)) .
'.php';
```

```
echo getcwd().$configFileName."<br/>";
if (file_exists(getcwd() . $configFileName))
{
    include_once getcwd() . $configFileName;
} else {

    // сохраняем токены пользователя,
устанавливающего приложение
    $appsConfig = $_REQUEST;
    saveParams($appsConfig);

    // регистрируем событие исходящего
звонка
    restCommand('event.bind', array(
            "event" =>
'ONEXTERNALCALLSTART',
            "handler" => $script_url."?
action=outcoming",
        ),
        $b24domain, $appsConfig['AUTH_ID']);

    /* тестовое событие для проверки
механизма
    restCommand('event.bind', array(
        "event" => 'ONAPPTEST',
        "handler" => $script_url."?
action=test",
        ),
        $b24domain, $appsConfig['AUTH_ID']);
    */

}

$action = $_REQUEST['action'];

// мы просто запустили приложение в
интерфейсе Битрикс24
```

```
if ($action == '') {
?>
<html>
<head>
    <meta charset="utf-8"><meta http-
equiv="X-UA-Compatible" content="IE=edge">
<script
src="https://ajax.googleapis.com/ajax/libs/j
query/3.1.1/jquery.min.js"></script>

<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootst
rap/3.3.7/css/bootstrap.min.css"
integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4V
a+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">

<!-- Optional theme -->
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootst
rap/3.3.7/css/bootstrap-theme.min.css"
integrity="sha384-
rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRY
ooPp2bWYgmgJQIXwl/Sp"
crossorigin="anonymous">

<!-- Latest compiled and minified JavaScript
-->
<script
src="https://maxcdn.bootstrapcdn.com/bootstr
ap/3.3.7/js/bootstrap.min.js"
integrity="sha384-
Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWN
IpG9mGCD8wGNIcPD7Txa"
crossorigin="anonymous"></script>
```

```
    <script
src="//api.bitrix24.com/api/v1/"></script>
</head>
<body>

<div class="form-group">
    <label for="IncomingNumber">Incoming
call number</label>
    <input type="text" class="form-control"
id="incomingNumber" placeholder="phone">
</div>
<div class="form-group">
    <label for="user1">User 1</label>
    <input type="text" class="form-control"
id="user1" placeholder="user id" value="1">
</div>
<div class="form-group">
    <label for="user2">User 2 (for call
transfer)</label>
    <input type="text" class="form-control"
id="user2" placeholder="user id">
</div>
<a class="btn btn-default" href="#"
role="button" id="incoming">Incoming</a>
<a class="btn btn-default" href="#"
role="button" id="redirect">Redirect</a>
<a class="btn btn-default" href="#"
role="button" id="drop">Drop</a>
<a class="btn btn-default" href="#"
role="button" id="test">Event test</a>
<div id="debug"></div>

<?
    // если любопытно, можно посмотреть,
какие параметры авторизации передает
Битрикс24 скрипту приложения
```

```
    // в случае выполнения приложения во
фрейме внутри Битрикс24
    //echo "<pre>";
        //print_r($_REQUEST);
        //echo "</pre>";

?>
<script>
    $( "#incoming" ).on( "click", function(
event ) {
        // здесь мы имитируем работу внешней
АТС, в частности, получение входящего звонка
        // поэтому AJAX, передача параметров
авторизации и т.д.
        // в реальной практике, REST
телефонии будет вызываться со стороны АТС, а
там мы уже сохранили
        // авторизационные токены (см.
$appsConfig$appsConfig) и сами знаем, на
какой Битрикс24 отправлять
        // вызов REST, каким пользователям
показывать карточку и т.д.
        auth = BX24.getAuth();
        $.ajax({
            url: "<?=$script_url?>",
            data: {
                action: 'incoming',
                user1: $( "#user1" ).val(),
                phone: $( "#incomingNumber"
).val(),
                DOMAIN: auth['domain']
            },
            success: function( result ) {
                $( "#debug" ).html( result
);
            }
        });
```

```
    });

    $( "#redirect" ).on( "click", function(
event ) {
        auth = BX24.getAuth();
        $.ajax({
            url: "<?=$script_url?>",
            data: {
                action: 'redirect',
                user1: $( "#user1" ).val(),
                user2: $( "#user2" ).val(),
                DOMAIN: auth['domain']
            },
            success: function( result ) {
                $( "#debug" ).html( result
);
            }
        });
    });

    $( "#drop" ).on( "click", function(
event ) {
        auth = BX24.getAuth();
        $.ajax({
            url: "<?=$script_url?>",
            data: {
                action: 'drop',
                user1: $( "#user1" ).val(),
                user2: $( "#user2" ).val(),
                DOMAIN: auth['domain']
            },
            success: function( result ) {
                $( "#debug" ).html( result
);
            }
        });
    });
```

```
    /* инициация тестового события на
стороне серверного скрипта, ничего важного
    $( "#test" ).on( "click", function(
event ) {
        auth = BX24.getAuth();
        $.ajax({
            url: "<?=$script_url?>",
            data: {
                action: 'eventtest',
                DOMAIN: auth['domain']
            },
            success: function( result ) {
                $( "#debug" ).html( result
);
            }
        });
    });
    */
</script>
</body>
</html>
<? } else {

    switch ($action) {
        case 'test': writeToLog(array('test'
=> $_REQUEST), 'telephony test event');
                break;

        case 'outcoming':

            writeToLog(array('outcoming' =>
$_REQUEST), 'telephony event');

            $result =
restCommand('telephony.externalCall.register
',
                array(
```

```
                    "USER_ID" =>
$_REQUEST['data']['USER_ID'],
                    "PHONE_NUMBER"   =>
$_REQUEST['data']['PHONE_NUMBER'],
                    "TYPE" => '1',
                    "CRM_CREATE" => 1
                ),
                $b24domain,
$appsConfig['AUTH_ID']);

            $appsConfig['CALL'] =
$result['result'];


            saveParams($appsConfig);

            break;
        case 'eventtest':

            writeToLog(array('eventtest' =>
$_REQUEST), 'test event call');
            $result =
restCommand('event.test',
                array(
                ),
                $b24domain,
$appsConfig['AUTH_ID']);

            echo "test event call";

            break;
        case 'incoming':

            $result =
restCommand('telephony.externalCall.register
',
                array(
```

```
                    "USER_ID" =>
$_REQUEST['user1'],
                    "PHONE_NUMBER"   =>
$_REQUEST['phone'],
                    "TYPE" => '2',
                    "CRM_CREATE" => true
                ),
                $b24domain,
$appsConfig['AUTH_ID']);

            $appsConfig['CALL'] =
$result['result'];

            saveParams($appsConfig);

            echo "incoming <pre>";
            print_r($appsConfig);
            echo "</pre>";

            break;
        case 'redirect':

            echo "redirect <pre>";
            print_r($appsConfig);
            echo "</pre>";

            if ($appsConfig['CALL']
['CALL_ID'] != '') {

                $result =
restCommand('telephony.externalCall.hide',
                    array(
                        "CALL_ID" =>
$appsConfig['CALL']['CALL_ID'],
                        "USER_ID" =>
$_REQUEST['user1']
                    ),
```

```
                    $b24domain,
$appsConfig['AUTH_ID']);

                $result =
restCommand('telephony.externalCall.show',
                    array(
                        "CALL_ID" =>
$appsConfig['CALL']['CALL_ID'],
                        "USER_ID" =>
$_REQUEST['user2']
                    ),
                    $b24domain,
$appsConfig['AUTH_ID']);
            }
            echo "redirected to
".$_REQUEST['user2'];
            break;
        case 'drop':
            writeToLog(array('config' =>
$appsConfig), 'call is finishing');

            if ($appsConfig['CALL']
['CALL_ID'] != '') {

                $result =
restCommand('telephony.externalCall.finish',
                    array(
                        "CALL_ID" =>
$appsConfig['CALL']['CALL_ID'],
                        "USER_ID" =>
$_REQUEST['user1'],
                        "DURATION"   =>
'120',
                        "STATUS_CODE" =>
'200',
                        "ADD_TO_CHAT" =>
true
```

```
                    ),
                    $b24domain,
$appsConfig['AUTH_ID']);

                $appsConfig['CALL'] =
$result['result'];

                saveParams($appsConfig);

                echo "finished <pre>";
                print_r($appsConfig);
                echo "</pre>";

                writeToLog(array('request'
=> $_REQUEST, 'config' => $appsConfig),
'call is finished');
            }
            echo "dropped and saved";
            break;
    }

}

/**
 * Save application configuration.
 *
 * @param $params
 *
 * @return bool
 */
function saveParams($params) {
    $config = "<?php\n";
    $config .= "\$appsConfig = " .
var_export($params, true) . ";\n";
    $config .= "?>";
    $configFileName = '/config_' .
trim(str_replace('.', '_',
```

```
$_REQUEST['DOMAIN'])) . '.php';
    file_put_contents(getcwd() .
$configFileName, $config);
    return true;
}
/**
 * Send rest query to Bitrix24.
 *
 * @param       $method - Rest method, ex:
methods
 * @param array $params - Method params, ex:
array()
 * @param array $auth   - Authorize data,
ex: array('domain' =>
'https://test.bitrix24.com', 'access_token'
=> '7inpwszbuu8vnwr5jmabqa467rqur7u6')
 *
 * @return mixed
 */
function restCommand($method, array $params
= array(), $auth_domain, $access_token) {
    $queryUrl  = 'https://' . $auth_domain .
'/rest/' . $method;
    $queryData =
http_build_query(array_merge($params,
array('auth' => $access_token)));
    writeToLog(array('URL' => $queryUrl,
'PARAMS' => array_merge($params,
array("auth" => $access_token))), 'telephony
send data');
    $curl = curl_init();
    curl_setopt_array($curl, array(
        CURLOPT_SSL_VERIFYPEER => 0,
        CURLOPT_POST           => 1,
        CURLOPT_HEADER         => 0,
        CURLOPT_RETURNTRANSFER => 1,
        CURLOPT_URL            => $queryUrl,
```

```
        CURLOPT_POSTFIELDS     =>
$queryData,
        CURLOPT_VERBOSE         => 1
    ));
    $result = curl_exec($curl);
    writeToLog(array('raw' => $result),
'telephony got data');
    curl_close($curl);
    $result = json_decode($result, 1);
    return $result;
}

/**
 * Write data to log file.
 *
 * @param mixed  $data
 * @param string $title
 *
 * @return bool
 */
function writeToLog($data, $title = '') {
    $log = "\n------------------------\n";
    $log .= date("Y.m.d G:i:s") . "\n";
    $log .= (strlen($title) > 0 ? $title :
'DEBUG') . "\n";
    $log .= print_r($data, 1);
    $log .= "\n------------------------\n";
    file_put_contents(getcwd() . '/tel.log',
$log, FILE_APPEND);
    return true;
}

?>
```

voximplant.url.get

Возвращает набор ссылок для навигации по страницам телефонии. Метод не имеет ограничений по правам.

Параметры функции

Входных параметров нет.

Описание результата

Параметр	Описание
detail_statistics	Страница детальной статистики (таблица).
buy_connector	Страница для покупки SIP коннектора.
edit_config	Страница для настройки подключенной линии (SIP-номера), `#CONFIG_ID#` нужно заменить на необходимый идентификатор настройки.

Пример

```
BX24.callMethod(
    'voximplant.url.get',
    {},
```

```
        function(result)
        {
                    if(result.error())

console.error(result.error());
                    else

console.info(result.data());
        }
    );
```

voximplant.sip.get

Описание и пример

Возвращает список всех sip-линий, созданных приложением. Списочный метод. Метод доступен обладателю права **Управление номерами - изменение - любые**.

Пример

```
BX24.callMethod(
      'voximplant.sip.get',
      {
         "FILTER": {"CONFIG_ID":12},
         "SORT": "CONFIG_ID",
         "ORDER": "DESC",
      },
      function(result)
      {
                if(result.error())

console.error(result.error());
                else

console.info(result.data());
      }
   );
```

Параметры

Параметр	Описание
FILTER	Поля для сортировки.
SORT	По какому полю производится сортировка.
ORDER	Порядок сортировки (ASC/DESC).
Все параметры не обязательные	

Описание результата

Параметр	Описание
CONFIG_ID	Идентификатор настройки sip-линии.
TYPE	Тип АТС (см. список типов АТС).
TITLE	Название подключения.
SERVER	Адрес сервера sip-регистрации.
LOGIN	Логин для сервера.
PASSWORD	Пароль для сервера.
REG_ID	Идентификатор sip-регистрации (**только для Облачной АТС**).
INCOMING_SERVER	Адрес сервера для подключения при исходящем звонке (**только для Облачной АТС**).
INCOMING_LOGIN	Логин для подключения (**только для Облачной АТС**).
INCOMING_PASSWORD	Пароль для подключения (**только**

	для Облачной АТС).

voximplant.sip.add

Описание и пример

Создает новую sip-линию с привязкой к приложению. После создания данная линия становится исходящей линией по умолчанию. Метод доступен обладателю права **Управление номерами - изменение - любые**.

Пример

```
   BX24.callMethod(
      'voximplant.sip.add',
      {
         "TYPE": "cloud",
         "TITLE": "sipnet",
         "SERVER": "sipnet.ru",
         "LOGIN": "YYYYY",
         "PASSWORD": "ZZZZZ"
      },
      function(result)
      {
                if(result.error())

console.error(result.error());
                else

console.info(result.data());
      }
   );
```

Параметры

Параметр	Описание
TYPE	Тип АТС (см. список типов АТС, необязательный параметр, по умолчанию: **Облачная АТС**)
TITLE	Название подключения.
SERVER	Адрес сервера sip-регистрации.
LOGIN	Логин для сервера.
PASSWORD	Пароль для сервера.

Описание результата

Параметр	Описание
CONFIG_ID	Идентификатор настройки sip-линии.
TYPE	Тип АТС (см. список типов АТС).
TITLE	Название подключения.
SERVER	Адрес сервера sip-регистрации для Облачной АТС или адрес сервера Офисной АТС.
LOGIN	Логин для сервера.
PASSWORD	Пароль для сервера.
REG_ID	Идентификатор sip-регистрации (**только для Облачной АТС**).

INCOMING_SERVER	Адрес сервера для подключения при исходящем звонке (**только для Облачной АТС**).
INCOMING_LOGIN	Логин для подключения (**только для Облачной АТС**).
INCOMING_PASSWORD	Пароль для подключения (**только для Облачной АТС**).

Коды специфических ошибок

Код	Описание
MAX_CLOUD_PBX	Вы не можете подключить более 5 облачных АТС.
TITLE_EXISTS	Линия с таким названием уже существует.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

voximplant.callback.start

Описание

Метод запускает обратный звонок. Метод доступен обладателю права **Исходящий звонок - Выполнение - любые**.

Алгоритм обратного звонка выглядит так:

0. Клиент заполняет некую форму на сайте, указывает свой номер.

1. По факту заполнения формы, стороннее приложение запускает rest-апи метод.

2. Система выполняет **входящий** звонок на указанную в параметре FROM_LINE линию, в соответствии с настройками линии и дожидается соединения с менеджером. входящий звонок - настоящий, со всеми правилами обработки. т.е. если, например, на линии включена переадресация на мобильный - звонок пойдет на мобильный.

3. После того, как менеджер возьмет трубку, система произносит для менеджера текст, указанный в параметре TEXT_TO_PRONOUNCE, голосом, указанным в параметре VOICE. Это необходимо, чтобы менеджер понял, что ему поступил не обычный входящий звонок, а именно обратный звонок.

4. Система выполняет исходящий звонок на номер, указанный в параметре TO_NUMBER, и, после того, как клиент возьмет трубку, соединяет его с менеджером.

Для доступа к методу приложение должно запросить право доступа **Совершение звонков (call)**. Право указывается при регистрации приложения.

Параметры

Параметр	Описание
FROM_LINE	ID линии, с которой будет выполняться звонок. Список доступных линий можно получить методом voximplant.line.get.
TO_NUMBER	Номер, на который звонить.
TEXT_TO_PRONOUNCE	Текст, который произносится менеджеру перед началом звонка.
VOICE	Голос, которым произнести этот текст (необязательный). Список голосов можно получить методом voximplant.tts.voices.get.

Пример

```
BX24.callMethod(
        'voximplant.callback.start',
        {
                "FROM_LINE": "reg1332",
                "TO_NUMBER": "7911xxxxxxx",
                "TEXT_TO_PRONOUNCE": "Вам
поступил запрос на обратный звонок, соединяю
с клиентом.",
                "VOICE": "ruinternalfemale"
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else
```

```
console.info(result.data());
        }
);
```

voximplant.infocall.startwithso

Осуществляет звонок на указанный номер с проигрыванием файла формата mp3 по URL. Метод доступен обладателю права **Исходящий звонок - Выполнение - любые**.

Для доступа к методу приложение должно запросить право доступа **Совершение звонков (call)**. Право указывается при регистрации приложения.

Параметры функции

Параметр	Описание
FROM_LINE	ID линии, с которой будет выполняться звонок. Список доступных линий можно получить методом voximplant.line.get.
TO_NUMBER	Номер, на который звонить.
URL	Адрес mp3-записи для проигрывания.

Пример

```
BX24.callMethod(

'voximplant.infocall.startwithsound',
        {
```

```
                "FROM_LINE": "reg1332",
                "TO_NUMBER": "7911xxxxxxx",
                "URL":
"http://your.domain/path/file.mp3",
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.info(result.data());
        }
);
```

voximplant.infocall.startwithte

Осуществляет звонок на указанный номер с автоматическим произнесением заданного текста. Метод доступен обладателю права **Исходящий звонок - Выполнение - любые**.

Для доступа к методу приложение должно запросить право доступа **Совершение звонков (call)**. Право указывается при регистрации приложения.

Параметры функции

Параметр	Описание
FROM_LINE	ID линии, с которой будет выполняться звонок. Список доступных линий можно получить методом voximplant.line.get.
TO_NUMBER	Номер, на который звонить.
TEXT_TO_PRONOUNCE	Текст для произнесения.
VOICE	Голос, которым произнести этот текст (необязательный). Список голосов можно получить методом voximplant.tts.voices.get.

Пример

```
BX24.callMethod(
        'voximplant.infocall.startwithtext',
        {
                "FROM_LINE": "reg1332",
                "TO_NUMBER": "7911xxxxxxx",
                "PRONOUNCE": "Добрый день.
Ваша заявка выполнена",
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.info(result.data());
        }
);
```

voximplant.tts.voices.get

Возвращает массив доступных голосов для синтеза речи в формате ID голоса => название голоса. Метод не имеет ограничений по правам.

Параметры функции

Входящих параметров нет.

Пример

```
BX24.callMethod(
        'voximplant.tts.voices.get',
        {},
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.info(result.data());
        }
);
```

voximplant.user.activatePhone

Метод устанавливает сотруднику признак наличия sip-аппарата. Метод проверяет наличие права на модификацию пользователя.

Метод доступен обладателю права **Настройки пользоваля - Изменение** в соответствии с его значением этого права.

Параметры

Параметр	Описание	Тип
USER_ID	Идентификатор пользователя	int

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

voximplant.user.deactivatePho

Метод отключает сотруднику признак наличия sip-аппарата. Метод проверяет наличие права на модификацию пользователя.

Метод доступен обладателю права **Настройки пользоваля - Изменение** в соответствии с его значением этого права.

Параметры

Параметр	Описание	Тип
USER_ID	Идентификатор пользователя	int

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

voximplant.user.get

Метод возвращает настройки пользователей. Метод проверяет наличие права на модификацию пользователя и запрашивает подтверждение администратора перед выполнением.

Метод доступен обладателю права **Настройки пользователя - Изменение** в соответствии с его значением этого права.

Параметры

Параметр	Описание	Тип
USER_ID	Массив идентификаторов пользователей	int

Результат

Результатом будет массив записей с настройками пользователей.

Поля записи	Описание
ID	Идентификатор пользователя
DEFAULT_LINE	Номер для исходящих по умолчанию
INNER_NUMBER	Внутренний номер
PHONE_ENABLED	Флаг наличия у пользователя sip-аппарата

	(Y|N).
SIP_SERVER	Адрес сервера для подключения sip-аппарата.
SIP_LOGIN	Логин для подключения sip-аппарата.
SIP_PASSWORD	Пароль sip-аппарата.

voximplant.sip.update

Описание

Обновляет существующую sip-линию (созданную приложением). Метод доступен обладателю права **Управление номерами - изменение - любые**.

Описание результата

Возвращает 1 при успешном выполнении или исключение.

Параметры

Параметр	Описание
CONFIG_ID	Идентификатор настройки sip-линии.
TYPE	Тип АТС (см. список типов АТС, необязательный параметр, по умолчанию - **Облачная АТС**).
TITLE	Название подключения (необязательное поле).
SERVER	Адрес сервера sip-регистрации (необязательное поле).
LOGIN	Логин для сервера (необязательное поле).
PASSWORD	Пароль для сервера (необязательное поле).

Для успешного вызова необходимо наличие одного из полей: **TITLE**, **SERVER**, **LOGIN**, **PASSWORD**.

Коды специфических ошибок

Код	Описание
TITLE_EXISTS	Линия с таким названием уже существует.

Пример

```
BX24.callMethod(
            "voximplant.sip.update",
            {
               "CONFIG_ID": 69,
               "TITLE": "название линии",

            },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else

console.info(result.data());
            }
         );
```

© «Битрикс», 2001-2008, «1С-

voximplant.sip.delete

Удаляет существующую sip-линию (созданную приложением). Метод доступен обладателю права **Управление номерами - изменение - любые**.

Параметры функции

Параметр	Описание
CONFIG_ID	Идентификатор настройки sip-линии.

Описание результата

Возвращает 1 при успешном выполнении или исключение.

Пример

```
BX24.callMethod(
   "voximplant.sip.delete",
   {
      "CONFIG_ID": 87,
   },
   function(result)
   {
       if(result.error())
```

```
console.error(result.error());
                else

console.info(result.data());
            }
         );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

voximplant.sip.status

Возвращает текущий статус sip-регистрации (**только для облачных АТС**). Метод доступен обладателю права **Управление номерами - изменение - любые**.

Параметры функции

Параметр	Описание
REG_ID	Идентификатор sip-регистрации.

Описание результата

Код	Описание
REG_ID	Идентификатор sip-регистрации.
LAST_UPDATED	Дата последнего изменения sip-регистрации.
ERROR_MESSAGE	Текстовое описание кода ошибки.
STATUS_CODE	Цифровой код ошибки.
STATUS_RESULT	Состояние sip-регистрации (см. таблицу состояний).

Пример

```
BX24.callMethod(
            "voximplant.sip.status",
            {
               "REG_ID": 5505,
            },
            function(result)
            {
                if(result.error())

console.error(result.error());
                else

console.info(result.data());
            }
         );
```

voximplant.sip.connector.statu

Возвращает текущий статус SIP-Коннектора. Метод доступен обладателю права **Управление номерами - изменение - любые**.

Параметры функции

Входных параметров нет.

Описание результата

Код	Описание
FREE_MINUTES	Количество бесплатных минут для настройки и тестирования интеграции.
PAID	Оплачен или нет коннектор.
PAID_DATE_END	До какой даты оплачен коннектор (если факт оплаты был).

Пример

```
BX24.callMethod(
   'voximplant.sip.connector.status',
```

```
        {},
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.info(result.data());
        }
    );
```

voximplant.line.outgoing.sip.se

Установка выбранной sip-линии в качестве исходящей линии по-умолчанию. Метод доступен обладателю права **Управление номерами - изменение - любые**.

Параметры функции

Параметр	Описание
CONFIG_ID	Идентификатор настройки sip-линии.

Описание результата

Возвращает 1 при успешном выполнении.

Пример

```
BX24.callMethod(

"voximplant.line.outgoing.sip.set",
            {
               "CONFIG_ID": 57,
            },
            function(result)
```

```
            {
                if(result.error())

console.error(result.error());
                else

console.info(result.data());
            }
        );
```

voximplant.line.get

Возвращает список всех доступных исходящих линий. Метод доступен обладателю права **Управление номерами - изменение - любые**.

Пример

```
BX24.callMethod(
      'voximplant.line.get',
      {},
      function(result)
      {
                if(result.error())

console.error(result.error());
                else

console.info(result.data());
      }
   );
```

voximplant.line.outgoing.set

Установка выбранной линии в качестве исходящей линии по умолчанию. Метод доступен обладателю права **Настройки телефонии - изменение - любые**.

Параметры функции

Параметр	Описание
LINE_ID	Идентификатор линии, полученный от метода voximplant.line.get или voximplant.line.outgoing.get.

Описание результата

Возвращает 1 при успешном выполнении.

Пример

```
BX24.callMethod(

"voximplant.line.outgoing.set",
                    {
                       "LINE_ID": 55,
                    },
```

```
                    function(result)
                    {
                  if(result.error())

console.error(result.error());
                  else

console.info(result.data());
                    }
                );
```

voximplant.line.outgoing.get

Возвращает текущую выбранную линии в качестве исходящей линии по умолчанию. Метод доступен обладателю права **Управление номерами - изменение - любые**.

Параметры функции

Входных параметров нет.

Описание результата

Возвращает идентификатор линии (цифровой для арендованных, regXXX - для Облачных АТС, sipXXX - для Офисных АТС).

Пример

```
   BX24.callMethod(
      'voximplant.line.outgoing.get',
      {},
      function(result)
      {
                if(result.error())

console.error(result.error());
                else
```

```
console.info(result.data());
        }
    );
```

voximplant.statistic.get

Описание

Возвращает список истории звонков. Списочный метод. Метод доступен обладателю права **Статистика звонков - Просмотр** в соответствии с его значением этого права.

Параметры

Параметр	Описание
FILTER	Поля для сортировки.
SORT	По какому полю производится сортировка.
ORDER	Порядок сортировки (ASC/DESC).
Все параметры не являются обязательными	

Описание результата

Параметр	Описание
CALL_ID	Идентификатор звонка.
ID	Идентификатор звонка (для внутренних целей).
CALL_TYPE	Тип вызова (см. описание типов

	звонка).
CALL_VOTE	По умолчанию - 0. Оценка звонка используется только для внутренней телефонии.
COMMENT	Комментарий к звонку.
PORTAL_USER_ID	Идентификатор ответившего оператора (если это тип звонка: 2 - Входящий) или идентификатор позвонившего оператора (если это тип звонка: 1 - Исходящий).
PORTAL_NUMBER	Номер, на который поступил звонок (если это тип звонка: 2 - Входящий) или номер, с которого был совершен звонок (1 - Исходящий).
PHONE_NUMBER	Номер, с которого звонит абонент (если это тип звонка: 2 - Входящий) или номер, которому звонит оператор (1 - Исходящий).
CALL_DURATION	Продолжительность звонка в секундах.
CALL_START_DATE	Время инициализации звонка. При сортировке по этому полю нужно указывать дату в формате ISO-8601.
COST	Стоимость звонка.
COST_CURRENCY	Валюта звонка (RUR, USD, EUR).
CALL_FAILED_CODE	Код вызова (см. таблицу кодов вызова).
CALL_FAILED_REASON	Текстовое описание кода вызова (латиница).
CRM_ACTIVITY_ID	Идентификатор дела CRM, созданного на основании звонка.

CRM_ENTITY_ID	Идентификатор объекта CRM, к которому прикреплено дело.
CRM_ENTITY_TYPE	Тип объекта CRM, к которому прикреплено дело, например: LEAD.
REST_APP_ID	Идентификатор приложения интеграции внешней телефонии.
REST_APP_NAME	Название приложения интеграции внешней телефонии.
REDIAL_ATTEMPT	Номер попытки дозвониться (для обратных звонков)
SESSION_ID	Идентификатор сессии звонка на стороне Voximplant.
TRANSCRIPT_ID	Идентификатор расшифровки звонка
TRANSCRIPT_PENDING	Y\N. Признак того, что расшифровка будет получена позднее.
RECORD_FILE_ID	Идентификатор файла с записью звонка.

Пример

```
BX24.callMethod(
   'voximplant.statistic.get',
   {
      "FILTER": {">CALL_DURATION":60},
      "SORT": "CALL_DURATION",
      "ORDER": "DESC",
   },
   function(result)
   {
            if(result.error())
```

```
console.error(result.error());
                else

console.info(result.data());
      }
   );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

OnVoximplantCallInit

Событие вызывается при инициализации звонка (о поступлении или начале исходящего звонка).

Обратите внимание, что событие будет вызвано без данных авторизации.

В событие поступает массив с данными:

Параметр	Описание
CALL_ID	Идентификатор звонка.
CALL_TYPE	Тип вызова (см. описание типов вызова).
ACCOUNT_SEARCH_ID	Идентификатор линии - цифровой для арендованных, regXXX - для Облачных АТС, sipXXX - для Офисных АТС.
PHONE_NUMBER	Номер, которому звонит оператор (если тип звонка: 1 - Исходящий) или номер, на который звонит абонент (если тип звонка: 2 - Входящий).
CALLER_ID	Идентификатор линии (если тип звонка: 1 - Исходящий) или номер телефона, который позвонил на

	портал (если тип звонка: 2 - Входящий).

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

OnVoximplantCallStart

Событие вызывается при начале разговора (ответе оператора при входящем и ответе абонента при исходящем).

Обратите внимание, что событие будет вызвано без данных авторизации.

В событие поступает массив с данными:

Параметр	Описание
CALL_ID	Идентификатор звонка.
USER_ID	Идентификатор пользователя, который ответил.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

OnVoximplantCallEnd

Событие вызывается при окончании разговора (запись в историю).

Обратите внимание, что событие будет вызвано без данных авторизации.

В событие поступает массив с данными:

Параметр	Описание
CALL_ID	Идентификатор звонка.
CALL_TYPE	Тип вызова (см. описание типов вызова).
PHONE_NUMBER	Номер, с которого звонит абонент (если тип звонка 2 - Входящий) или номер, которому звонит оператор (если тип звонка 1 - Исходящий).
PORTAL_NUMBER	Номер на который поступил звонок (если тип звонка 2 - Входящий) или номер, с которого был совершен звонок (если тип звонка 1 - Исходящий).
PORTAL_USER_ID	Идентификатор ответившего оператора (если тип звонка 2 - Входящий) или идентификатор

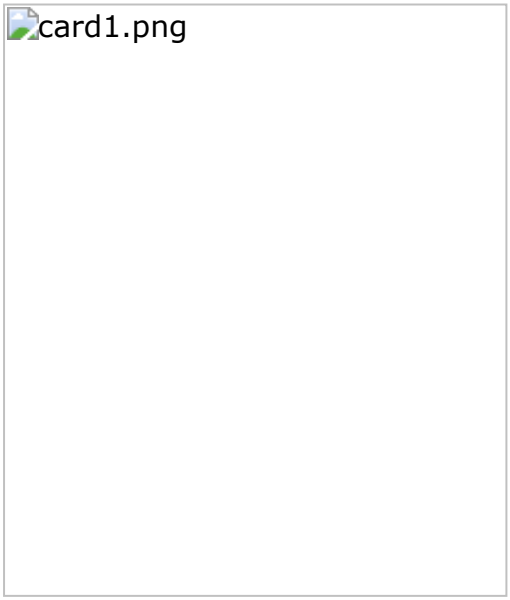
	позвонившего оператора (если тип звонка 1 - Исходящий).
CALL_DURATION	Длительность звонка.
CALL_START_DATE	Дата в ISO формате.
COST	Стоимость звонка.
COST_CURRENCY	Валюта звонка (RUR, USD, EUR).
CALL_FAILED_CODE	Код вызова (см. таблицу кодов вызова).
CALL_FAILED_REASON	Текстовое описание кода вызова (латиница).
CRM_ACTIVITY_ID	ID дела crm, связанного со звонком.

© «Битрикс», 2001-2008, «1С-

Карточка звонка

Общее описание

Рассмотрим карточку звонка:



Чтобы изменить титульник карточки (область 1), требуется вызвать метод CallCardSetCardTitle и передать объект со свойством **title**.

Пример:

```
BX24.placement.call('CallCardSetCardTitle',
{title: 'Card Title'}, () => { //some code
});
```

Чтобы изменить текст в области 2, требуется вызвать метод CallCardSetStatusText и передать объект со свойством **statusText**.

Пример:

```
BX24.placement.call('CallCardSetStatusText',
{statusText: 'Status Text'}, () => { //some
code });
```

Всего у карточки звонка 14 состояний интерфейса. Получить их можно с помощью вызова метода CallCardGetListUiStates. В функцию обратного вызова будет передан массив с доступными состояния карточки звонка.

Пример:

```
BX24.placement.call('CallCardGetListUiStates
', (data) => { console.log(data); });
```

Переход на другое состояние карточки осуществляется вызовом метода CallCardSetUiState с передачей туда объекта со свойством **uiState**.

Пример:

```
BX24.placement.call('CallCardSetUiState', {
uiState: 'connected'}, () => { //some code
});
```

Чтобы обрабатывать нажатия оператором кнопок в карточке звонка, требуется подписаться на соответствующие события.

Состояния карточки

Состояние	Описание	Обраб
[dw]incoming[/dw][di] card01-02.png [/di]	Для принятия входящих звонков	▪ Ответить ▪ Пропусти
[dw]transferIncoming[/dw][di] card01-02.png [/di]	Для принятия перенаправленного входящего вызова	▪ Ответить ▪ Пропусти
[dw]outgoing[/dw][di]	Для показа карточки	▪ Позвони

card03.png [/di]	исходящего звонка	Backgrou
[dw]connectingIncoming[/dw] [di] card04-05-09.png [/di]	Для показа карточки в момент подключения к входящему звонку	▪ Заверши Backgrou
[dw]connectingOutgoing[/dw][di]	Для показа карточки в момент подключения к исходящему звонку	▪ Заверши Backgrou

card04-05-09.png [/di]		

[dw]connected[/dw][di]
card06.png
[/di]
	Для показа после подключения к звонку	▪ Заверши
		`Backgrou`
		▪ Поставит
		`Backgrou`
		▪ Выключи
		`Backgrou`
		▪ Перенап
		`Backgrou`
		▪ Нажатие
		`Backgrou`
		▪ Оценить
		`Backgrou`
		
		Также, в этом
		приложения в
		микрофона и
		методов `Call`
		соответственн
		
		Пример:
		
		`BX24.placem`
		`true }, ()`
		`// значение`
		`false - на`

		BX24.placem
		}, () => {
		// значение
		удержание,

[dw]transferring[/dw][di] card07.png [/di]	Для подтверждения перенаправления звонка на другого оператора	▪ Перенап Backgrou ▪ Вернуть Backgrou
[dw]transferFailed[/dw][di] card08.png [/di]	Если перенаправить звонок не получилось	▪ Вернуть Backgrou
[dw]transferConnected[/dw][di]	Если	

card04-05-09.png [/di]	перенаправление завершилось успешно и требуется выйти из карточки звонка	▪ Заверши Backgrou
[dw]error[/dw][di] card010.png [/di]	Если произошла некоторая ошибка	▪ Закрыть
[dw]moneyError[/dw][di]	Если на счету закончились деньги и требуется проиформировать об этом администратора портала	▪ Уведоми Backgrou ▪ Закрыть

card011.png [/di]		
[dw]redial[/dw][di] card012.png [/di]	Если абонент занят, дать возможность оператору повторно позвонить на этот номер, не скрывая карточку звонка	▪ Перезво `Backgrou`
Таймер в карточке звонка	По умолчанию, при переходе на сос включается таймер звонка. Данное помимо `uiState: 'connected'` еще значением `true`. При переходе же н останавливаться.	

События

Полный список событий:

Событие	Описание
BackgroundCallCard::initialized	Происходит после создания карточки звонка. в функцию обратного вызова будет передан номер телефона.
BackgroundCallCard::addCommentButtonClick	Происходит при сохранении комментария в карточке звонка. В функцию обратного вызова передается текст комментария.
BackgroundCallCard::muteButtonClick	Происходит по нажатию на кнопку выключения микрофона. В функцию обратного вызова передается

	булево значение: **true** - ожидается выключение, **false** - ожидается включение.
BackgroundCallCard::holdButtonClick	Происходит по нажатию на кнопку удержания звонка. В функцию обратного вызова передается булево значение: **true** - ожидается включение удержания, **false** - ожидается выключение удержания.
BackgroundCallCard::closeButtonClick	Происходит по нажатию на кнопку завершения звонка. В функцию обратного вызова ничего не передается.
BackgroundCallCard::transferButtonClick	Происходит при выборе оператора, на которого текущий оператор хочет перевести звонок. В функцию обратного вызова передается объект со

	свойствами **phoneNumber** - номер текущего звонка и **target** - идентификатор пользователя Битрикс24.
BackgroundCallCard::cancelTransferButtonClick	Нажатие на кнопку "вернуться к звонку". В функцию обратного вызова ничего не передается.
BackgroundCallCard::completeTransferButtonClick	Нажатие на кнопку "перенаправить". В функцию обратного вызова ничего не передается.
BackgroundCallCard::completeTransferButtonClick	Нажатие на кнопку "завершить". В функцию обратного вызова ничего не передается.
BackgroundCallCard::nextButtonClick	Нажатие на кнопку "следующий". В функцию обратного вызова ничего не передается.
BackgroundCallCard::skipButtonClick	Нажатие на кнопку "пропустить". В функцию обратного

	вызова ничего не передается.
BackgroundCallCard::answerButtonClick	Нажатие на кнопку "ответить". В функцию обратного вызова ничего не передается.
BackgroundCallCard::entityChanged	Происходит в карточке обзвона при смене текущего обзваниемого. В функцию обратного вызова ничего не передается.
BackgroundCallCard::entityChanged	Нажатие на кнопку "перезвонить". В функцию обратного вызова ничего не передается.
BackgroundCallCard::qualityMeterClick	Происходит при оценке качества связи. В функцию обратного вызова передается целочисленное значение от 1 до 5.
BackgroundCallCard::dialpadButtonClick	Происходит по нажатию на одну из цифровых кнопок телефона. В функцию

	обратного вызова передается нажатый символ.
BackgroundCallCard::notifyAdminButtonClick	Нажатие на кнопку "уведомить администратора". В функцию обратного вызова ничего не передается.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

CallCardSetMute

Метод позволяет со стороны приложения выключить микрофон оператора.

Приложение должно передавать в метод объект со свойством **muted**, в котором находится булево значение: **true** - микрофон должен быть выключен, **false** - включен. В функцию обратного вызова не передаются какие-либо данные.

Пример вызова

```
BX24.placement.bindEvent('BackgroundCallCard
::initialized', event => {

BX24.placement.call('CallCardSetMute', {
muted: true }, () => {
// some code
         });
     });
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

CallCardSetHold

Метод позволяет со стороны приложения поставить звонок на удержание.

Приложение должно передавать в метод объект со свойством **held**, в котором находится булево значение: **true** - включить удержание, **false** - выключить. В функцию обратного вызова не передаются какие-либо данные.

Пример вызова

```
BX24.placement.bindEvent('BackgroundCallCard
::initialized', event => {

BX24.placement.call('CallCardSetHold', {
held: true }, () => {
            // some code
        });
 });
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

CallCardSetUiState

Метод позволяет со стороны приложения изменить состояние интерфейса карточки звонка.

Приложение должно передавать в метод объект со свойством **uiState**, значение которого должно одним из получаемых из метода **getListUiStates**. В функцию обратного вызова не передаются какие-либо данные.

Пример вызова

```
BX24.placement.bindEvent('BackgroundCallCard
::initialized', event => {
   BX24.placement.call(CallCardSetUiState',
{uiState: 'connected'}, () => {
      // some code
   })
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

CallCardGetListUiStates

Метод позволяет получить список доступных состояний интерфейса карточки звонка.

При вызове в функцию обратного вызова передается объект со всеми возможными состояниями интерфейса карточки.

Пример вызова

```
BX24.placement.bindEvent('BackgroundCallCard
::initialized', event => {

BX24.placement.call('CallCardGetListUiStates
', data => {
       // some code
    })
});
```

CallCardSetCardTitle

Метод позволяет изменить со стороны приложения титульник карточки звонка.

При вызове требуется передать объект со свойством **title**. В функцию обратного вызова ничего не передается.

Пример вызова

```
BX24.placement.bindEvent('BackgroundCallCard
::initialized', event => {

BX24.placement.call('CallCardSetCardTitle',
{ title: 'hello world! }, () => {
       // some code
    })
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

CallCardSetStatusText

Метод позволяет со стороны приложения изменить текст в центре карточки звонка.

При вызове требуется передать объект со свойством **statusText**. В функцию обратного вызова ничего не передается.

Пример вызова

```
BX24.placement.bindEvent('BackgroundCallCard
::initialized', event => {

BX24.placement.call('CallCardSetStatusText',
{ statusText: 'hello world! }, () => {
        // some code
    })
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

CallCardClose

Метод позволяет со стороны приложения закрыть карточку звонка.

Внимание! После вызова данного метода карточка звонка закрывается и дальнейшие операции с ней произвести не получится..

Пример вызова

```
BX24.placement.bindEvent('BackgroundCallCard
::initialized', event => {
    BX24.placement.call('CallCardClose', {},
() => {
        // some code
    })
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Торговый каталог::Ресурс торгового каталога

Торговый каталог:


```
{
        "catalog": {
                "iblockId": 13,
                "iblockTypeId": 0,
                "id": 13,
                "lid": "s1",
                "name": "Товарный каталог
CRM",
                "productIblockId": null,
                "skuPropertyId": null,
                "subscription": "N",
                "vatId": 1,
                "yandexExport": "N"
        }
}
```

Торговый каталог::catalog.catalog.add

```
catalog.catalog.add(fields)
```

Метод добавляет торговый каталог в коллекцию торговых каталогов.

Если операция успешна, возвращается ресурс торгового каталога в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'catalog.catalog.add',
        {
        fields: {
                                iblockId: 1
```

```
        }
    },
    function(result) {
        if (result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
    });
```

Торговый каталог::catalog.catalog.delete

```
catalog.catalog.delete(id)
```

Метод удаляет торговый каталог из коллекции торговых каталогов.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер торгового каталога.

Примеры

```
BX24.callMethod(
    'catalog.catalog.delete',
    {
      id: 14
    },
    function(result)
    {
```

```
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Торговый каталог::catalog.catalog.get

```
catalog.catalog.get(id)
```

Метод для доступа к значению полей торгового каталога по ID.

Если операция успешна, возвращается ресурс торгового каталога в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер торгового каталога.

Примеры

```
BX24.callMethod(
    'catalog.catalog.get',
    {
      id: 13
    },
    function(result)
```

```
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Торговый каталог::catalog.catalog.getFie

```
catalog.catalog.getFields()
```

Метод возвращает поля торгового каталога.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.catalog.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание	Тип	Примечание
iblockId	Идентификатор информационного блока	integer	Неизменяемое, обязательное поле.
iblockTypeId	Тип информационного блока	integer	Только для чтения.
id	Идентификатор каталога	integer	Только для чтения.
lid	Сайт	string	Только для чтения.
name	Название	string	Только для чтения.
productIblockId	Идентификатор родительского информационного блока товаров	integer	Заполнено только у информационного блока торговых предложений.
skuPropertyId	Идентификатор свойства, в котором храниться ссылка на родительский элемент информационного блока Товаров	integer	Заполнено только у информационного блока торговых предложений.
subscription	Продажа контента	char	
vatId	Идентификатор НДС	integer	
yandexExport	Экспортировать в	char	

	Яндекс.Товары		

Торговый каталог::catalog.catalog.isOffe

```
catalog.catalog.isOffers(id)
```

Метод проверки является ли указанный торговый каталог торговым каталогом предложений.

Если проверка успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер торгового каталога.

Примеры

```
BX24.callMethod(
    'catalog.catalog.isOffers',
    {
      id: 14
    },
    function(result)
```

```
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Торговый каталог::catalog.catalog.list

```
catalog.catalog.list(select, filter, order,
start)
```

Метод получает список торговых каталогов по фильтру.

Если операция успешна, возвращается список торговых каталогов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
start	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'catalog.catalog.list',
    {
      select:{
                id,
        },
          filter:{
                iblockId: 1
          },
          order:{
                id: ASC
          },
          start: 1
    },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Торговый каталог::catalog.catalog.updat

```
catalog.catalog.update(id, fields)
```

Метод для обновления полей каталога.

Если операция успешна, возвращается ресурс торгового каталога в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер торгового каталога.
fields	object	Поля, соответствующие доступному списку полей fields .

Примеры

```
BX24.callMethod(
    'catalog.catalog.update',
    {
```

```
            id: 13,
            fields: {
                  yandexExport: Y
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.enum.getRoundTypes

```
catalog.enum.getRoundTypes()
```

Метод возвращает список типов округления.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.enum.getRoundTypes',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс наценки

Наценка:

```
{
        "extra": {
                "id": 1,
                "name": "test",
                "percentage": "10.00"
        }
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.extra.get

```
catalog.extra.get(id)
```

Метод для доступа к значениям полей наценки по ID.

Если операция успешна, возвращается ресурс наценки в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер наценки.

Примеры

```
BX24.callMethod(
    'catalog.extra.get',
    {
      id: 7
    },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.extra.getFields

```
catalog.extra.getFields()
```

Метод возвращает поля наценки

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.extra.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание	Тип	Примечание
id	Идентификатор наценки	integer	Только для чтения.
name	Название	string	Обязательное поле.
percentage	Величина наценки	string	Обязательное поле.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.extra.list

```
catalog.extra.list(select, filter, order,
start)
```

Метод получает список наценок по фильтру.

Если операция успешна, возвращается список наценок в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
start	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'catalog.extra.list',
    {
      select:{
                id,
          },
          filter:{
                percentage: 10
          },
          order:{
                id: ASC
          },
      start: 1
    },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс единицы измерения

Единица измерения:

```
{
    "code": 6,
    "id": 1,
    "isDefault": "N",
    "measureTitle": null,
    "symbol": null,
    "symbolIntl": "m",
    "symbolLetterIntl": "MTR"
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.measure.get

```
catalog.measure.get(id)
```

Метод для доступа к значениям полей единицы измерения по ID.

Если операция успешна, возвращается ресурс единицы измерения в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер единицы измерения.

Примеры

```
BX24.callMethod(
    'catalog.measure.get',
    {
      id: 7
    },
    function(result)
    {
```

```
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.measure.getFields

```
catalog.measure.getFields()
```

Метод возвращает поля единицы измерения.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.measure.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание	Тип	Примечание
code	Код	integer	Обязательное поле.
id	Идентификатор единицы измерения	integer	Только для чтения.
isDefault	По умолчанию	char	
measureTitle	Наименование единицы измерения	string	Обязательное поле.
symbol	Условное обозначение	string	
symbolIntl	Условное обозначение (международное)	string	
symbolLetterIntl	Кодовое буквенное обозначение (международное)	string	

catalog.measure.list

```
catalog.measure.list(select, filter, order,
start)
```

Метод получает список единиц измерения по фильтру.

Если операция успешна, возвращается список единиц измерения в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
start	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'catalog.measure.list',
    {
      select:{
                id,
          },
          filter:{
                isDefault: N
          },
          order:{
                id: ASC
          },
     start: 1
    },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс цены Цена:


```
{
    "catalogGroupId": 1,
    "currency": "RUB",
    "extraId": null,
    "id": 122,
    "price": 2000,
    "priceScale": 2000,
    "productId": 8,
    "quantityFrom": null,
    "quantityTo": null,
    "timestampX": "2019-06-
13T17:29:26+03:00"
}
```

catalog.price.delete

```
catalog.price.delete(id)
```

Метод для удаления цены товара из коллекции цен товаров.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер цены товара.

Примеры

```
BX24.callMethod(
    'catalog.price.delete',
    {
      id: 56
    },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.price.get

```
catalog.price.get(id)
```

Метод для доступа к значению полей цены товара по ID.

Если операция успешна, возвращается ресурс цены товара в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер цены товара.

Примеры

```
BX24.callMethod(
    'catalog.price.get',
    {
      id: 122
    },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.price.getFields

```
catalog.price.getFields()
```

Метод возвращает поля цены товара.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.price.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание	Тип	Примечание
catalogGroupId	Тип цены	integer	Обязательное поле.
currency	Валюта	string	Обязательное поле.
extraId	Идентификатор наценки	integer	
id	Идентификатор цены	integer	Только для чтения.
price	Цена	double	Обязательное поле.
priceScale	Базовая цена	double	Только для чтения.
productId	Идентификатор товара	integer	Неизменяемое, обязательное поле.
quantityFrom	Количество от	integer	
quantityTo	Количество до	integer	
timestampX	Дата изменения	datetime	Только для чтения.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.price.list

```
catalog.price.list(select, filter, order,
start)
```

Метод получает список цен товаров по фильтру.

Если операция успешна, возвращается список цен в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
start	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'catalog.price.list',
    {
      select:{
                id
          },
          filter:{
                productId: 8
          },
          order:{
                id: ASC
          },
     start: 1
    },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.price.modify

```
catalog.price.modify(fields)
```

Метод для изменения элементов коллекции цен товара.

Внимание! Все сущности, которые не переданы или у которых не указаны ID, будут удалены.

Если операция успешна, возвращается ресурс цены в теле ответа.

Параметры

Параметр	Тип	Описание
fields.product.id	string	Номер цены.
fields.product.prices[]	list	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'catalog.price.modify',
    {
```

```
           fields: {
               product: {
                     id: 8
                         prices: [
                                 {

catalogGroupId: 1,
                                   currency:
RUB,
                                   price:
2001,

quantityFrom: 1,

quantityTo: 2
                                 },
                                 {

catalogGroupId: 1,
                                   currency:
RUB,
                                   price:
2001,

quantityFrom: 3,

quantityTo: 4
                                 },
                                 {

catalogGroupId: 1,
                                   currency:
RUB,
                                   price:
2001,

quantityFrom: 5,
```

```
                                        id:122
                                    },
                            ]
                    },
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс типа цены Тип цены:


```
{
        "base": "Y",
        "createdBy": 1,
        "dateCreate": "2018-10-
22T11:30:43+03:00",
        "id": 1,
        "modifiedBy": 1,
        "name": "BASE",
        "sort": 100,
        "timestampX": "2018-10-
22T11:31:16+03:00",
        "xmlId": "BASE"
}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.priceType.get

```
catalog.priceType.get(id)
```

Метод для доступа к значениям полей типа цены по ID.

Если операция успешна, возвращается ресурс типа цены в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер типа цены.

Примеры

```
BX24.callMethod(
    'catalog.priceType.get',
    {
      id: 7
    },
    function(result)
    {
```

```
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.priceType.getFields

```
catalog.priceType.getFields()
```

Метод возвращает поля типа цены.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.priceType.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание	Тип	Примечание
base	Является ли тип цены базовым	string	Обязательное, неизменяемое поле.
createdBy	Кем создан	integer	
dateCreate	Дата создания	datetime	
id	Идентификатор	integer	Только для чтения.
modifiedBy	Кем изменен	integer	
name	Название	string	
sort	Сортировка	integer	
timestampX	Дата изменения	datetime	
xmlId	Внешний код	string	

© «Битрикс», 2001-2008, «1С-

catalog.priceType.list

```
catalog.priceType.list(select, filter,
order, start)
```

Метод получает список типов цен по фильтру.

Если операция успешна, возвращается список типов цен в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
start	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'catalog.priceType.list',
    {
      select:{
                id
          },
          filter:{
                modifiedBy: 1
          },
          order:{
                id: ASC
          },
     start: 1
    },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс товара Товар:


```
{
        "active": "Y",
        "available": "Y",
        "bundle": "N",
        "canBuyZero": "Y",
        "code": "sports-suit-gentle-warmth",
        "createdBy": 1,
        "dateActiveFrom": null,
        "dateActiveTo": null,
        "dateCreate": "2018-10-
22T11:31:15+03:00",
        "detailPicture": {
                "id": "48",
                "url":
"/rest/catalog.product.download?
fields%5BfieldName%5D=detailPicture&fields%5
BfileId%5D=48&fields%5BproductId%5D=9",
                "urlMachine":
"/rest/catalog.product.download?
fields%5BfieldName%5D=detailPicture&fields%5
BfileId%5D=48&fields%5BproductId%5D=9"
        },
        "detailText": "Замечательный
спортивный костюм, состоящий из куртки и
брюк. Манжеты, талия, низ изделия и капюшон
регулируются в 2 направлениях одной рукой.
Капюшон сворачивается и убирается в ворот
под молнию. Изделия легко складываются в
```

маленькую сумочку, которая идет в комплекте к костюму.\n\t\t\t\t\t\t\t
\n\t\t\t\t\t\t\t
\n\t\t\t\t\t\t\t
\n\n\t\t\t\t\t\t\t
\n\t\t\t\t\t\t\t
Дополнительное описание:
\n\n\t\t\t\t\t\t\t
\n\t\t\t\t\t\t\t

\n\t\t\t\t\t\t\t

- **Вид застежки:** Липучка

\n\n\t\t\t\t\t\t\t

- **Длина рукава:** Длинные, 79.0 см

\n\n\t\t\t\t\t\t\t

- **Вид застежки:** Молния

\n\n\t\t\t\t\t\t\t

- **Тип карманов:** Втачные

\n\n\t\t\t\t\t\t\t

- **Габариты предметов:** Длина, 75.0 см

\n\n\t\t\t\t\t\t\t

- **Брюки:** Ширина брючин, 34.0 см

\n\n\t\t\t\t\t\t\t

- **Брюки:** Высота посадки, 28.0 см

\n\n\t\t\t\t\t\t\t

- **Брюки:** Длина по внутреннему шву, 78.0 см

\n\n\t\t\t\t\t\t\t

- **Сезон:** демисезон

\n\n\t\t\t\t\t\t\t

- **Пол:** Женский

\n\n\t\t\t\t\t\t\t

- **Стиль:** Спортивный стиль

\n\n\t\t\t\t\t\t\t

- **Страна бренда:** Швеция

\n\n\t\t\t\t\t\t\t
- **Страна производитель:** Китай
\n\t\t\t\t\t\t\t

\n\t\t\t\t\t\t\t
\n\t\t\t\t\t\t\t
\n\t\t\t\t\t\t",
"detailTextType": "html",
"height": null,
"iblockId": 13,
"iblockSectionId": 12,
"id": 9,
"length": null,
"measure": 5,
"modifiedBy": 1,
"name": "Спортивный Костюм Нежная Теплота",
"previewPicture": null,
"previewText": null,
"previewTextType": "text",
"property52": null,
"property53": null,
"property54": null,
"property55": null,
"property56": null,
"property57": null,
"property58": null,
"property59": {
"value": "189-01-xx",
"valueId": "87"
},
"property60": {
"value": "Китай "Гун Джой Ли"",
"valueId": "88"
},
"property61": [

```
                {
                        "value": "95%
хлопок, 5% эластан",
                        "valueId": "89"
                }
        ],
        "property62": null,
        "property63": [
                {
                        "value": {
                                "id": "49",
                                "url":
"/rest/catalog.product.download?
fields%5BfieldName%5D=property63&fields%5Bfi
leId%5D=49&fields%5BproductId%5D=9",

"urlMachine":
"/rest/catalog.product.download?
fields%5BfieldName%5D=property63&fields%5Bfi
leId%5D=49&fields%5BproductId%5D=9"
                        },
                        "valueId": "90"
                },
                {
                        "value": {
                                "id": "50",
                                "url":
"/rest/catalog.product.download?
fields%5BfieldName%5D=property63&fields%5Bfi
leId%5D=50&fields%5BproductId%5D=9",

"urlMachine":
"/rest/catalog.product.download?
fields%5BfieldName%5D=property63&fields%5Bfi
leId%5D=50&fields%5BproductId%5D=9"
                        },
                        "valueId": "91"
```

```
                }
        ],
        "property64": null,
        "property65": null,
        "property66": null,
        "property67": null,
        "property68": null,
        "property69": null,
        "property70": null,
        "property71": null,
        "property72": null,
        "property73": null,
        "property74": null,
        "property75": null,
        "property76": null,
        "property91": null,
        "property92": null,
        "property93": null,
        "property94": null,
        "property95": null,
        "property96": null,
        "property97": null,
        "property98": null,
        "property99": null,
        "property100": null,
        "property101": null,
        "property102": null,
        "property103": null,
        "property104": null,
        "property107": null,
        "property108": null,
        "property109": null,
        "property110": null,
        "property111": null,
        "property112": null,
        "property113": null,
        "property114": null,
```

```
        "property115": null,
        "property116": null,
        "property121": null,
        "property122": null,
        "property126": null,
        "property129": null,
        "property130": null,
        "property131": null,
        "property132": null,
        "property137": null,
        "property138": null,
        "purchasingCurrency": null,
        "purchasingPrice": null,
        "quantity": 99,
        "quantityReserved": null,
        "quantityTrace": "N",
        "sort": 420,
        "subscribe": "Y",
        "timestampX": "2018-10-
22T11:31:17+03:00",
        "type": 1,
        "vatId": null,
        "vatIncluded": "Y",
        "weight": null,
        "width": null,
        "xmlId": "1000000404"
}
```

catalog.product.add

```
catalog.product.add(fields)
```

Метод добавляет товар торгового каталога.

Если операция успешна, возвращается ресурс товара торгового каталога в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'catalog.product.add',
    {
          "fields":{
                "active":"Y",
                "bundle":"N",
                "canBuyZero":"Y",
```

```
                "code":"t-shirt",
                "createdBy":"1",
                "dateActiveFrom":"2019-06-
04T19:36:00+03:00",
                "dateActiveTo":"2019-06-
05T19:57:00+03:00",
                "dateCreate":"2018-10-
22T11:31:15+03:00",

"detailText":"\u041e\u0442\u043b\u0438\u0447
\u043d\u0430\u044f
\u0444\u0443\u0442\u0431\u043e\u043b\u043a\u
0430",
                "detailTextType":"html",
                "iblockId":"13",
                "iblockSectionId":"11",
                "length":"123",
                "measure":"5",
                "modifiedBy":"1",

"name":"\u0424\u0443\u0442\u0431\u043e\u043b
\u043a\u0430
\u0416\u0435\u043d\u0441\u043a\u0438\u0439
\u0421\u043e\u0431\u043b\u0430\u0437\u043d",
                "previewText":"previewText",
                "previewTextType":"text",
                "purchasingCurrency":"RUB",
                "purchasingPrice":"1000",
                "quantity":"100",
                "quantityReserved":"1",
                "quantityTrace":"N",
                "sort":"340",
                "subscribe":"Y",
                "vatId":"1",
                "vatIncluded":"Y",
                "weight":"10",
                "width":"20",
```

```
                "xmlId":"bx123123",
                "property91":{

"value":"\u0441\u0442\u0440\u043e\u043a\u043
0"
                        },
                "property92":[

{"value":"\u0441\u0442\u0440\u043e\u043a\u04
30_\u043c1"},

{"value":"\u0441\u0442\u0440\u043e\u043a\u04
30_\u043c2"}
                ],
                "property93":
{"value":"\u0447\u0438\u0441\u043b\u043e"},
                "property94":[

{"value":"\u0447\u0438\u0441\u043b\u043e1"},

{"value":"\u0447\u0438\u0441\u043b\u043e2"}
                ],
                "property95":{"value":"63"},
                "property96":[
                        {"value":"68"},
{"value":"69"}
                ],
                "property99":{"value":"25"},
                "property100":[
                        {"value":"13"},
                        {"value":"12"}
                ],
                "property101":{"value":"4"},
                "property102":[
                        {"value":"2"}
                ],
                "property107":
```

{"value":"2019-05-31T03:00:00+03:00"},
                "property108":[
                        {"value":"2019-06-
01T03:00:00+03:00"},
                        {"value":"2019-06-
02T03:00:00+03:00"}
                ],
                "property109":
{"value":"2019-05-31T16:10:00+03:00"},
                "property110":[
                        {"value":"2019-06-
01T16:10:00+03:00"},
                        {"value":"2019-06-
02T17:10:00+03:00"}
                ],
                "property111":
{"value":"1|RUB"},
                "property112":[
                        {"value":"1|RUB"},
                        {"value":"2|USD"}
                ],
                "property113":
{"value":"55.778579844603,37.66815246582"},
                "property114":[

{"value":"55.746059279807,37.576141967773"},

{"value":"55.705757975708,37.69012512207"}
                ],
                "property115":
{"value":"55.806334255977,37.591355457306"},
                "property116":[

{"value":"55.817909211916,37.653153553009"},

{"value":"55.789351426368,37.499344959259"}
                ],

```
                "property121":{"value":"1"},
                "property122":[
                        {"value":"1"}
                ],
                "property126":
{"value":"12"},
                "property129":
{"value":"C_11"},
                "property130":[
                        {"value":"C_10"},
                        {"value":"CO_2"}
                ],
                "property131":
{"value":"13"},
                "property132":[
                        {"value":"13"},
                        {"value":"12"}
                ],
                "property137":{"value":"1"},
                "property138":[
                        {"value":"1"},
                        {"value":"2"}
                ],
                "property103":{"value":
{"text":"\u0442\u0435\u043a\u0441\u0442","ty
pe":"HTML"}},
                "property104":{
                        "1":{"value":
{"text":"\u0442\u0435\u043a\u0441\u04421","t
ype":"HTML"}},
                        "2":{"value":
{"text":"\u0442\u0435\u043a\u0441\u04422","t
ype":"HTML"}}
                        },
                "height":"1",
                "property98":[
                        {"value":{
```

"fileData":[

"previewSmall.png",

"iVBORw0KGgoAAAANSUhEUgAAAEUAAABFCAYAAAAcjSs
pAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAA
JcEhZcwAADsQAAA7EAZUrDhsAAAqZSURBVHhe7VxbbBx
XGf7W6\/U6vjeOYztJkzSOG3KRk4CKUFEV1AdeEChcVO
4XNVJBFUgVlMtDIkDlAQRI3MVDVakVKioUBA9IPLUqJe
IWaEJEE4hShSp1LnYudnxde3f5vjNz7PHszO7s7KwvyJ
9yMuN\/ds7lm\/\/8lzNnN1UksIIosPVCAcjz2JBySrr
BvbhCWDZSZueB8WlgKgfMzAFzeWCeZASCPRIxjWmgOeO
Uzg1AS5N7vc6oKykTs8DoHeAOj3kSIC3gP\/OfOerUnn
hgOuT2Sgf1UBql
0XQpjZgY6tzvR5InBTVdmWMZEw454YId
BBBFQD1afO2inX1QJs6QSyJCpJJEaKNGH4NnBjctE2CL
USEQTbYx3nWdqywNau5KZXIqRcG3e0w5JRDyLCYLVHD0
V2Z2d37e3XRMo0jebFEcdzpJeZDD8MOSwiZ tdQE
7eyEGYpMi7dB0ydBDCCtJiBcajbxaezMw0OMKq0QsUi5
cp2ulR5HbXC1k
CGNUd\/29AJNja4wIqom5dUrfBKMMRpEiCtbrbBacy
JqcYIRyZFLlCE6NMiZK1A\/VWguItTSYY4CiIPzxDC41
oiRNAUkt17jQ5hko4hCiIN8ZzVkNU
X0JgiblwFcgx3aiEiqTI5Ur91pqG
CFilEudJzGVUHao15W3zKxdDfFDxGgo\/7nm\/B2GUFK
U1Q7fIrv8hCr7f4E0XkGn4qwwhHofGVab2YahyGsFSys
\/Ww4ptqKqVEyDcpH8Q
fR\/F90pNTvMrbDeqShbcFrN4GkKMN9Q1riRqtBECGZm
Ztou3aKxDShsbWA
RYmHsVgdlJkbTx7CIUUK2Uw1XDqHHs2hyZW1NzQiGI2m
Wwuxdghd1cHJnf1liVGmfYGZte7N7sCDwJJOXO5ci4zz
zHs sNxtF\/8DQMAJhv3H6Jp9\/k8U7NbfXEKV7o
jZGOB9B44w4yH30ccx2tGOzZikyGlamX9rOxIdVjHfk8
Tj\/9eWSYpIZBH1Ngp1RAKYEXJcqjbFdcRLEjhTRz9qZ
Olg7e0MLa2pyS0goQB5pS4WdUkOWQneaKrLzYxs
zFFrYow283qrC85oK67D1VYDGp6mjGeFHiab8k1oSJf2
Xpuw8

QQ6L\/2emtIF3HeA08HV1wI1pmM\/dfMzzt8uhjktNTX
TtyeQeeQE5qkpA939yDZmcKmQwon5RpCqWJhmOZbO44j
mzFwBp3\/6aFlNsZBt0RTSmozFEk0ZoQsWQ1G0pCxkV5
qpl5pOT74HeOYjwFNHGQW
YOxJEG6yiFK1H7eMyNBVCWmLvKwXJaTIliSCAh
BbMXBDwAHSMzB9wEbd1DuXvdBHTGF7ccqvLeMXwiF7lX
4L09rsUCK4pIcx1GzllhIW1TXfZ8A3vwh4C08bh5g685
lP9QnXdKCVazCe8s4m7KQtoxwWlsskHKDQrFWF
SmyDonOF2wISoAfQxkulhaYxaad
xsCFHDCtC4b3pIWTC0SvrswkwUlDW0eeYGmx4AdlBDhO
O9JjQePvI9jA59EGk Fr
hdaxCAhAv83TJEQ2tYLw479u\/xYnNjKZorURTp254go
x\/9XXg8EMkzpX5oaBhjD7EW
QaNDf8cnVWcr1d88tjwCqC3k8JhhS9tNKDSsye
JFiMylqg6x4kELoMfUw1vnBJ4Fv01Op6Hx3H30l4w6\/
fO9Wejeq6vdpp7zygzTkVlurhKbQOBXcnOs\/vcpMnBA
RIVV
8TvAyz8CXvouI6V\/BbsIE1pyim1hZHwPgwYVne\/pB
7eWCoXKXrRs4XXvPKhux3tigGNf8YNyA0p0
H2Lz7USp4VX30VuP5vpqXMdaZvhzfk9YkWIkvFDy0SB3
0
JiEW9nZDyqxISZoVE6dwynz4KeD9PwQeehIYfDsH5F73
I6gDEgW5RH026PM1uk\/xb95Z6w\/ZrGWB2gnqtwajRQ
4\/tHaoLQp
5CgLkksWkxjLsYnV5JLPvuH0tRptqcolf2uInS1g
G3fwOjeo0gzASpxyZO09rYTljxNETtIr1yP1IbeS
QckZJCXq\/GJVvo9kGaNkNKlKUCP6oi5fIr7PgchhsGM
NrYjfQtHylaY lm
PXZdy5qgPZcPP0yQ81x4Avvcua4lf\/sJOtktvTldy\/
KtfXgl38BztL1cyBxSRmgzTbTR3FKXbHtML3IW5k5dzs
eyQ89eblfeRF5FhWd7 NxO
\/Z5ZMfoJfZRm\/jlev8sFwyR1YDxIUhpS4r9VbtXv
boymXeRwfdVsMQJg3WUbvI4gLh5Qqpk1kaEpMMe1
9lPA848CP38YeO2PwXGKENQHyYLmdDXzvEqIi\/qRUqB
9aWkHvsKA7bE\/AV88Cxw6yqfsXvdC7StN90NaEvT2Sr
arDsTIXi9oSmJrKJWgdtSyH9oW8NeLwMd vFg
\/hPgd5x254dL5c\/92TG4CcKuPzbZhFCGe

```
miZMJ47hGWY8AFeo2glTfZh356sBPvBb5Ej6JynFqlMF
87\/vxyhfOyKwlD77gWpo W
uvJCXr3AX172RAHHtSQpsm9JGAPc\/eh7U7RubzMjk2l
8sP3JGJU\/bAbkAwp2rtRV015x2PAkceZxO2nVrgyP6r
1PglD41fiLRhSzEo21aYuxORpFGV0GbyZfCLMfgXJJQs
zqIGfD6u8MpTzdDQ754YUWVwZmESRpvGYuAV8jer
TWrI1xlYvfIL6qh73QuNJWiBSD0N0go9vbkADVLkFYMY
qwzt7muOheVIbeoz72QMTZUROcyfYZhuOsrcZ7IFo7lM
4Hsfs5jRRaNqB6WDWTjliXYRL5FP8j92u1Mv4NwOW7me
LqdctcuRMrD7
PyEBQq0tdu8uUwazR10b4xXsp3OUkJYG3KBSgonSKiK9
oBIlqXGlcgpkwHQ6phfbtipDhr3Rq18u1ggRaGCSK7Zr
gT16e\/PsjwDXGcsEjZNpQlSU2
RLEwuhMmrhGx8TxApgjbkxl5bMR3izbN8eqaDnqpPPw
c RXV 7
Fl3wszmt0sV4yrVCWtJKpVOXLUpIMUOLQ4zeDZw8A7zE
KHRsivPa8y7y2K Bh3\/LbPZByl2ZiwYSMjZ9B
evXcalW1eXkrkMkJZoedeLkh6ImNi2xRg93txIMz7GnO
cfn1sspxjRXmVEGzBmEaMmdUwG0QagcTbTbHhfrgslve
inPVSVUbSlIU8DmKOJz9HD5JgR3z8IPEgv1Mqbi7xW5F
QyRxbMUgsdN0qXh9QEtUllcgrtxTQOdvZie5ZG2choNO
OWKZZptVceGp
0RPv4\/VhwyV7UtJNJlAYwWmknE7Lav1J6X7Wo204mYX
3PWwC0vHGOWbu0JaanW7VQkNxHM9HL2RqEUMumOEhWWf
lY0k9yJaFMYAO1NIwQoay530xPpE1ysb3RKoMeroaib3
WUQ1lSBO0e1DpD3Vf86wwRomnzpj5XUAYVSRH29jt2Za
1qjDWsgyQkyheiIpEi7BMxPK41jbGE6Ps
CuejINT7hGH9m2EhWP8OYQhW
7dNtQFqN6dMHMQmRVj\/XnIZrH
DPQTq1PpvHYRAta3\/KkYZmN9PITlaU7YZt
HFHnUaQJTpkNsrHdRDEaH7le5rf8 a
v2UMKz\/0k4VsFNBHkyaIC9mX
WsDID\/AZreGI9CxIpzAAAAAElFTkSuQmCC\n"
                                        ]
                                    }
                            }
                    ]
```

```
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.product.delete

```
catalog.product.delete(id)
```

Метод удаляет товар торгового каталога.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер товара торгового каталога.

Примеры

```
BX24.callMethod(
    'catalog.product.delete',
    {
      id: 14
    },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.product.download

```
catalog.product.download(fileld, productId,
fieldName)
```

Метод скачивания файлов товара торгового каталога по переданным параметрам.

Если проверка успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
fileId	string	Номер зарегистрированного файла.
productId	string	Номер товара торгового каталога.
fieldName	string	Имя поля (свойства или поля элемента информационного блока) в котором хранится файл.

Примеры

```
BX24.callMethod(
    'catalog.product.download',
        {
    fields:{
                fileId: 79,
                productId: 8,
                fieldName: 'property98'
            }
    },
    function(result) {
        if (result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
    });
```

catalog.product.get

```
catalog.product.get(id)
```

Метод для доступа к значению полей товара торгового каталога по ID.

Если операция успешна, возвращается ресурс товара торгового каталога в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер товара торгового каталога.

Примеры

```
BX24.callMethod(
    'catalog.product.get',
    {
      id: 13
    },
    function(result)
    {
```

```
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.product.getFieldsByFilt

Описание, параметры, пример

```
catalog.product.getFieldsByFilter(filter)
```

Метод возвращает поля товара торгового каталога по фильтру.

Параметры

Параметр	Тип	Описание
filter	object	В фильтр передаются обязательных поля: ▪ **iblockId** - тип информационного блока; ▪ **productType** - тип продукта. ▪ **iblockSectionId** - идентификатор раздела. ▪ **name** - название.

Примеры

```
BX24.callMethod(
    'catalog.product.getFieldsByFilter',
        {
        filter: {
                            iblockId: 13,
                            productType: 1
        }
    },
    function(result) {
        if (result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
    });
```

Возвращаемые поля

Поле	Описание	Тип	Примечан
active	Активность	char	
available	Доступность	char	Только для чтения
bundle	Набор	char	
canBuyZero	Покупка при отсутствии	char	
code	Символьный код	string	
createdBy	Кем создан (id)	integer	
dateActiveFrom	Начало активности	datetime	

dateActiveTo	Окончание активности	datetime	
dateCreate	Дата создания	datetime	
detailPicture	Детальная картинка	file	
detailText	Детальное описание	string	
detailTextType	Тип детального описания	string	
height	Высота	double	
iblockId	Идентификатор информационного блока	integer	Обязательное поле.
iblockSectionId	Идентификатора раздела информационного блока	integer	Обязательное поле.
id	Идентификатор товара	integer	Только для чтения.
length	Длина	double	
measure	Единица измерения	integer	
modifiedBy	Кем изменен	integer	
name	Название	string	Обязательное поле.
negativeAmountTrace	Разрешение отрицательного количества товара	char	Только для чтения.
previewPicture	Картинка для анонса	file	

previewText	Описание для анонса	string	
previewTextType	Тип описания для анонса	string	
priceType	Тип цены	char	
propertyN	Свойства товара	array	Зависит от ти свойства. Мас включает в се value (описан свойства в зависимости типа) и value (идентификат свойства).
purchasingCurrency	Валюта	string	
purchasingPrice	Цена	string	
quantity	Доступное количество	double	
quantityReserved	Зарезервированное количество	double	
quantityTrace	Включить количественный учет	char	
sort	Сортировка	integer	
subscribe	Подписка на товар	char	
timestampX	Дата изменения	datetime	
type	Тип	integer	Только для чтения.
vatId	Идентификатор НДС	integer	

vatIncluded	НДС включен в цену	char	
weight	Вес	double	
width	Ширина	double	
xmlId	Внешний код	string	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.product.list

```
catalog.product.list(select, filter, order,
start)
```

Метод получает список товаров торгового каталога по фильтру.

Если операция успешна, возвращается список товаров в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей getFieldsByFilter.
filter	object	Поля, соответствующие доступному списку полей getFieldsByFilter.
order	object	Поля, соответствующие доступному списку полей getFieldsByFilter.
start	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'catalog.product.list',
    { select:{
                id
          } ,
          filter:{
                '>property109':{
                        value: '2019-03-
01T16:10:00+02:00'
                }
          },
          order:{
                id: ASC
          },
          start: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.product.update

```
catalog.product.update(id, fields)
```

Метод для обновления полей товара торгового каталога.

Если операция успешна, возвращается ресурс товара торгового каталога в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	номер товара торгового каталога.
fields	object	Поля, соответствующие доступному списку полей getFieldsByFilter .

Примеры

```
BX24.callMethod(
    'catalog.product.update',
    {
          id: 13,
```

```
            fields: {
                  active: N,
                  createdBy: 1,
                  iblockId: 13,
                  sort: 340
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс коэффициента единицы измерения

Коэффициент единицы измерения:

```
{
        "id": 2,
        "isDefault": "Y",
        "productId": 2,
        "ratio": 1
}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.ratio.get

```
catalog.ratio.get(id)
```

Метод для доступа к значению полей коэффициента единицы измерения по ID.

Если операция успешна, возвращается ресурс коэффициента единицы измерения в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер коэффициента единицы измерения.

Примеры

```
BX24.callMethod(
    'catalog.ratio.get',
    {
      id: 7
    },
```

```
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.ratio.getFields

```
catalog.ratio.getFields()
```

Метод возвращает поля коэффициента единицы измерения.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.ratio.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание	Тип	Примечание
id	Идентификатор единицы измерения	integer	Только для чтения.
isDefault	Единица измерения по умолчанию	char	
productId	Идентификатор товара	integer	Обязательное поле.
ratio	Коэффициент единицы измерения	double	Обязательное поле.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.ratio.list

```
catalog.ratio.list(select, filter, order,
start)
```

Метод получает список коэффициентов единиц измерения по фильтру.

Если операция успешна, возвращается список коэффициентов единиц измерения в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
start	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'catalog.ratio.list',
    { select:{
                id
          } ,
          filter:{
                isDefault: Y
          },
          order:{
                id: asc
          },
          start: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс правила округления цен

Правило округления цен:


```
{
        "catalogGroupId": 1,
        "createdBy": 1,
        "dateCreate": "2019-06-
28T10:38:32+03:00",
        "dateModify": "2019-06-
28T10:38:32+03:00",
        "id": 1,
        "modifiedBy": 1,
        "price": 1000,
        "roundPrecision": 1,
        "roundType": 1
}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.roundingRule.get

```
catalog.roundingRule.get(id)
```

Метод для доступа к значению полей правила округления цен по ID.

Если операция успешна, возвращается ресурс правила округления цен в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер правила округления цен.

Примеры

```
BX24.callMethod(
    'catalog.roundingRule.get',
    {
      id: 7
    },
    function(result)
```

```
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.roundingRule.getFields

```
catalog.roundingRule.getFields()
```

Метод возвращает поля правила округления цен.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.roundingRule.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание	Тип	Примечание
catalogGroupId	Тип цены	integer	Обязательное поле.
createdBy	Кем создано	integer	Обязательное поле.
dateCreate	Дата создания	datetime	Обязательное поле.
dateModify	Дата изменения	datetime	Обязательное поле.
id	Идентификатор правила округления цен	integer	Только для чтения.
modifiedBy	Кем изменено	integer	Обязательное поле.
price	Минимальная цена	double	Обязательное поле.
roundPrecision	Точность округления	double	Обязательное поле.
roundType	Тип округления	integer	Обязательное поле.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.roundingRule.list

```
catalog.roundingRule.list(select, filter,
order, start)
```

Метод получает список правил округления цен по фильтру.

Если операция успешна, возвращается список правил округления в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
start	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'catalog.roundingrule.list',
    { select:{
                id
          } ,
          filter:{
                modifiedBy: 1
          },
          order:{
                id: ASC
          },
          start: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс секции

Секция торгового каталога:


```
{
    "active": "Y",
    "code": null,
    "description": null,
    "descriptionType": "text",
    "iblockId": 1,
    "iblockSectionId": 1,
    "id": 2,
    "name": "Бухгалтерия",
    "sort": 500,
    "xmlId": null
}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.section.add

```
catalog.section.add(fields)
```

Метод добавляет секцию в коллекцию секций торговых каталогов.

Если операция успешна, возвращается ресурс секции торгового каталога в теле ответа.

Параметры

Параметр	Тип	Описание
fields	object	Поля, соответствующие доступному списку полей fields.

Примеры

```
BX24.callMethod(
    'catalog.section.add',
        {
        fields: {
                                iblockId: 5,
                                iblockSectionId:
```

```
3,
                                name: 'Новости',
                                code: 'news'
        }
    },
    function(result) {
        if (result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
    });
```

catalog.section.delete

```
catalog.section.delete(id)
```

Метод удаляет секцию каталога из коллекции секций торгового каталога.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	номер секции каталога.

Примеры

```
BX24.callMethod(
    'catalog.section.delete',
    {
      id: 14
    },
    function(result)
    {
```

```
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.section.get

```
catalog.section.get(id)
```

Метод для доступа к значению полей секции торгового каталога по ID.

Если операция успешна, возвращается ресурс секции торгового каталога в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	номер секции торгового каталога.

Примеры

```
BX24.callMethod(
    'catalog.section.get',
    {
      id: 13
    },
    function(result)
```

```
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.section.getFields

```
catalog.section.getFields()
```

Метод возвращает поля секции торгового каталога.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.section.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание	Тип	Примечание
active	Активность	char	
code	Символьный код	string	Обязательное поле.
description	Описание	string	
descriptionType	Тип описания	string	
iblockId	Идентификатор инфоблока	integer	Обязательное поле.
iblockSectionId	Идентификатор родительского раздела	integer	
id	Идентификатор раздела	integer	Только для чтения
name	Название	string	Обязательное поле.
sort	Сортировка	integer	
xmlId	Внешний код	string	

catalog.section.list

```
catalog.section.list(select, filter, order,
start)
```

Метод получает список секций торговых каталогов по фильтру.

Если операция успешна, возвращается список разделов каталога в теле ответа. При получении списка необходимо анализировать такие поля, как *total* и *next*. которые укажут на наличие элементов, которые еще не попали в выборку.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
start	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'catalog.section.list',
    {
      select:{
                id
          },
          filter:{
                name: Битрикс
          },
          order:{
                id: ASC
          },
     start: 1
    },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.section.update

```
catalog.section.update(Id, fields)
```

Метод для обновления полей секции каталога.

Если операция успешна, возвращается ресурс секции торгового каталога в теле ответа.

Параметры

Параметр	Тип	Описание
Id	string	номер секции торгового каталога.
fields	object	Поля, соответствующие доступному списку полей fields .

Примеры

```
BX24.callMethod(
    'catalog.section.update',
    {
          id: 13,
```

```
            fields: {
                  yandexExport: Y
            }
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Ресурс склада Склад:


```
{
        "active": "Y",
        "address": "пр. Московский д. 51",
        "code": null,
        "dateCreate": "2018-10-
22T11:30:56+03:00",
        "dateModify": "2018-10-
22T11:30:56+03:00",
        "description": "Здесь вы найдёте
товары ведущих производителей мира",
        "email": null,
        "gpsN": 54.71411,
        "gpsS": 20.56675,
        "id": 1,
        "imageId": 12,
        "issuingCenter": "Y",
        "locationId": null,
        "modifiedBy": null,
        "phone": "8 (495) 212 85 06",
        "schedule": "Пн.-Пт. с 9:00 до
20:00, Сб.-Вс. с 11:00 до 18:00",
        "shippingCenter": "Y",
        "siteId": null,
        "sort": 100,
        "title": "Склад",
        "userId": null,
        "xmlId": null
}
```

catalog.store.add

```
catalog.store.add(fields)
```

Метод для добавления склада.

Если операция успешна, возвращается id добавленного склада.

Параметры

Параметр	Тип	Описание
fields	array	Параметры добавляемого склада.

Примеры

```
BX24.callMethod(
    'catalog.store.add',
    {
        fields: {
                        'title': 'Склад 1',
                        'sort': '100',
            'active': 'Y',
                        'issuingCenter':
```

```
'Y',
                        'shippingCenter':
'Y',
                        'code': 'store_1',
            'address': 'пр. Московский д.
52',
            'phone': '+0 123 456 789',
            'schedule': 'Пн.-Пт. с 9:00 до
20:00, Сб.-Вс. с 11:00 до 18:00',
            'xmlId': 'store_1',
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.store.add',
    [
       'fields' => [
            'title' => 'Склад 1',
            'sort' => '100',
            'active' => 'Y',
            'issuingCenter' => 'Y',
            'shippingCenter' => 'Y',
            'code' => 'store_1',
            'address' => 'пр. Московский д.
52',
            'phone' => '+0 123 456 789',
            'schedule' => 'Пн.-Пт. с 9:00 до
```

```
20:00, Сб.-Вс. с 11:00 до 18:00',
            'xmlId' => 'store_1',
        ],
    ]
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

catalog.store.delete

```
catalog.store.delete(id)
```

Метод для удаления склада.

Если операция успешна, возвращается `Y` в теле ответа.

Параметры

Параметр	Тип	Описание
id	integer	Идентификатор склада.

Примеры

```
BX24.callMethod(
    'catalog.store.delete',
    {
        id: 42,
    },
    function(result)
    {
         if(result.error())
```

```
            console.error(result.error());
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.store.delete',
    [
        'id' => 42,
    ]
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

catalog.store.get

```
catalog.store.get(id)
```

Метод для доступа к значениям полей склада по ID.

Если операция успешна, возвращается ресурс склада в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер налога.

Примеры

```
BX24.callMethod(
    'catalog.store.get',
    {
      id: 7
    },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.store.getFields

```
catalog.store.getFields()
```

Метод возвращает поля складов.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.store.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание	Тип	Примечание
active	Активность	char	
address	Адрес	string	Обязательное поле
code	Символьный код	string	
dateCreate	Дата создания	datetime	
dateModify	Дата изменения	datetime	
description	Описание	string	
email	Email	string	
gpsN	GPS широта	double	
gpsS	GPS долгота	double	
id	Идентификатор склада	integer	Только для чтения.
imageId	Идентификатор изображения	integer	
issuingCenter	Пункт выдачи	char	
locationId	Идентификатор местоположения	integer	
modifiedBy	Кем изменен	integer	
phone	Телефон	string	
schedule	График работы	string	
shippingCenter	Для отгрузки	char	

siteId	Идентификатор сайта	string	
sort	Сортировка	integer	
title	Название	string	
userId	Кем создан	integer	
xmlId	Внешний код	string	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.store.list

```
catalog.store.list(select, filter, order,
start)
```

Метод получает список складов по фильтру.

Если операция успешна, возвращается список складов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
start	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'catalog.store.list',
    { select:{
                id
          } ,
          filter:{
                modifiedBy: 1
          },
          order:{
                id: ASC
          },
          start: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.store.update

```
catalog.store.update(id, fields)
```

Метод для обновления склада.

Если операция успешна, возвращается id обновлённого склада.

Параметры

Параметр	Тип	Описание
id	string	Идентификатор склада.
fields	array	Параметры обновляемого склада.

Примеры

```
BX24.callMethod(
    'catalog.store.update',
    {
                id: 42,
                fields: {
                        'title': 'Склад 1',
```

```
                            'sort': '100',
            'active': 'Y',
                            'issuingCenter':
'Y',
                            'shippingCenter':
'Y',
                            'code': 'store_1',
            'address': 'пр. Московский д.
52',
            'phone': '+0 123 456 789',
            'schedule': 'Пн.-Пт. с 9:00 до
20:00, Сб.-Вс. с 11:00 до 18:00',
            'xmlId': 'store_1',
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.store.update',
    [
        'id' => 42,
        'fields' => [
            'title' => 'Склад 1',
            'sort' => '100',
            'active' => 'Y',
            'issuingCenter' => 'Y',
            'shippingCenter' => 'Y',
            'code' => 'store_1',
```

```
            'address' => 'пр. Московский д.
52',
            'phone' => '+0 123 456 789',
            'schedule' => 'Пн.-Пт. с 9:00 до
20:00, Сб.-Вс. с 11:00 до 18:00',
            'xmlId' => 'store_1',
        ],
    ]
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Складской учёт::catalog.document.add

```
catalog.document.add(fields)
```

Метод для добавления документа складского учёта.

Если операция успешна, возвращается id добавленного документа.

Параметры

Параметр	Тип	Описание
fields	array	Параметры добавляемого документа.

Примеры

```
BX24.callMethod(
    'catalog.document.add',
    {
        'fields': {
            'DOC_TYPE': 'S',
//\CCatalogDocsTypes::TYPE_STORE_ADJUSTMENT,
```

```
тип - оприходывание
            'CONTRACTOR_ID': '1', //
поставщик
            'RESPONSIBLE_ID': '1', // ID
ответственного,
            'DATE_MODIFY': '2000-01-
01T00:00:00+02:00',
            'DATE_CREATE': '2000-01-
01T00:00:00+02:00',
            'CREATED_BY': '1',
            'MODIFIED_BY': '1',
            'CURRENCY': 'USD', //
используемая валюта,
            'STATUS': 'S',
            'DATE_STATUS': '2000-01-
01T00:00:00+02:00',
            'DATE_DOCUMENT': '2000-01-
01T00:00:00+02:00', //DATE_ATOM
            'STATUS_BY': '1',
            'TOTAL': '100',
            'COMMENTARY': 'first document.',
//комментарий
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.document.add',
```

```
    [
        'fields' => [
            'DOC_TYPE' => 'S',
            'CONTRACTOR_ID' => '1',
            'RESPONSIBLE_ID' => '1',
            'DATE_MODIFY' => '2000-01-
01T00:00:00+02:00',
            'DATE_CREATE' => '2000-01-
01T00:00:00+02:00',
            'CREATED_BY' => '1',
            'MODIFIED_BY' => '1',
            'CURRENCY' => 'USD',
            'STATUS' => 'S',
            'DATE_STATUS' => '2000-01-
01T00:00:00+02:00',
            'DATE_DOCUMENT' => '2000-01-
01T00:00:00+02:00',
            'STATUS_BY' => '1',
            'TOTAL' => '100',
            'COMMENTARY' => 'first
document.',
        ],
    ]
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Складской учёт::catalog.document.confirr

```
catalog.document.confirm(id)
```

Метод для проведения документа.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	integer	Идентификатор документа.

Примеры

```
BX24.callMethod(
    'catalog.document.confirm',
    {
        'id': 42,
    },
    function(result)
```

```
    {
        if(result.error())
            console.error(result.error());
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.document.confirm',
    [
        'id' => 42,
    ]
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Складской учёт::catalog.document.delete

```
catalog.document.delete(id)
```

Метод для удаления документа.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	integer	Идентификатор документа.

Примеры

```
BX24.callMethod(
    'catalog.document.delete',
    {
        'id': 42,
    },
    function(result)
```

```
    {
        if(result.error())
            console.error(result.error());
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.document.delete',
    [
        'id' => 42,
    ]
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Складской учёт::catalog.document.fields

Описание и пример

```
catalog.document.fields()
```

Метод возвращает список полей документов.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.document.fields',
    {},
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.log(result.data());
```

```
    }
);
```

```
$result = CRest::call(
    'catalog.document.fields'
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Возвращаемые поля

Поле	Описание	Тип	Примечание
ID	Идентификатор	integer	
DOC_TYPE	Тип документа	char	
SITE_ID	Привязка к сайту	char	
CONTRACTOR_ID	Поставщик	integer	
RESPONSIBLE_ID	Ответственный	integer	Обязательное
DATE_MODIFY	Дата изменений	datetime	
DATE_CREATE	Дата создания	datetime	
CREATED_BY	Создатель	integer	
MODIFIED_BY	Кем изменён	integer	
CURRENCY	Валюта	char	Обязательное

STATUS	Статус	char	
DATE_STATUS	Дата установки статуса	datetime	
DATE_DOCUMENT	Дата документа	datetime	
STATUS_BY	Кем установлен статус	integer	
TOTAL	Общая сумма товаров	double	
COMMENTARY	Комментарий	char	

© «Битрикс», 2001-2008, «1С-

Складской учёт::catalog.document.list

```
catalog.document.list(order, filter, select,
offset, limit)
```

Метод для получения списка документов.

Если операция успешна, возвращается список документов в теле ответа.

Параметры

Параметр	Тип	Описание
order	array	Сортировка.
filter	array	Фильтр.
select	array	Список получаемых полей.
offset	integer	Страница выборки, работает аналогично start.
limit	integer	Размер страницы от 1 до 500 (если указан 0 или число больше максимального, то используется значение по умолчанию 50).

Примеры

```
BX24.callMethod(
    'catalog.document.list',
    {
        'order': {
            'ID': 'ASC',
        },
        'filter': {
          '>ID': 0,
        },
        'offset': 0,
        'limit': 50,
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.document.list',
    [
        'order' => [
            'ID' => 'ASC',
        ],
        'filter' => [
            '>ID' => 0,
        ],
        'offset' => 0,
        'limit' => 5,
```

```
    ]
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Складской учёт::catalog.document.mode.s

```
catalog.document.mode.status()
```

Метод для получения информации о том, включен ли складской учёт.

Возвращается статус складского учёта:

- `Y` - складской учёт включен;
- `N` - складской учёт выключен.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.document.mode.status',
    {},
    function(result)
    {
        if(result.error())
```

```
            console.error(result.error());
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.document.mode.status'
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Складской учёт::catalog.document.unconf

```
catalog.document.unconfirm(id)
```

Метод для отмены проведения документа.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	integer	Номер документа.

Примеры

```
BX24.callMethod(
    'catalog.document.unconfirm',
    {
        'id': 42,
    },
    function(result)
```

```
    {
        if(result.error())
            console.error(result.error());
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.document.unconfirm',
    [
        'id' => 42,
    ]
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Складской учёт::catalog.document.update

```
catalog.document.update(id, fields)
```

Метод для обновления документа складского учёта.

Если операция успешна, возвращается `true`.

Параметры

Параметр	Тип	Описание
id	integer	Идентификатор документа.
fields	array	Параметры документа.

Примеры

```
BX24.callMethod(
    'catalog.document.update',
    {
        'id': 42,
```

```
        'fields': {
            'TOTAL': '1000', // общая сумма
всех PURCHASING_PRICE умноженных на AMOUNT
            'COMMENTARY': 'first document.',
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.document.update',
    [
        'id' => 42,
        'fields' => [
            'TOTAL' => '1000',
            'COMMENTARY' => 'first
document.',
        ],
    ]
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Товары документа складского учёта::catalog.document.elem

```
catalog.document.element.add(fields)
```

Метод для добавления товара документа складского учёта.

Если операция успешна, возвращается id добавленного товара.

Параметры

Параметр	Тип	Описание
fields	array	Параметры добавляемого товара.

Примеры

```
BX24.callMethod(
    'catalog.document.element.add',
    {
        'fields': {
            'DOC_ID': 1,
```

```
            'STORE_FROM': 0,
            'STORE_TO': 1,
            'ELEMENT_ID': 42,
            'AMOUNT': 10,
            'PURCHASING_PRICE': 25,
        }
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.document.element.add',
    [
        'fields' => [
            'DOC_ID' => 1,
            'STORE_FROM' => 0,
            'STORE_TO' => 1,
            'ELEMENT_ID' => 42,
            'AMOUNT' => 10,
            'PURCHASING_PRICE' => 25,
        ],
    ]
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

© «Битрикс», 2001-2008, «1С-

Товары документа складского учёта::catalog.document.elem

```
catalog.document.element.deleted(id)
```

Метод для удаления товара документа складского учёта.

Если операция успешна, возвращается `true` в теле ответа.

Параметры

Параметр	Тип	Описание
id	integer	Идентификатор документа.

Примеры

```
BX24.callMethod(
    'catalog.document.element.delete',
    {
        'id': 42,
```

```
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.document.element.delete',
    [
        'id' => 42,
    ]
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Товары документа складского учёта::catalog.document.elem

Описание и пример

```
catalog.document.element.fields()
```

Метод возвращает список полей товаров документа складского учёта.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.document.element.fields',
    {},
    function(result)
    {
        if(result.error())
            console.error(result.error());
```

```
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.document.element.fields'
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Возвращаемые поля

Поле	Описание	Тип	Примечан
ID	Идентификатор	integer	
DOC_ID	Идентификатор документа	integer	Обязательное
STORE_FROM	Склад-источник	integer	
STORE_TO	Склад	integer	Обязательное
ELEMENT_ID	Идентификатор товара (catalog.product.list)	integer	Обязательное
AMOUNT	Количество	double	Обязательное
PURCHASING_PRICE	Закупочная цена	double	Обязательное

© «Битрикс», 2001-2008, «1С-

Товары документа складского учёта::catalog.document.elem

```
catalog.document.element.list(order, filter,
select, offset, limit)
```

Метод для получения списка товаров в документах складского учёта.

Если операция успешна, возвращается список документов в теле ответа.

Параметры

Параметр	Тип	Описание
order	array	Сортировка.
filter	array	Фильтр.
select	array	Список получаемых полей.
offset	integer	Страница выборки, работает аналогично start.
limit	integer	Размер страницы от 1 до 500 (если указан 0 или число больше

		максимального, то используется значение по умолчанию 50).

Примеры

```
BX24.callMethod(
    'catalog.document.element.list',
    {
        'order': {
            'ID': 'ASC',
        },
        'filter': {
          '>ID': 0,
        },
        'offset': 0,
        'limit': 50,
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
            console.log(result.data());
    }
);
```

```
$result = CRest::call(
    'catalog.document.element.list',
    [
        'order' => [
            'ID' => 'ASC',
        ],
```

```
        'filter' => [
            '>ID' => 0,
        ],
        'offset' => 0,
        'limit' => 5,
    ]
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Товары документа складского учёта::catalog.document.elem

```
catalog.document.element.update(id, fields)
```

Метод для обновления товара документа складского учёта.

Если операция успешна, возвращается `true`.

Параметры

Параметр	Тип	Описание
id	integer	Идентификатор обновляемого товара.
fields	array	Параметры обновляемого товара.

Примеры

```
BX24.callMethod(
        'catalog.document.element.update',
        {
                'fields': {
                        'AMOUNT': 10,
                        'PURCHASING_PRICE':
25,
                }
        },
        function(result)
        {
                if(result.error())

console.error(result.error());
                else

console.log(result.data());
        }
);
```

```
$result = CRest::call(
    'catalog.document.element.update',
    [
        'id' => 11,
        'fields' => [
            'AMOUNT' => 10,
            'PURCHASING_PRICE' => 25,
        ],
    ]
);

echo '<pre>';
print_r($result);
echo '</pre>';
```

Ресурс налога Налог:


```
{
        "active": "Y",
        "id": 1,
        "name": "Без НДС",
        "rate": 0,
        "sort": 100,
        "timestampX": "2018-10-
22T11:30:43+03:00"
}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

catalog.vat.get

```
catalog.vat.get(id)
```

Метод для доступа к значениям полей налога по ID.

Если операция успешна, возвращается ресурс налога в теле ответа.

Параметры

Параметр	Тип	Описание
id	string	Номер налога.

Примеры

```
BX24.callMethod(
    'catalog.vat.get',
    {
      id: 7
    },
    function(result)
    {
        if(result.error())
```

```
console.error(result.error().ex);
        else
            console.log(result.data());
});
```

catalog.vat.getFields

Метод возвращает поля налога.

Параметры

Без параметров.

Примеры

```
BX24.callMethod(
    'catalog.vat.getFields',
    {},
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

Возвращаемые поля

Поле	Описание	Тип	Примечание

active	Активность	char	
id	Идентификатор	integer	Только для чтения.
name	Название	string	Обязательное поле.
rate	Ставка	double	Обязательное поле.
sort	Сортировка	integer	
timestampX	Дата изменения	datetime	

catalog.vat.list

```
catalog.vat.list(select, filter, order,
start)
```

Метод получает список налогов по фильтру.

Если операция успешна, возвращается список налогов в теле ответа.

Параметры

Параметр	Тип	Описание
select	object	Поля, соответствующие доступному списку полей fields.
filter	object	Поля, соответствующие доступному списку полей fields.
order	object	Поля, соответствующие доступному списку полей fields.
start	string	Номер страницы вывода.

Примеры

```
BX24.callMethod(
    'catalog.vat.list',
    { select:{
                id
          } ,
          filter:{
                modifiedBy: 1
          },
          order:{
                id: ASC
          },
          start: 1
        },
    function(result)
    {
        if(result.error())

console.error(result.error().ex);
        else
            console.log(result.data());
});
```

lists.add

Описание

```
lists.add (params)
```

Метод создаёт список. В случае успешного создания списка ответ *true*, иначе *Exception*.

Параметры

Параметр	Описание
IBLOCK_TYPE_ID	id типа инфоблока (обязательное): ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп
IBLOCK_CODE	код инфоблока (обязательное);
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);
FIELDS	поля инфоблока: ▪ **NAME** - название инфоблока (обязательно);

	▪ **DESCRIPTION** - описание; ▪ **SORT** - сортировка; ▪ **PICTURE** - изображение; ▪ **BIZPROC** - включение поддержки бизнес-процессов.
MESSAGES	подписи к элементам и разделам списка;
RIGHTS	управление правами доступа (если не заполнены, то авторизованному пользователю, который работает с rest устанавливается полный доступ на созданный инфоблок).

Пример

```
    var params = {
        'IBLOCK_TYPE_ID': 'lists_socnet',
        'IBLOCK_CODE': 'rest_1',
        'SOCNET_GROUP_ID': '4',
        'FIELDS': {
            'NAME': 'List 1',
            'DESCRIPTION': 'Test list',
            'SORT': '10',
            'PICTURE':
document.getElementById('iblock-image-add'),
            'BIZPROC': 'Y'
        },
        'MESSAGES': {
            'ELEMENT_NAME': 'Element',
            'ELEMENTS_NAME': 'Elements',
            'ELEMENT_ADD': 'Add element',
            'ELEMENT_EDIT': 'Edit element',
            'ELEMENT_DELETE': 'Delete
element',
            'SECTION_NAME': 'Section',
```

```
            'SECTIONS_NAME': 'Sections',
            'SECTION_ADD': 'Add section',
            'SECTION_EDIT': 'Edit section',
            'SECTION_DELETE': 'Delete
section'
        },
        'RIGHTS': {
            'SG4_A': 'W',
            'SG4_E': 'W',
            'SG4_K': 'W',
            'AU': 'D',
            'G2': 'D'
        }
    };
    BX24.callMethod(
        'lists.add',
        params,
        function(result)
        {
            if(result.error())
                alert("Error: " +
result.error());
            else
                alert("Success: " +
result.data());
        }
    );
```

lists.delete

```
lists.delete (params)
```

Метод удаляет список. В случае успешного удаления ответ true, иначе false.

Параметры

Параметр	Описание
IBLOCK_TYPE_ID	id типа инфоблока (обязательное): ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп
IBLOCK_CODE/IBLOCK_ID	код или id инфоблока (обязательное);
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);

Пример

```
    var params = {
        'IBLOCK_TYPE_ID': 'lists_socnet',
        'IBLOCK_CODE': 'rest_1'
    };
    BX24.callMethod(
        'lists.delete',
        params,
        function(result)
        {
            if(result.error())
                alert("Error: " +
result.error());
            else
                alert("Success: " +
result.data());
        }
    );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

lists.get

Описание

```
lists.get (params)
```

Метод возвращает данные инфоблока. В случае успеха будут возвращены данные инфоблока, иначе пустой массив. С помощью метода можно получить список данных сразу всех инфоблоков указанного типа инфоблока. При работе с группами соц сети обязательно необходимо указывать ID группы, иначе произойдёт ошибка доступа.

Параметры

Параметр	Описание
IBLOCK_TYPE_ID	id типа инфоблока (обязательное): ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп
IBLOCK_CODE/IBLOCK_ID	код или id инфоблока (если не указан в ответе будут данные по всем спискам указанного типа инфоблока)

SOCNET_GROUP_ID	id группы, обязателен, если список находится в группах.
IBLOCK_ORDER	Сортировка. Массив полей разделов информационного блока. Направление сортировки: **asc** (по возрастания) или **desc** (по убыванию) Пример: `'IBLOCK_ORDER': { "ID": "DESC" }`

Пример

```
    var params = {
        'IBLOCK_TYPE_ID': 'lists',
        'IBLOCK_CODE': 'rest_1'
    };
    BX24.callMethod(
        'lists.get',
        params,
        function(result)
        {
            if(result.error())
                alert("Error: " +
result.error());
            else
                console.log(result.data());
        }
    );
```

lists.get.iblock.type.id

Метод возвращает id типа инфоблока. В случае успеха будет возвращена строка с id типа инфоблока, иначе *null*.

Параметры

Метод	Описание	С версии
IBLOCK_CODE/IBLOCK_ID	код или id инфоблока	

Пример:

```
var params = {
    'IBLOCK_ID': '41'
};
BX.rest.callMethod(
   'lists.get.iblock.type.id',
   params,
   function(result)
   {
       if(result.error())
           alert("Error: " +
result.error());
       else
```

```
                console.log(result.data());
    }
);
```

lists.update

Описание

```
lists.update (params, fields)
```

Метод обновляет существующий список. В случае успешного обновления списка ответ true, иначе Exception.

Параметры

Параметр	Описание
IBLOCK_TYPE_ID	id типа инфоблока (обязательное): ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп
IBLOCK_CODE/IBLOCK_ID	код или id инфоблока (обязательное);
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);
FIELDS	поля инфоблока:

	▪ **NAME** - название инфоблока (обязательно); ▪ **DESCRIPTION** - описание; ▪ **SORT** - сортировка; ▪ **PICTURE** - изображение; ▪ **BIZPROC** - включение поддержки бизнес-процессов.
MESSAGES	подписи к элементам и разделам списка;
RIGHTS	управление правами доступа.

Пример

```
    var params = {
        'IBLOCK_TYPE_ID': 'lists_socnet',
        'IBLOCK_CODE': 'rest_1',
        'FIELDS': {
            'NAME': 'List 1 (Update)',
            'DESCRIPTION': 'Test list
(Update)',
            'SORT': '20',
            'PICTURE':
document.getElementById('iblock-image-
update')
        },
        'RIGHTS': {
            'G1': 'X'
        }
    };
    BX24.callMethod(
        'lists.update',
        params,
        function(result)
        {
```

```
            if(result.error())
                alert("Error: " +
result.error());
            else
                alert("Success: " +
result.data());
        }
    );
```

lists.element.add

Описание

```
lists.element.add (params, fields)
```

Метод создаёт элемент списка. В случае успешного создания элемента ответ true, иначе Exception.

Чтобы загрузить файлы в поле типа Файл (Диск) необходимо:

1. использовать rest api модуля disk: disk.folder.uploadfile и disk.storage.uploadfile. В ответе при загрузке этих файлов, вы будете получать "FILE_ID": 290.
2. Получить список ID загруженных файлов.
3. Затем с помощью rest api модуля lists добавлять файлы в нужное поле:

```
var params = {
 'IBLOCK_TYPE_ID': 'lists',
 'IBLOCK_ID': '41',
 'ELEMENT_CODE': 'element1',
 'FIELDS': {
     'NAME': 'Test element 1',
     'PROPERTY_121': { 'n0':["n1582"]}
 }
};
BX.rest.callMethod(
 'lists.element.add',
```

```
 params,
 function(result)
 {
      if(result.error())
         alert("Error: " +
result.error());
     else
         alert("Success: " +
result.data());
 }
);
```

Параметры

Параметр	Описание
IBLOCK_TYPE_ID	id типа инфоблока (обязательно): ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп
IBLOCK_CODE/IBLOCK_ID	код или id инфоблока (обязательно)
ELEMENT_CODE	код элемента инфоблока (обязательно)
LIST_ELEMENT_URL	шаблон адреса к элементам списка
FIELDS	массив полей и значений
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);

Пример

```
var params = {
        'IBLOCK_TYPE_ID': 'lists_socnet',
        'IBLOCK_CODE': 'rest_1',
        'ELEMENT_CODE': 'element_1',
        'LIST_ELEMENT_URL':
'#list_id#/element/#section_id#/#element_id#
/',
        'FIELDS': {
            'NAME': 'Test element',
            'PROPERTY_62': 'Text string',
            'PROPERTY_63': {
                '0': '7',
                '1': '9',
                '2': '10'
            }
        }
    };
    BX24.callMethod(
        'lists.element.add',
        params,
        function(result)
        {
            if(result.error())
                alert("Error: " +
result.error());
            else
                alert("Success: " +
result.data());
        }
    );
```

Пример добавления файла:

```
var params = {
        'IBLOCK_TYPE_ID': 'lists',
        'IBLOCK_ID': '41',
        'ELEMENT_CODE': 'element1',
        'FIELDS': {
             'NAME': 'Test element 1',
             'PROPERTY_122':
document.getElementById('fileInputId') //
PROPERTY_122 - Пользовательское свойство
типа "Файл"
                }
        };
   BX.rest.callMethod(
        'lists.element.add',
        params,
        function(result)
        {
             if(result.error())
                alert("Error: " +
result.error());
            else
                alert("Success: " +
result.data());
        }
);
```

lists.element.delete

```
lists.element.delete (params)
```

Метод удаляет элемент списка. В случае успешного удаления элемента ответ true, иначе Exception.

Параметры

Параметр	Описание
IBLOCK_TYPE_ID	id типа инфоблока (обязательно): ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп
IBLOCK_CODE/IBLOCK_ID	код или id инфоблока (обязательно)
ELEMENT_CODE/ELEMENT_ID	код или id элемента
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);

Пример

```
var params = {
        'IBLOCK_TYPE_ID': 'lists_socnet',
        'IBLOCK_CODE': 'rest_1',
        'ELEMENT_CODE': 'element_1'
    };
    BX24.callMethod(
        'lists.element.delete',
        params,
        function(result)
        {
            if(result.error())
                alert("Error: " +
result.error());
            else
                alert("Success: " +
result.data());
        }
    );
```

lists.element.get

Описание

```
lists.element.get(params)
```

Метод возвращает список элементов или элемент. В случае успеха возвращает данные по элементу(там), иначе пустой массив.

Параметры

Параметр	Описание
IBLOCK_TYPE_ID	id типа инфоблока (обязательно): ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);
IBLOCK_CODE/IBLOCK_ID	код или id инфоблока (обязательно)

ELEMENT_CODE/ELEMENT_ID	код или id элемента (Если не указан, вернет список всех элементов списка)
ELEMENT_ORDER	Сортировка. Массив полей элементов информационного блока. Направление сортировки: **asc** (по возрастания) или **desc** (по убыванию) Пример: `'ELEMENT_ORDER': { "ID": "DESC" }` Не поддерживается сортировка всех множественных свойств, а так же свойств: S:Money, PREVIEW_TEXT, DETAIL_TEXT, S:ECrm, S:map_yandex, PREVIEW_PICTURE, DETAIL_PICTURE, S:DiskFile, IBLOCK_SECTION_ID, BIZPROC, COMMENTS.
FILTER	Фильтрация элементов. Объект с списком полей и значений. Для фильтрации доступны почти все поля из фильтра CIBlockElement::GetList. Например для фильтрации по полю число, нужно указать знак равно: `'FILTER': { '=ID': [120,121], }` Так же есть возможность использовать полнотекстовый поиск. Для этого Нужно использовать поле SEARCHABLE_CONTENT с префиксом "*";

Пример

```
var params = {
        'IBLOCK_TYPE_ID': 'lists_socnet',
        'IBLOCK_CODE': 'rest_1',
        'ELEMENT_CODE': 'element_1'
    };
    BX24.callMethod(
        'lists.element.get',
        params,
        function(result)
        {
            if(result.error())
                alert("Error: " +
result.error());
            else
                console.log(result.data());
        }
    );
```

```
var params = {
        'IBLOCK_TYPE_ID': 'lists',
        'IBLOCK_ID': '41',
        'FILTER': {
            '>=DATE_CREATE': '27.03.2018
00:00:00',
            '<=DATE_CREATE': '27.03.2018
23:59:59',
        }
};
```

Пример отключения постраничной навигации:

```
$result = CRest::call(
    'lists.element.get',
```

```
    [
        'IBLOCK_TYPE_ID' => 'lists',
        'IBLOCK_ID' => '42',
        'start' => -1
    ]
);
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

lists.element.get.file.url

Описание

```
lists.element.get.file.url (
   params
)
```

Метод возвращает путь к файлу. В случае успеха будет возвращен массив со списком url для нужного поля типа Файл или Файл (Диск).

Параметры

Метод	Описание	С версии

params	Все поля обязательные
	▪ IBLOCK_TYPE_ID - id типа инфоблока:
	  ▪ lists - тип инфоблока списка;
	  ▪ bitrix_processes - тип инфоблока процессов;
	  ▪ lists_socnet - тип инфоблока списков групп. В этом	

	случае обязательно передавать параметр SOCNET_GROUP_ID - id группы. ▪ IBLOCK_ID - id инфоблока ▪ ELEMENT_ID - id элемента ▪ FIELD_ID - id поля (свойства инфоблока) Файл или Файл (Диск), без префикса "PROPERTY_" ▪ SEF_FOLDER: '/my_section/lists/' - путь до папки, с которой работает компонент. Параметр не обязательный. По умолчанию будет выбираться значение в зависимости от одного из системных типов инфоблока.	
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);	

Пример

```
var params = {
    'IBLOCK_TYPE_ID': 'lists',
    'IBLOCK_ID': '41',
   'ELEMENT_ID': '683',
   'FIELD_ID': '120'
};
BX.rest.callMethod(
```

```
    'lists.element.get.file.url',
    params,
    function(result)
    {
        if(result.error())
            alert("Error: " +
result.error());
        else
            console.log(result.data());
    }
)
```

lists.element.update

Описание

```
lists.element.update(params, fields)
```

Метод обновляет элемент списка. В случае успешного обновления элемента ответ true, иначе Exception.

Важно! Все поля элемента и их значения должны передаваться в запросе.

Чтобы загрузить файлы в поле типа Файл (Диск) необходимо:

1. использовать rest api модуля disk: disk.folder.uploadfile и disk.storage.uploadfile. В ответе при загрузке этих файлов, вы будете получать "ID": 290.
2. Получить список ID загруженных файлов.
3. Затем с помощью rest api модуля lists добавлять файлы в нужное поле. В случае если у поля уже есть прикрепленные файлы вам нужно получить предыдущие значения из lists.element.get и передать их вместе с новыми:

```
var params = {
 'IBLOCK_TYPE_ID': 'lists',
 'IBLOCK_ID': '41',
 'ELEMENT_CODE': 'element1',
 'FIELDS': {
```

```
     'NAME': 'Test element 1',
     'PROPERTY_121': {'4754': ['50',
'n1582']} // либо без id 'PROPERTY_121':
{'n0': ['50', 'n1582']}
 }
};
BX.rest.callMethod(
 'lists.element.update',
 params,
 function(result)
 {
     if(result.error())
         alert("Error: " +
result.error());
     else
         alert("Success: " +
result.data());
 }
);
```

Значения в поле Файл (Диск) без префикса "n" это уже прикрепленные файлы (attachedId), а с префиксом это ваши новые файлы, уже загруженные предварительно в диск.

Параметры

Параметр	Описание
<u>Все поля</u> элемента и их значения должны передаваться в запросе.	
IBLOCK_TYPE_ID	id типа инфоблока (обязательно): ▪ **lists** - тип инфоблока списка

	▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп
IBLOCK_CODE/IBLOCK_ID	код или id инфоблока (обязательно)
ELEMENT_CODE/ELEMENT_ID	код или id элемента (обязательно)
FIELDS	массив полей и значений
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);

Пример

```
var params = {
        'IBLOCK_TYPE_ID': 'lists_socnet',
        'IBLOCK_CODE': 'rest_1',
        'ELEMENT_CODE': 'element_1',
        'FIELDS': {
            'NAME': 'Test element (Update)',
            'PROPERTY_62': {
                '599': 'Text string
(Update)'
            },
            'PROPERTY_63': {
                '600': '73',
                '601': '97',
                '602': '17'
            }
        }
    };
    BX24.callMethod(
        'lists.element.update',
```

```
        params,
        function(result)
        {
            if(result.error())
                alert("Error: " +
result.error());
            else
                alert("Success: " +
result.data());
        }
    );
```

lists.field.add

Описание

```
lists.field.add (params)
```

Метод создает поле списка. В случае успешного создания поля ответ *true*, иначе *Exception*.

Параметры

Параметр	Описание
IBLOCK_TYPE_ID	id типа инфоблока (обязательно): ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп
IBLOCK_CODE/IBLOCK_ID	код или id инфоблока (обязательно)
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);
FIELDS	(ключи такие же как при создании поля из интерфейса *Битрикс24*)

- **NAME** название (обязательно)
- **IS_REQUIRED** метка обязательности
- **MULTIPLE** метка множественности
- **TYPE** тип (обязательно)
  - **S** - Строка
  - **N** - Число
  - **L** - Список
  - **F** - Файл
  - **G** - Привязка к разделам
  - **E** - Привязка к элементам
  - **S:Date** - Дата
  - **S:DateTime** - Дата/Время
  - **S:HTML** - HTML/текст
  - **E:EList** - Привязка к элементам в виде списка. При создании поля с этим типом необходимо обязательно указать `LINK_IBLOCK_ID` id информационного блока, элементы которого будут отображаться в селекторе этого поля.
  - **N:Sequence** - Счетчик
  - **S:Money** - Деньги
  - **S:DiskFile** - Файл (Диск)
  - **S:map_yandex** - Привязка к Яндекс.Карте
  - **S:employee** - Привязка к сотруднику
  - **S:ECrm** - Привязка к элементам CRM
- **SORT** сортировка
- **DEFAULT_VALUE** значение по умолчанию

- **LIST** - может использоваться для добавления значений поля типа "Список".
    - n0 - строка пункта
        - SORT - значение сортировки
        - VALUE - значение пункта
- **LIST_TEXT_VALUES** - может использоваться для добавления значений поля типа "Список" с помощью строки.(Каждая уникальная строчка станет отдельным значением списка)
- **LIST_DEF** - значение по умолчанию для поля типа "Список"(Формат: массив с значением, где значение id пункта списка)
- **CODE** код (обязательно, если поле является свойством инфоблока)
- **SETTINGS** все ключи должны присутствовать, иначе будет происходить затирание значениями по умолчанию
    - SHOW_ADD_FORM - показывать в форме добавления
    - SHOW_EDIT_FORM - показывать в форме редактирования
    - ADD_READ_ONLY_FIELD - только для чтения (форма добавления)
    - EDIT_READ_ONLY_FIELD - только для чтения (форма редактирования)
    - SHOW_FIELD_PREVIEW - показать поле при

	формировании ссылки на элемент списка ▪ **USER_TYPE_SETTINGS** - ключ для передачи пользовательских настроек ▪ **ROW_COUNT/COL_COUNT** - настройка для полей textarea ▪ **LINK_IBLOCK_ID** - id привязываемого списка (раздела инфоблока)

Пример

```
   var params = {
        'IBLOCK_TYPE_ID': 'lists_socnet',
        'IBLOCK_CODE': 'rest_1',
        'SOCNET_GROUP_ID': '7'
        'FIELDS': {
            'NAME': 'List field',
            'IS_REQUIRED': 'Y',
            'MULTIPLE': 'N',
            'TYPE': 'L',
            'SORT': '20',
            'CODE': 'fieldList',
            'LIST_TEXT_VALUES':
'one\ntwo\nthree',
            'SETTINGS': {
                'SHOW_ADD_FORM': 'Y',
                'SHOW_EDIT_FORM': 'Y',
                'ADD_READ_ONLY_FIELD': 'N',
                'EDIT_READ_ONLY_FIELD': 'N',
                'SHOW_FIELD_PREVIEW': 'N'
            }
        }
    };
    BX24.callMethod(
```

```
        'lists.field.add',
        params,
        function(result)
        {
            if(result.error())
                alert("Error: " +
result.error());
            else
                alert("Success: " +
result.data());
        }
    );
```

lists.field.delete

```
lists.field.delete (params, fields)
```

Метод удаляет поле списка. В случае успешного удаления поля ответ *true*, иначе *Exception*.

Параметры

Параметр	Описание
IBLOCK_TYPE_ID	id типа инфоблока (обязательно): ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп
IBLOCK_CODE/IBLOCK_ID	код или id инфоблока (обязательно)
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);
FIELD_ID	ID поля. (обязательно. Если поле свойство инфоблока, то формат: "PROPERTY_propertyId")

Пример

```
    var params = {
        'IBLOCK_TYPE_ID': 'lists_socnet',
        'IBLOCK_CODE': 'rest_1',
        'FIELD_ID': 'PROPERTY_61'
    };
    BX24.callMethod(
        'lists.field.delete',
        params,
        function(result)
        {
            if(result.error())
                alert("Error: " +
result.error());
            else
                alert("Success: " +
result.data());
        }
    );
```

lists.field.get

```
lists.field.get (params)
```

Метод возвращает данные поля. В случае успеха будет возвращен список полей с данными, иначе пустой массив.

Параметры

Параметр	Описание
IBLOCK_TYPE_ID	id типа инфоблока (обязательно): ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп
IBLOCK_CODE/IBLOCK_ID	код или id инфоблока (обязательно)
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);
FIELD_ID	ID поля. (Если поле свойство инфоблока, то формат: "PROPERTY_propertyId". Если не

	указан в ответе будут данные по всем полям указанного списка)

Пример

```
    var params = {
        'IBLOCK_TYPE_ID': 'lists_socnet',
        'IBLOCK_CODE': 'rest_1'
    };
    BX24.callMethod(
        'lists.field.get',
        params,
        function(result)
        {
            if(result.error())
                alert("Error: " +
result.error());
            else
                console.log(result.data());
        }
    );
```

lists.field.type.get

```
lists.field.type.get (params, fields)
```

Метод возвращает доступные типа полей для указанного списка. В случае успеха будет возвращен список доступных типов полей, иначе пустой массив.

Параметры

Параметр	Описание
IBLOCK_TYPE_ID	id типа инфоблока (обязательно): ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп
IBLOCK_CODE/IBLOCK_ID	код или id инфоблока (обязательно)
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);
FIELD_ID	ID поля. (Если поле свойство инфоблока, то формат: `PROPERTY_propertyId`. Если

	указан, то возвращает доступные типы, с типом указанного поля)

Пример

```
    var params = {
        'IBLOCK_TYPE_ID': 'lists_socnet',
        'IBLOCK_CODE': 'rest_1'
    };
    if(!params.IBLOCK_CODE &&
!params.IBLOCK_ID)
        return;
    BX24.callMethod(
        'lists.field.type.get',
        params,
        function(result)
        {
            if(result.error())
            {
                alert("getFieldTypes: " +
result.error());
            }
            else
            {
                var types = result.data(),
html = '';
                for(var typeId in types)
                {

if(!types.hasOwnProperty(typeId)) continue;
                    html += '
'+types[typeId]+'
';
                }
```

```
                    $('#field-type').html(html);
                }
            }
        );
```

lists.field.update

Описание

```
lists.field.update (params, fields)
```

Метод обновляет поле списка. В случае успешного обновления поля ответ true, иначе Exception.

Параметры

Параметр	Описание
IBLOCK_TYPE_ID	id типа инфоблока (обязательно): ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп
IBLOCK_CODE/IBLOCK_ID	код или id инфоблока (обязательно)
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);
FIELD_ID	ID поля. (обязательно. Если поле свойство инфоблока, то формат:

	"PROPERTY_propertyId")
FIELDS	(ключи такие же как при создании поля из интерфейса *Битрикс24*) ▪ **NAME** название (обязательно) ▪ **IS_REQUIRED** метка обязательности ▪ **MULTIPLE** метка множественности ▪ **TYPE** тип (обязательно) ▪ **SORT** сортировка ▪ **DEFAULT_VALUE** значение по умолчанию ▪ **LIST** - может использоваться для добавления значений поля типа "Список". ▪ n0 - строка пункта ▪ SORT - значение сортировки ▪ VALUE - значение пункта ▪ **LIST_TEXT_VALUES** - может использоваться для добавления значений поля типа "Список" с помощью строки.(Каждая уникальная строчка станет отдельным значением списка) ▪ **LIST_DEF** - значение по умолчанию для поля типа "Список"(Формат: массив с значением, где значение id пункта списка) ▪ **CODE** код (обязательно, если поле является свойством инфоблока) ▪ **SETTINGS** все ключи должны присутствовать, иначе будет происходить затирание значениями по умолчанию

	▪ SHOW_ADD_FORM - показывать в форме добавления ▪ SHOW_EDIT_FORM - показывать в форме редактирования ▪ ADD_READ_ONLY_FIELD - только для чтения (форма добавления) ▪ EDIT_READ_ONLY_FIELD - только для чтения (форма редактирования) ▪ SHOW_FIELD_PREVIEW - показать поле при формировании ссылки на элемент списка ▪ **USER_TYPE_SETTINGS** - ключ для передачи пользовательских настроек ▪ **ROW_COUNT/COL_COUNT** - настройка для полей textarea ▪ **LINK_IBLOCK_ID** - id привязываемого раздела

Пример

```
var params = {
    'IBLOCK_TYPE_ID': 'lists_socnet',
    'IBLOCK_CODE': 'rest_1',
    'FIELD_ID': 'PROPERTY_61',
    'FIELDS': {
        'NAME': 'List field (Update)',
        'IS_REQUIRED': 'N',
        'MULTIPLE': 'N',
        'TYPE': 'L',
        'SORT': '20',
        'CODE': 'fieldList',
        'LIST': {
```

```
                '58': {
                    'SORT': '10',
                    'VALUE': 'one'
                },
                '59': {
                    'SORT': '20',
                    'VALUE': 'two'
                },
                '60': {
                    'SORT': '30',
                    'VALUE': 'three'
                }
            },
            'LIST_DEF': {
                '0': '59'
            },
            'SETTINGS': {
                'SHOW_ADD_FORM': 'Y',
                'SHOW_EDIT_FORM': 'Y',
                'ADD_READ_ONLY_FIELD': 'N',
                'EDIT_READ_ONLY_FIELD': 'Y',
                'SHOW_FIELD_PREVIEW': 'N'
            }
        }
    };
    BX24.callMethod(
        'lists.field.update',
        params,
        function(result)
        {
            if(result.error())
                alert("Error: " +
result.error());
            else
                alert("Success: " +
result.data());
```

```
            }
        );
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

lists.section.add

Описание

```
lists.section.add(
   params
)
```

Метод создаёт раздел списка. В случае успешного создания раздела ответ *true*, иначе Exception.

Параметры

Параметр	Описание	С версии
IBLOCK_TYPE_ID	Идентификатор типа инфоблока (обязательный). Возможные значения: ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп	

IBLOCK_CODE/IBLOCK_ID	Код или идентификатор инфоблока (обязательный).	
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);	
IBLOCK_SECTION_ID	Идентификатор раздела родителя, если не задан то раздел будет корневой	
FIELDS	Массив полей и значений: Доступные поля описаны в документации Структура таблиц модуля информационных блоков. Обязательные поля: NAME.)	
SECTION_CODE	Символьный код раздела (обязательный).	

Пример

```
/* lists.section.add */
var params = {
  'IBLOCK_TYPE_ID': 'lists',
  'IBLOCK_CODE': 'rest_1',
  'SECTION_CODE': 'Section_code_1'
  'FIELDS': {
  'NAME': 'Section_1',
```

```
    }
};
BX.rest.callMethod(
    'lists.section.add',
    params,
    function(result)
    {
        if(result.error())
            alert("Error: " +
result.error());
        else
            console.log(result.data());
    }
);
```

lists.section.delete

```
lists.section.delete(
   params
)
```

Метод удаляет раздел списка. В случае успешного создания раздела ответ *true*, иначе Exception.

Параметры

Параметр	Описание	С версии
IBLOCK_TYPE_ID	Идентификатор типа инфоблока (обязательный). Возможные значения: ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп	

IBLOCK_CODE/IBLOCK_ID	Код или идентификатор инфоблока (обязательный).	
SECTION_CODE/SECTION_ID	Код или идентификатор раздела (обязательный).	
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);	

Пример

```
/* lists.section.delete */
var params = {
   'IBLOCK_TYPE_ID': 'lists',
   'IBLOCK_CODE': 'rest_1',
'SECTION_CODE': 'Section_code_1'
};
BX.rest.callMethod(
   'lists.section.delete',
   params,
   function(result)
   {
       if(result.error())
           alert("Error: " +
result.error());
       else
           console.log(result.data());
   }
);
```

lists.section.get

```
lists.section.get(
   params
)
```

Метод возвращает список разделов или раздел. В случае успеха возвращает данные по разделу(лам), иначе пустой массив.

Параметр	Описание	С версии
IBLOCK_TYPE_ID	Идентификатор типа инфоблока (обязательный). Возможные значения: ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп	
IBLOCK_CODE/IBLOCK_ID	Код или идентификатор инфоблока (обязательный).	
FILTER	Массив полей и значений	

	для фильтрации. Для фильтрации доступны поля из фильтра CIBlockSection::GetList *	
SELECT	Массив с полями для выборки. Доступные поля описаны в документации CIBlockSection::GetList *	
SOCNET_GROUP_ID	Идентификатор группы (обязательно, если список создается для группы);	

* - кроме пользовательских (UF_) полей.

Пример

```
/* lists.section.get */
var params = {
   'IBLOCK_TYPE_ID': 'lists',
   'IBLOCK_CODE': 'rest_1',
'FILTER': {
'NAME': 'section_%'
},
'SELECT': ['ID', 'NAME']
};
BX24.callMethod(
   'lists.section.get',
   params,
   function(result)
   {
       if(result.error())
           alert("Error: " +
```

```
result.error());
        else
            console.log(result.data());
    }
);
```

lists.section.update

```
lists.section.update(
   params
)
```

Метод обновляет раздел списка. В случае успешного обновления элемента ответ *true*, иначе Exception.

Параметр	Описание	С версии
IBLOCK_TYPE_ID	Идентификатор типа инфоблока (обязательный). Возможные значения: ▪ **lists** - тип инфоблока списка ▪ **bitrix_processes** - тип инфоблока процессов ▪ **lists_socnet** - тип инфоблока списков групп	
IBLOCK_CODE/IBLOCK_ID	Код или идентификатор инфоблока (обязательный).	

IBLOCK_SECTION_ID	Идентификатор раздела родителя, если не задан то раздел будет корневой	
FIELDS	Массив полей и значений: Доступные поля описаны в документации Структура таблиц модуля информационных блоков. Обязательные поля: NAME)	
SOCNET_GROUP_ID	id группы (обязательно, если список создается для группы);	

Пример

```
/* lists.section.update */
var params = {
   'IBLOCK_TYPE_ID': 'lists',
   'IBLOCK_CODE': 'rest_1',
'SECTION_CODE': 'Section_code_1',
'FIELDS': {
'NAME': 'Section_1 (Updated)'
}
};
BX.rest.callMethod(
   'lists.section.update',
   params,
   function(result)
   {
```

```
        if(result.error())
            alert("Error: " +
result.error());
        else
            console.log(result.data());
    }
);
```

timeman.close

Описание

Метод завершает текущий рабочий день.

Пример

Пример вызова:

```
https://portal.bitrix24.com/rest/timeman.close?auth=xxxxx&time=2017-04-21T17%3A30%3A00%2B09%3A00&report=test&lat=52.5162434&lon=13.3774829&user_id=5
```

Пример ответа:


```
{
    "result": {
        "STATUS": "CLOSED",
        "TIME_START": "2017-04-21T07:30:00+08:00",
        "TIME_FINISH": "2017-04-21T16:30:00+08:00",
        "DURATION": "09:00:00",
        "TIME_LEAKS": "00:00:00",
        "ACTIVE": false,
```

```
            "IP_OPEN": "192.168.0.1",
            "IP_CLOSE": "192.168.0.100",
            "LAT_OPEN": 54.7199881,
            "LON_OPEN": 20.4879224,
            "LAT_CLOSE": 52.5162434,
            "LON_CLOSE": 13.3774829,
            "TZ_OFFSET": 28800
        }
    }
```

Параметры

Параметр	Описание
USER_ID	Идентификатор пользователя. Не обязателен, по умолчанию возвращаются настройки текущего пользователя.
TIME	Время окончания рабочего дня. Не обязателен, по умолчанию рабочий день закрывается текущим моментом с часовым поясом, в котором рабочий день был начат. Дата должна совпадать с датой начала рабочего дня.
REPORT	Причина изменения рабочего дня. Обязателен при указании параметра TIME и отключенном для сотрудника свободном графике.
LAT	Географическая широта начала рабочего дня. Не обязателен.
LON	Географическая долгота начала рабочего дня. Не обязателен

Если часовая зона времени завершения рабочего дня отличается от часового пояса начала рабочего дня, то время будет пересчитано в часовой пояс начала дня.

Поля ответа

Поле	Значение	Описание	При
STATUS	string	Статус текущего рабочего дня.	Вариан значени ▪ OP ра ид ▪ CL ра за ▪ PA ра пр ▪ EX ра ис от на те ка су за
TIME_START	datetime	Дата-время начала рабочего дня.	Часовоі соответ часовом начала дня
TIME_FINISH	datetime	Дата-время заверения рабочего дня	Для незавер рабочеі возвраі
DURATION	HH:MM:SS	Длительность рабочего дня	Для незавер рабочеі возвраі 00:00:0
TIME_LEAKS	HH:MM:SS	Суммарная	

		длительность перерыва за день	
ACTIVE	true|false	Подтвержденность рабочего дня	Значени означае изменен рабочем ожидает подтвер руковод
IP_OPEN	string	IP-адрес, с которого начат рабочий день	
IP_CLOSE	string	IP-адрес, с которого завершен рабочий день	Для незавер рабочем возвраш
LAT_OPEN	double	Географическая широта точки начала рабочего дня	
LON_OPEN	double	Географическая долгота точки начала рабочего дня	
LAT_CLOSE	double	Географическая широта точки завершения рабочего дня	
LON_CLOSE	double	Географическая долгота точки завершения рабочего дня	
TZ_OFFSET	int	Смещение часового пояса сотрудника	Подразу часовой котором начат. I заверш

			время з приводі часовом начала
TIME_FINISH_DEFAULT	datetime	Время завершения дня по умолчанию	Выводи для раб статусе "Рекоме значени заверш которое выводи пользов качеств по умол

timeman.open

Описание и пример

Метод начинает новый рабочий день либо возобновляет закрытый или приостановленный.

Пример

Пример вызова:

```
https://portal.bitrix24.com/rest/timeman.open?auth=xxxxxx&time=2017-04-21T07%3A30%3A00%2B08%3A00&report=Forgot&lat=54.7199881&lon=20.4879224&user_id=5
```

Пример ответа:


```
{
    "result": {
        "STATUS": "OPENED",
        "TIME_START": "2017-04-21T07:30:00+08:00",
        "TIME_FINISH": null,
        "DURATION": "00:00:00",
        "TIME_LEAKS": "00:00:00",
        "ACTIVE": false,
```

```
            "IP_OPEN": "192.168.1.1",
            "IP_CLOSE": "",
            "LAT_OPEN": 54.7199881,
            "LON_OPEN": 20.4879224,
            "LAT_CLOSE": 0,
            "LON_CLOSE": 0,
            "TZ_OFFSET": 28800
        }
    }
```

Параметры

Параметр	Описание
USER_ID	Идентификатор пользователя. Не обязателен, по умолчанию возвращаются настройки текущего пользователя.
TIME	Время начала рабочего дня. Не обязателен, по умолчанию рабочий день открывается текущим моментом с сохраненным часовым поясом сотрудника. Дата должна совпадать с текущей календарной датой.
REPORT	Причина изменения рабочего дня. Обязателен при указании параметра TIME и отключенном для сотрудника свободном графике.
LAT	Географическая широта начала рабочего дня. Не обязателен.
LON	Географическая долгота начала рабочего дня. Не обязателен

Временная зона из параметра TIME считается временной зоной текущего рабочего дня. Сотруднику выводится время в этой временной зоне, а время завершения дня приводится к этой временной зоне, даже если указано в другой. Параметр TIME принимается только если рабочий день находится в статусе CLOSED, во всех остальных случаях вернется ошибка.

Поля ответа

Поле	Значение	Описание	При
STATUS	string	Статус текущего рабочего дня.	Вариан значени ▪ OP ра ид ▪ CL ра за ▪ PA ра пр ▪ EX ра ис от на те ка су за
TIME_START	datetime	Дата-время начала рабочего дня.	Часовоі соответ часовом начала дня
TIME_FINISH	datetime	Дата-время заверения рабочего дня	Для незавер рабочеі возвраі
DURATION	HH:MM:SS	Длительность рабочего дня	Для незавер рабочеі возвраі 00:00:0
TIME_LEAKS	HH:MM:SS	Суммарная	

		длительность перерыва за день	
ACTIVE	true|false	Подтвержденность рабочего дня	Значени означае изменен рабочем ожидает подтвер руковод
IP_OPEN	string	IP-адрес, с которого был начат рабочий день	
IP_CLOSE	string	IP-адрес, с которого был завершен рабочий день	Для незавер рабочем возвраш
LAT_OPEN	double	Географическая широта точки начала рабочего дня	
LON_OPEN	double	Географическая долгота точки начала рабочего дня	
LAT_CLOSE	double	Географическая широта точки завершения рабочего дня	
LON_CLOSE	double	Географическая долгота точки завершения рабочего дня	
TZ_OFFSET	int	Смещение часового пояса сотрудника	Подразу часовой котором начат. I

			заверш время з приводи часовом начала
TIME_FINISH_DEFAULT	datetime	Время завершения дня по умолчанию	Выводи для раб статусе "Рекоме значени заверш которое выводи пользов качеств по умол

timeman.pause

Описание и примеры

Метод приостанавливает текущий рабочий день.

Примеры

Пример вызова:

```
https://portal.bitrix24.com/rest/timeman.pau
se?user_id=5
```

Пример ответа:


```
{
    "result": {
        "STATUS": "PAUSED",
        "TIME_START": "2017-04-
21T07:30:00+08:00",
        "TIME_FINISH": "2017-04-
21T22:03:58+08:00",
        "DURATION": "00:00:00",
        "TIME_LEAKS": "00:00:00",
        "ACTIVE": false,
        "IP_OPEN": "192.168.0.1",
        "IP_CLOSE": "192.168.0.100",
```

```
            "LAT_OPEN": 54.7199881,
            "LON_OPEN": 20.4879224,
            "LAT_CLOSE": 0,
            "LON_CLOSE": 0,
            "TZ_OFFSET": 28800
        }
    }
```

Параметры

Параметр	Описание
USER_ID	Идентификатор пользователя. Не обязателен, по умолчанию возвращаются настройки текущего пользователя.

Поля ответа

Поле	Значение	Описание	При
STATUS	string	Статус текущего рабочего дня.	Вариант значени ▪ OP ра ид ▪ CL ра за ▪ PA ра пр ▪ EX ра ис от на те

			ка су за
TIME_START	datetime	Дата-время начала рабочего дня.	Часовой соответ часовом начала дня
TIME_FINISH	datetime	Дата-время заверения рабочего дня	Для незавер рабочем возвраш
DURATION	HH:MM:SS	Длительность рабочего дня	Для незавер рабочем возвраш 00:00:0
TIME_LEAKS	HH:MM:SS	Суммарная длительность перерыва за день	
ACTIVE	true|false	Подтвержденность рабочего дня	Значени означае изменен рабочем ожидает подтвер руковод
IP_OPEN	string	IP-адрес, с которого был начат рабочий день	
IP_CLOSE	string	IP-адрес, с которого был завершен рабочий день	Для незавер рабочем возвраш
LAT_OPEN	double	Географическая широта точки	

		начала рабочего дня	
LON_OPEN	double	Географическая долгота точки начала рабочего дня	
LAT_CLOSE	double	Географическая широта точки завершения рабочего дня	
LON_CLOSE	double	Географическая долгота точки завершения рабочего дня	
TZ_OFFSET	int	Смещение часового пояса сотрудника	Подраз часовой котором начат. I заверш время з приводі часовом начала
TIME_FINISH_DEFAULT	datetime	Время завершения дня по умолчанию	Выводи для раб статусе "Рекоме значени заверш которое выводи пользов качеств по умол

timeman.settings

Описание и примеры

Метод возвращает настройки рабочего времени пользователя.

Примеры

Пример вызова:

```
https://portal.bitrix24.com/rest/timeman.settings/?auth=xxxxxx&user_id=1
```

Пример ответа:

```
HTTP/1.1 200 OK

{
    "result": {
        "ADMIN": true,
        "UF_TIMEMAN": true,
        "UF_TM_ALLOWED_DELTA": "00:15:00",
        "UF_TM_FREE": false,
        "UF_TM_MAX_START": "09:15:00",
        "UF_TM_MIN_DURATION": "08:00:00",
        "UF_TM_MIN_FINISH": "17:45:00"
```

```
        }
 }
```

Параметры

Параметр	Описание
USER_ID	Идентификатор пользователя. Не обязателен, по умолчанию возвращаются настройки текущего пользователя.

Поля ответа

Поле	Значение	Описание	Примеч
ADMIN	true|false	Обладает ли пользователь правами на управление чужими рабочими днями	Возвраща только дл текущего пользоват
UF_TIMEMAN	true|false	Включен ли учет рабочего времени для пользователя	
UF_TM_FREE	true|false	Включен ли для пользователя свободный график работы	При включ свободном графике изменения рабочего не требую подтвержд указания причины
UF_TM_MAX_START	HH:MM:SS	Настроенное	Рабочий д

		для пользователя максимальное время начала рабочего дня	начатый п указанног времени, (считаться нарушени
UF_TM_MIN_FINISH	HH:MM:SS	Настроенное для пользователя минимальное время завершения рабочего дня	Рабочий д завершен раньше указанног времени, (считаться нарушени
UF_TM_MIN_DURATION	HH:MM:SS	Настроенная для пользователя минимальная длительность рабочего дня	Рабочий д длительно меньше указанной считаться нарушени
UF_TM_ALLOWED_DELTA	HH:MM:SS	Настроенный для пользователя допустимый промежуток изменения рабочего времени	Изменение рабочего д промежуто меньше указанног будет треб подтвержд руководит

timeman.status

Описание и примеры

Метод возвращает информацию о текущем рабочем дне.

Примеры

Пример вызова:

```
https://portal.bitrix24.com/rest/timeman.sta
tus/?auth=xxxxxx&user_id=1
```

Пример ответа:


```
{
    "result": {
        "STATUS": "PAUSED",
        "TIME_START": "2017-04-
21T02:33:44+02:00",
        "TIME_FINISH": "2017-04-
21T15:11:26+02:00",
        "DURATION": "00:00:00",
        "TIME_LEAKS": "00:00:00",
        "ACTIVE": false,
        "IP_OPEN": "192.168.0.1",
        "IP_CLOSE": "192.168.0.100",
```

```
            "LAT_OPEN": 55.33,
            "LON_OPEN": 20.5,
            "LAT_CLOSE": 0,
            "LON_CLOSE": 0,
            "TZ_OFFSET": 7200
        }
    }
```

Параметры

Параметр	Описание
USER_ID	Идентификатор пользователя. Не обязателен, по умолчанию возвращаются настройки текущего пользователя.

Поля ответа

Поле	Значение	Описание	При
STATUS	string	Статус текущего рабочего дня.	Вариант значени ▪ OP ра ид ▪ CL ра за ▪ PA ра пр ▪ EX ра ис от на те

			ка су за
TIME_START	datetime	Дата-время начала рабочего дня.	Часовой соответ часовом начала дня
TIME_FINISH	datetime	Дата-время заверения рабочего дня	Для незавер рабочем возвраш
DURATION	HH:MM:SS	Длительность рабочего дня	Для незавер рабочем возвраш 00:00:0
TIME_LEAKS	HH:MM:SS	Суммарная длительность перерыва за день	
ACTIVE	true|false	Подтвержденность рабочего дня	Значени означае изменен рабочем ожидает подтвер руковод
IP_OPEN	string	IP-адрес, с которого был начат рабочий день	
IP_CLOSE	string	IP-адрес, с которого был завершен рабочий день	Для незавер рабочем возвраш
LAT_OPEN	double	Географическая широта точки	

		начала рабочего дня	
LON_OPEN	double	Географическая долгота точки начала рабочего дня	
LAT_CLOSE	double	Географическая широта точки завершения рабочего дня	
LON_CLOSE	double	Географическая долгота точки завершения рабочего дня	
TZ_OFFSET	int	Смещение часового пояса сотрудника	Подраз часовой котором начат. I заверш время з приводі часовом начала
TIME_FINISH_DEFAULT	datetime	Время завершения дня по умолчанию	Выводи для раб статусе "Рекоме значени заверш которое выводи пользов качеств по умол

Если пользователь ни разу не начинал рабочий день, будет отдан сокращенный ответ, содержащий только поле STATUS со значением CLOSED.

timeman.networkrange.check

Описание и параметры

Метод для проверки IP-адреса на вхождение в диапазоны сетевых адресов офисной сети.

Параметры

Параметр	Пример	Обязательный	Описание
IP	10.10.255.255	Нет	IP-адрес.

Если не указать параметр `IP`, то проверка будет выполнена для текущего IP-адреса.

Пример вызова

JavaScript

```
BX24.callMethod('timeman.networkrange.check'
, {
        'IP': '10.10.255.255'
}, function(result){
        if(result.error())
        {
```

```
console.error(result.error().ex);
        }
        else
        {
                console.log(result.data());
        }
});
```

PHP

```
$result =
restCommand('timeman.networkrange.check',
Array(
        'IP' => '10.10.255.255'
), $_REQUEST["auth"]);
```

Пример ответа

```
{
        "result": {
                ip: "10.10.255.255",
                range: "10.0.0.0-
10.255.255.255",
                name: "Офисная сеть
10.x.x.x"
        }
}
```

Описание ключей

- **ip** - IP-адрес, который был проверен.

- **range** - диапазон в который входит указанный IP-адрес.
- **name** - название диапазона в который входит указанный IP-адрес.

Пример ответа при возникновении ошибки

```
{
    "error": "ACCESS_ERROR",
    "error_description": "You don't have
access to user this method"
}
```

- Ключ **error** - код возникшей ошибки.
- Ключ **error_description** - краткое описание возникшей ошибки.

Возможные коды ошибок

Код	Описание
ACCESS_ERROR	Указанный метод доступен только администраторам.

timeman.networkrange.get

Описание

Метод для получения диапазонов сетевых адресов, входящих в офисную сеть.

Параметры

Без параметров.

Пример вызова

JavaScript

```
BX24.callMethod('timeman.networkrange.get',
{}, function(result){
        if(result.error())
        {

console.error(result.error().ex);
        }
        else
        {
                console.log(result.data());
        }
});
```

PHP

```
$result =
restCommand('timeman.networkrange.get',
Array(), $_REQUEST["auth"]);
```

Пример ответа

```
{
        "result": [
                {"ip_range":"10.0.0.0-
10.255.255.255","name":"Офисная сеть
10.x.x.x"},
                {"ip_range":"172.16.0.0-
172.31.255.255","name":"Офисная сеть
172.x.x.x"},
                {"ip_range":"192.168.0.0-
192.168.255.255","name":"Офисная сеть
192.168.x.x"}
        ]
}
```

Описание ключей

- **ip_range** - диапазон сетевых адресов.
- **name** - название диапазона.

Пример ответа при возникновении ошибок

```
{
    "error": "ACCESS_ERROR",
    "error_description": "You don't have
access to user this method"
}
```

- Ключ **error** - код возникшей ошибки.
- Ключ **error_description** - краткое описание возникшей ошибки.

Возможные коды ошибок

Код	Описание
ACCESS_ERROR	Указанный метод доступен только администраторам.

timeman.networkrange.set

Описание и параметры

Метод для установки диапазонов сетевых адресов, входящих в офисную сеть.

Параметры

Параметр	Пример	Обязательн
RANGES	[{"ip_range":"10.0.0.0-10.255.255.255","name":"Офисная сеть 10.x.x.x"}]	Да

Диапазон может содержать блок адресов, например 10.0.0.0-10.255.255.255 или всего один адрес 10.10.0.1

Пример вызова

JavaScript

```
BX24.callMethod('timeman.networkrange.set',
{
        ranges: '[{"ip_range":"10.0.0.0-
10.255.255.255","name":"Офисная сеть
10.x.x.x"},{"ip_range":"172.16.0.0-
172.31.255.255","name":"Офисная сеть
172.x.x.x"},{"ip_range":"192.168.0.0-
```

```
192.168.255.255","name":"Офисная сеть
192.168.x.x"}]'
}, function(result){
        if(result.error())
        {

console.error(result.error().ex);
        }
        else
        {
                var answer = result.data();
                if (answer.result)
                {
                        console.log('range
saved');
                }
                else
                {
                        console.warn('An
error occurred while saving, the following
ranges are incorrect', answer.error_ranges);
                }
        }
});
```

PHP

```
$result =
restCommand('timeman.networkrange.set',
Array(
        'RANGES' => Array(
                Array("ip_range" =>
"10.0.0.0-10.255.255.255", "name" =>
"Офисная сеть 10.x.x.x"),
                Array("ip_range" =>
"172.16.0.0-172.31.255.255", "name" =>
```

```
"Офисная сеть 172.x.x.x"),
                Array("ip_range" =>
"192.168.0.0-192.168.255.255", "name" =>
"Офисная сеть 192.168.x.x")
        )
), $_REQUEST["auth"]);
```

Пример ответа

При успешном сохранении

```
{
        "result": {
                result: true
        }
}
```

При возникновении ошибки разбора диапазонов

```
{
        "result": {
                result: false,
                error_range: [
                        {ip_range:
"a10.0.0.0-10.255.255.255", name: "Офисная
сеть 10.x.x.x"}
                ]
        }
}
```

Описание ключей

- **result** - результат сохранения.
- **error_range** - массив диапазонов в которых были найдены ошибки:
  - **ip_range** - диапазон сетевых адресов.
  - **name** - название диапазона.

Пример ответа при возникновении ошибок

```
{
    "error": "ACCESS_ERROR",
    "error_description": "You don't have
access to user this method"
}
```

- Ключ **error** - код возникшей ошибки.
- Ключ **error_description** - краткое описание возникшей ошибки.

Возможные коды ошибок

Код	Описание
ACCESS_ERROR	Указанный метод доступен только администраторам.
INVALID_FORMAT	Передан не корректный формат в поле RANGE.

timeman.timecontrol.report.ad

Описание и параметры

Метод для отправки отчета о выявленном отсутствии.

Параметры

Параметр	Пример	Обязательный	Описание
ID	468	Да	Идентификатор записи.
TYPE	work	Да	Тип отсутствия (`work` - по рабочим вопросам, `priva` - личные дела).
TEXT	'Был на обеде'	Да	Описание причины отсутствия.
CALENDAR	true	Нет	Занести отсутствие в календарь (толь для первичного отчета).
USER_ID	2	Нет	Идентификатор пользователя дл

			которого сформирован отчет (поле доступно только администратора

Пример вызова

JavaScript

```
BX24.callMethod('timeman.timecontrol.report.
add', {
        'id': 468,
        'type': 'private',
        'text': 'Был на обеде',
        'calendar': true
}, function(result){
        if(result.error())
        {

console.error(result.error().ex);
        }
        else
        {
                console.log(result.data());
        }
});
```

PHP

```
$result =
restCommand('timeman.timecontrol.report.add'
, Array(
        'ID' => 468,
        'TYPE' => 'private',
```

```
        'TEXT' => 'Был на обеде',
        'CALENDAR' => true
), $_REQUEST["auth"]);
```

Пример ответа

```
{
        "result": true
}
```

Пример ответа при возникновении ошибки

```
{
    "error": "TEXT_EMPTY",
    "error_description": "Text can't be
empty"
}
```

- Ключ **error** - код возникшей ошибки.
- Ключ **error_description** - краткое описание возникшей ошибки.

Возможные коды ошибок

Код	Описание
TEXT_EMPTY	Не передана причина отсутствия.
ACCESS_ERROR	У вас нет доступа к этому отчету.

timeman.timecontrol.reports.g

Описание и параметры

Метод для получения отчета о выявленных отсутствиях.

Параметры

Параметр	Пример	Обязательный	Описа
USER_ID	2	Да	Идентифика пользовател
YEAR	2018	Да	Год для формирован отчета.
MONTH	8	Да	Месяц для формирован отчета.
WORKDAY_HOURS	8	Нет	Необходима продолжите рабочего дн часах (по-умолчанию часов).
IDLE_MINUTES	15	Нет	Максимальн время отсут на рабочем

			которое не б учитываться отсутствие.

Параметр `IDLE_MINUTES` доступен только, если вы руководитель или администратор. Если не указывать, время автоматически берется из настроек модуля.

Пример вызова

JavaScript

```
BX24.callMethod('timeman.timecontrol.reports
.get', {
        user_id: 2,
        year: 2018,
        month: 8,
        workday_hours: 8,
        idle_minutes: 15
}, function(result){
        if(result.error())
        {

console.error(result.error().ex);
        }
        else
        {
                console.log(result.data());
        }
});
```

PHP

```
$result =
restCommand('timeman.timecontrol.reports.get
', Array(
```

```
        'USER_ID => 2,
        'YEAR' => 2018,
        'MONTH' => 8,
        'WORKDAY_HOURS' => 8,
        'IDLE_MINUTES' => 15
), $_REQUEST["auth"]);
```

Пример ответа


```
{
        "result": {
                report: {
                        month_title:
'Август',
                        date_start: "2018-
08-01T00:00:00+03:00",
                        date_finish: "2018-
08-31T23:59:59+03:00",
                        days: [
                                {

index: "20180816",

day_title: "16.08.2018",

workday_complete: false,

workday_date_start: "2018-08-
16T14:08:35+03:00",

workday_date_finish: "2018-08-
16T11:20:00+03:00",

workday_duration: 10115,
```

workday_duration_config: 28800,

workday_duration_final: 6914,

workday_time_leaks_user: 0,

workday_time_leaks_real: 3201,

workday_time_leaks_final: 21886,

reports: [

{

id: 459,

type: "TM_START",

date_start: "2018-08-16T14:08:35+03:00",

date_finish: "2018-08-16T14:08:35+03:00",

duration: 0,

active: false,

entry_id: 35,

report_type: "NONE",

report_text: null,

system_text: "",

source_start: "TM_EVENT",

source_finish: "TM_EVENT",

```
ip_start: "93.60.295.11",

ip_finish: "10.10.255.255",

ip_start_network: false,

ip_finish_network: {

ip: "10.10.255.255",

range: "10.0.0.0-10.255.255.255",

name: "Офисная сеть 10.x.x.x"

}

}
                                        ]
                                }
                        ]
                },
                user: {
                        id:2,
                        name:"Мария Ившина",
                        first_name:"Мария",
                        last_name:"Ившина",
                        work_position:"IT-
специалист",

avatar:"http://test.bitrix24.com/upload/resi
ze_cache/main/072/100_100_2/42-
17948709.gif",

last_activity_date:"2018-08-
15T16:25:34+03:00"
                }
```

```
            }
}
```

Описание ключей

- **report** - информация об отчете.
  - **month_title** - название месяца.
  - **date_start** - дата начала периода выборки в формате ATOM.
  - **date_finish** - дата окончания периода выборки в формате ATOM.
  - **days** - список отработанных дней.
    - **index** - индекс дня недели.
    - **day_title** - дата (в формате сайте).
    - **workday_complete** - рабочий день завершен (true / false).
    - **workday_date_start** - дата начала рабочего дня в формате ATOM.
    - **workday_date_finish** - дата окончания рабочего дня в формате ATOM (если `workday_complete = false`, в данном поле указывается дата на момент формирования отчета).
    - **workday_duration** - продолжительность рабочего дня по табелю в секундах (с учетом установленного пользователем перерыва).
    - **workday_duration_final** - продолжительность рабочего дня по фактической выработке в секундах (с учетом установленного пользователем перерыва, не подтвержденных отсутствий и отсутствий по личным делам).
    - **workday_duration_config** - необходимая продолжительность рабочего дня в секундах.
    - **workday_time_leaks_user** - продолжительность перерыва установленного пользователем в секундах.
    - **workday_time_leaks_real** - продолжительность перерыва установленного автоматической системой

фиксации (не подтвержденные отсутствий и отсутствий по личным делам).

- **workday_time_leaks_final** - если число положительное: количество не доработанного времени в секундах; если число отрицательное: количество времени отработанных сверх положенного времени (переработка).
- **reports** - список записей выявленных отсутствий (значения доступны только на уровне детализации head и full).
    - **id** - идентификатор записи, необходим для использования метода timeman.timecontrol.report.add.
    - **type** - тип записи (справочник типов описан ниже).
    - **active** - активность записи.
    - **date_start**- дата начала фиксации в формате ATOM.
    - **date_finish** - дата окончания фиксации в формате ATOM (если active = true в данном поле указывается дата на момент формирования отчета).
    - **report_type** - тип отсутствия (work - по рабочим вопросам, private - личные дела, none - не задан, приравнивается к private).
    - **report_text** - описание причины отсутствия.
    - **system_text** - системное описание причины отсутствия (параметр доступен только на уровне детализации head).
    - **source_start** - поставщик данных начала записи (справочник типов описан ниже).
    - **source_finish** - поставщик данных окончания записи (справочник типов описан ниже).
    - **ip_start** - ip-адрес на момент начала записи (параметр доступен только на уровне детализации head).
    - **ip_start_network** - расшифровка ip-адреса на момент начала записи, если ip-адрес не входит в офисную сеть - false (параметр доступен только на уровне детализации head).
        - **ip** - ip-адрес на момент начала записи .
        - **range** - диапазон, в который входит указанный IP-адрес.

        - **name** - название диапазона, в который входит указанный IP-адрес.

    - **ip_finish** - ip-адрес на момент окончания записи (параметр доступен только на уровне детализации head).
    - **ip_finish_network** - расшифровка ip-адрес на момент окончания записи, если ip-адрес не входит в офисную сеть - false (параметр доступен только на уровне детализации head).

        - **ip** - ip-адрес на момент окончания записи.
        - **range** - диапазон, в который входит указанный IP-адрес.
        - **name** - название диапазона, в который входит указанный IP-адрес.

- **user** - информация о пользователе.

    - **id** - идентификатор пользователя.
    - **name** - имя и фамилия пользователя.
    - **first_name** - имя пользователя.
    - **last_name** - фамилия пользователя.
    - **work_position** - должность.
    - **avatar** - ссылка на аватар (если пусто, значит аватар не задан).
    - **personal_gender** - пол пользователя.
    - **last_activity_date** - дата последнего действия пользователя в формате ATOM.

Возможные типы записей (type)

- `IDLE` - Отошел (фиксируется с помощью десктоп приложения).
- `OFFLINE` - Офлайн.
- `DESKTOP_ONLINE` - Зафиксирован запуск десктоп приложения (тип доступен только на уровне детализации head).
- `DESKTOP_OFFLINE` - Зафиксирован выключенное десктоп приложение (тип доступен только на уровне детализации head).

- `DESKTOP_START` - Зафиксирован запуск десктоп приложения (тип доступен только на уровне детализации head).
- `TM_START` - Начал рабочий день.
- `TM_PAUSE` - Ушел на перерыв.
- `TM_CONTINUE` - Продолжил день.
- `TM_END` - Завершил рабочий день.

Возможные источники записей (source_start, source_finish)

- `ONLINE_EVENT` - Событие авторизации пользователя.
- `OFFLINE_AGENT` - Агент проставляющий статус офлайн.
- `DESKTOP_OFFLINE_AGENT` - Агент проставляющий признак выключенного десктоп приложения.
- `DESKTOP_ONLINE_EVENT` - Событие проставляющий признак включенного десктоп приложения.
- `DESKTOP_START_EVENT` - Событие проставляющий признак включенного десктоп приложения.
- `IDLE_EVENT` - Событие изменение стаса отошел (фиксируется с помощью десктоп приложения).
- `TM_EVENT` - Событие изменения рабочего дня (начало, перерыв, окончание).

Пример ответа при возникновении ошибки

```
{
    "error": "ACCESS_ERROR",
    "error_description": "You don't have
access to this method"
}
```

- Ключ **error** - код возникшей ошибки.
- Ключ **error_description** - краткое описание возникшей ошибки.

Возможные коды ошибок

Код	Описание
ACCESS_ERROR	Указанный метод не доступен пользователю.
USER_ACCESS_ERROR	Нет доступа к указанному пользователю.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

timeman.timecontrol.reports.s

Описание

Метод для получения пользовательских настроек для построения интерфейса отчетов инструмента контроля времени.

Параметры

Без параметров.

Пример вызова

JavaScript

```
BX24.callMethod('timeman.timecontrol.reports
.settings.get', {}, function(result){
        if(result.error())
        {

console.error(result.error().ex);
        }
        else
        {
                console.log(result.data());
        }
});
```

PHP

```
$result =
restCommand('timeman.timecontrol.reports.set
tings.get', Array(), $_REQUEST["auth"]);
```

Пример ответа


```
{
        "result": {
                "active":true,
                "user_id":2,
                "user_admin":true,
                "user_head":true,
                "departments":[

{"id":"92","name":"Калининградский филиал"},

{"id":"93","name":"Администрация"},

{"id":"106","name":"ИТ-отдел"}
                ],
                "minimum_idle_for_report":1,
                "report_view_type":"head"
        }
}
```


Описание ключей

- **active** - доступность инструмента контроля времени.
- **user_id** - текущий идентификатор пользователя.

- **user_admin** - флаг являетесь ли вы администратором.
- **user_head** - флаг являетесь ли вы руководителем.
- **departments** - список доступных подразделений (доступно только если вы руководитель)
  - **id** - идентификатор подразделения
  - **name** - название подразделения
- **minimum_idle_for_report** - минимальное кол-во времени для запроса отчета (в минутах)
- **report_view_type** - уровень детализации отчета (head, full, simple)

timeman.timecontrol.reports.u

Описание и параметры

Метод для получения списка пользователей, относящихся к указанному подразделению.

Параметры

Параметр	Пример	Обязательный	Описани
DEPARTMENT_ID	52	Да*	Идентификато подразделени

* - параметр `DEPARTMENT_ID` необходимо указывать только, если пользователь - руководитель или администратор.

Пример вызова

JavaScript

```
BX24.callMethod('timeman.timecontrol.reports
.users.get', {
        'DEPARTMENT_ID': 52
}, function(result){
        if(result.error())
        {
```

```
console.error(result.error().ex);
        }
        else
        {
                console.log(result.data());
        }
});
```

PHP

```
$result =
restCommand('timeman.timecontrol.reports.use
rs.get', Array(
        'DEPARTMENT_ID' => 52
), $_REQUEST["auth"]);
```

Пример ответа

```
{
        "result": [
                {
                        "id":2,
                        "name":"Мария
Ившина",

"first_name":"Мария",

"last_name":"Ившина",
                        "work_position":"IT-
специалист",

"avatar":"http://test.bitrix24.com/upload/re
size_cache/main/072/100_100_2/42-
```

```
17948709.gif",

"last_activity_date":"2018-08-
15T16:25:34+03:00"
                }
        ]
}
```


Описание ключей

- **id** - идентификатор пользователя.
- **name** - имя и фамилия пользователя.
- **first_name** - имя пользователя.
- **last_name** - фамилия пользователя.
- **work_position** - должность.
- **avatar** - ссылка на аватар (если пусто, значит аватар не задан).
- **personal_gender** - пол пользователя.
- **last_activity_date** - дата последнего действия пользователя в формате ATOM.

timeman.timecontrol.settings.g

Описание

Метод для получения настроек инструмента контроля времени.

Параметры

Без параметров.

Пример вызова

JavaScript

```
BX24.callMethod('timeman.timecontrol.setting
s.get', {}, function(result){
        if(result.error())
        {

console.error(result.error().ex);
        }
        else
        {
                console.log(result.data());
        }
});
```

PHP

```
$result =
restCommand('timeman.timecontrol.settings.ge
t', Array(), $_REQUEST["auth"]);
```

Пример ответа

```
{
        "result":{
                "active":true,
                "minimum_idle_for_report":1,
                "register_offline":true,
                "register_idle":true,
                "register_desktop":true,
                "report_request_type":"all",
                "report_request_users":[],
                "report_simple_type":"all",
                "report_simple_users":[],
                "report_full_type":"all",
                "report_full_users":[]
        }
}
```

Описание ключей

- **active** - доступность инструмента контроля времени.
- **minimum_idle_for_report** - минимальное кол-во времени для запроса отчета (в минутах).
- **register_offline** - фиксировать факт перехода пользователя в режим офлайн.
- **register_idle** - фиксировать факт перехода пользователя в режим отошел.

- **register_desktop** - фиксировать факт включения и отключения десктоп приложения.
- **report_request_type** - у кого запрашивать отчет (`all` - у всех, `user` - только у указанных пользователей, `none` - ни у кого).
- **report_request_users** - список пользователей у кого запрашивать отчет (если `report_request_type == user`).
- **report_simple_type** - кому доступен упрощенный отчет (`all` - всем, `user` - только указанным пользователям).
- **report_simple_users** - список пользователей кому доступен упрощенный отчет (если `report_simple_type == user`).
- **report_full_type** - кому доступен расширенный отчет (`all` - всем, `user` - только указанным пользователям).
- **report_full_users** - список пользователей кому доступен расширенный отчет (если `report_simple_type == user`).

Пример ответа при возникновении ошибки

```
{
    "error": "ACCESS_ERROR",
    "error_description": "You don't have
access to user this method"
}
```

- Ключ **error** - код возникшей ошибки.
- Ключ **error_description** - краткое описание возникшей ошибки.

Возможные коды ошибок

Код	Описание
ACCESS_ERROR	Указанный метод доступен только администраторам.

timeman.timecontrol.settings.s

Описание и параметры

Метод для установки настроек инструмента контроля времени.

Параметры

Параметр	По умолчанию	Обязательны
ACTIVE	false	Нет
MINIMUM_IDLE_FOR_REPORT	15	Нет

REGISTER_OFFLINE	true	Нет
REGISTER_IDLE	true	Нет
REGISTER_DESKTOP	true	Нет
REPORT_REQUEST_TYPE	none	Нет
REPORT_REQUEST_USERS	[]	Нет*
REPORT_SIMPLE_TYPE	all	Нет
REPORT_SIMPLE_USERS	[]	Нет*

REPORT_FULL_TYPE	user	Нет
REPORT_FULL_USERS	[]	Нет*

* - если вы передаете параметр `REPORT_REQUEST_TYPE = user` (или `REPORT_SIMPLE_TYPE = user`, или `REPORT_FULL_TYPE = user`), вы обязательно должны передать соответственно `REPORT_REQUEST_USERS` (или `REPORT_SIMPLE_USERS`, или `REPORT_FULL_USERS`).

Пример вызова

JavaScript

```
BX24.callMethod('timeman.timecontrol.setting
s.set', {
        active: true,
        report_request_type: 'user',
        report_request_users: [1,2,3],
}, function(result){
        if(result.error())
        {

console.error(result.error().ex);
        }
        else
        {
                console.log(result.data());
```

```
        }
});
```

PHP

```
$result =
restCommand('timeman.timecontrol.settings.se
t', Array(
        active: true,
        report_request_type: 'user',
        report_request_users: [1,2,3],
), $_REQUEST["auth"]);
```

Пример ответа

```
{
        "result": true
}
```

Пример ответа при возникновения ошибки

```
{
    "error": "ACCESS_ERROR",
    "error_description": "You don't have
access to user this method"
}
```

- Ключ **error** - код возникшей ошибки.
- Ключ **error_description** - краткое описание возникшей ошибки.

Возможные коды ошибок

Код	Описание
ACCESS_ERROR	Указанный метод доступен только администраторам.
INVALID_FORMAT	Передан некорректный формат в поле `RANGE`.

© «Битрикс», 2001-2008, «1С-

timeman.schedule.get

Метод позволяет получить рабочий график по его идентификатору.

Параметры

Параметр	Описание	С версии
id	Идентификатор графика	

Пример

```
BX24.callMethod(
   "timeman.schedule.get",
   {
       id: 2
   },
   function(result)
   {
       if(result.error())
           console.error(result.error());
       else
       {
           console.dir(result.data());
           if(result.more())
```

```
                    result.next();
            }
        }
    );
```

entity.add

Создает хранилище данных. Перед созданием проверить наличие хранилища можно с помощью entity.get

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор хранилища, уникальны для данного приложения (максимальная длина - 13 символов).
NAME	Обязательный. Название хранилища
ACCESS	Описание прав доступа к хранилищу. Должно иметь вид ассоциативного массива, ключами которого являются идентификаторы прав доступа, значением - **R** (чтение), **W** (запись) или **X** (управление).

Создатель хранилища автоматически получает право **X**.

Пример

```
BX24.callMethod('entity.add', {'ENTITY':
'dish', 'NAME': 'Dishes', 'ACCESS':
{U1:'W',AU:'R'}});
```

entity.update

Обновляет параметры хранилища данных. Пользователь должен обладать правами на управление (**X**) хранилищем. Пользователь не может отнять у себя права на управление.

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор обновляемого хранилища.
NAME	Новое название хранилища.
ACCESS	Описание нового набора прав доступа к хранилищу. Должно иметь вид ассоциативного массива, ключами которого являются идентификаторы прав доступа, значением - **R** (чтение), **W** (запись) или **X** (управление).
ENTITY_NEW	Новый строковой идентификатор хранилища.

Пример

```
BX24.callMethod('entity.update', {'ENTITY':
'dish', 'ACCESS': {U1:'W',AU:'R'}});
```

entity.rights

Получение или изменение прав доступа к хранилищу. Возвращает текущий набор прав доступа.

Для изменения набора прав достуа пользователь должен обладать правами на управление (**X**) хранилищем. Пользователь не может отнять у себя права на управление.

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор обновляемого хранилища.
ACCESS	Описание нового набора прав доступа к хранилищу. Должно иметь вид ассоциативного массива, ключами которого являются идентификаторы прав доступа, значением - **R** (чтение), **W** (запись) или **X** (управление).

Пример

```
BX24.callMethod('entity.rights', {'ENTITY':
'dish'});
```

Ответ


```
{"result":
{"AU":"R","U254":"W","D115":"W","U255":"W","U260":"W"}}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

entity.get

Получение параметров хранилища или списка всех хранилищ приложения.

Параметры

Параметр	Описание
ENTITY	Строковой идентификатор требуемого хранилища.

Примеры

```
BX24.callMethod('entity.get');
```

Запрос

```
https://my.bitrix24.ru/rest/entity.get.json?
auth=59efe32d01c0e9dc5732e8dfa68a4baa
```

Ответ

```
{"result":[{"ENTITY":"dish","NAME":"Dishes"},
{"ENTITY":"menu","NAME":"Menu"}]}
```

Пример корректного получения списка всех доступных хранилищ:

```
BX24.callMethod(
         "entity.get",
         {},
         function(result)
         {
            if(result.error())

console.error(result.error());
            else
            {
               console.info("Список
созданных хранилищ:", result.data());
            }
         }
      );
```

entity.delete

Удаление хранилища. Пользователь должен обладать правами на управление (**X**) хранилищем.

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор удаляемого хранилища.

Пример

```
BX24.callMethod('entity.delete', {'ENTITY':
'test'});
```

Ответ

```
{"result":true}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

entity.section.get

Описание и пример

Получение списка разделов хранилища (секций инфоблока). Списочный метод.

Пользователь должен обладать хотя бы правами на чтение (**R**) хранилища.

Пример

Вызов

```
BX24.callMethod('entity.section.get',
{ENTITY: 'menu_new', SORT: {'NAME': 'ASC'}},
function(result){
    sections = result.data();
});
```

Запрос

```
https://my.bitrix24.ru/rest/entity.section.get.json?
ENTITY=menu_new&SORT%5BNAME%5D=ASC&auth=9affe382af74d9c5caa
588e28096e872
```

Ответ


```
{"result":[{"ID":"219","CODE":null,"TIMESTAMP_X":"2013-06-
23T10:11:59+03:00","DATE_CREATE":"2013-06-
23T10:11:59+03:00","CREATED_BY":"1","MODIFIED_BY":"1","ACTI
```

VE":"Y","SORT":"500","NAME":"\u0412\u0442\u043e\u0440\u0430
\u044f \u0442\u0435\u0441\u0442\u043e\u0432\u0430\u044f
\u0441\u0435\u043a\u0446\u0438\u044f","PICTURE":null,"DETAI
L_PICTURE":null,"DESCRIPTION":null,"LEFT_MARGIN":"1","RIGHT
_MARGIN":"2","DEPTH_LEVEL":"1","ENTITY":"menu_new","SECTION
":null},{"ID":"218","CODE":null,"TIMESTAMP_X":"2013-06-
23T10:24:46+03:00","DATE_CREATE":"2013-06-
23T10:08:54+03:00","CREATED_BY":"1","MODIFIED_BY":"1","ACTI
VE":"Y","SORT":"500","NAME":"\u041f\u0435\u0440\u0432\u0430
\u044f \u0442\u0435\u0441\u0442\u043e\u0432\u0430\u044f
\u0441\u0435\u043a\u0446\u0438\u044f","PICTURE":null,"DETAI
L_PICTURE":null,"DESCRIPTION":null,"LEFT_MARGIN":"3","RIGHT
_MARGIN":"4","DEPTH_LEVEL":"1","ENTITY":"menu_new","SECTION
":null}],"total":2}

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор хранилища.
SORT	Массив для сортировки, имеющий вид *by1*=>*order1*[, *by2*=>*order2* [, ..]], где *by1, ...* - поле сортировки, может принимать значения: ▪ **ID** - код раздела; ▪ **SECTION** - код родительской раздела; ▪ **NAME** - название раздела; ▪ **CODE** - символьный код раздела; ▪ **ACTIVE** - активности раздела ▪ **LEFT_MARGIN** - левая граница; ▪ **DEPTH_LEVEL** - глубина вложенности (начинается с 1); ▪ **SORT** - индекс сортировки; ▪ **CREATED** - по времени создания раздела; ▪ **CREATED_BY** - по идентификатору создателя раздела; ▪ **MODIFIED_BY** - по идентификатору пользователя изменившего раздела;

	▪ **TIMESTAMP_X** - по времени последнего изменения. *order1, ...* - порядок сортировки, может принимать значения: ▪ **ASC** - по возрастанию; ▪ **DESC** - по убыванию. Значение по умолчанию Array("SORT"=>"ASC") означает, что результат выборки будет отсортирован по возрастанию. Если задать пустой массив Array(), то результат отсортирован не будет.
FILTER	Массив вида array("фильтруемое поле"=>"значение" [, ...]). *Фильтруемое поле* может принимать значения: ▪ **ACTIVE** - фильтр по активности (Y|N); ▪ **NAME** - по названию (можно искать по шаблону [%_]); ▪ **CODE** - по символьному коду (по шаблону [%_]); ▪ **SECTION_ID** - по коду раздела-родителя (если указать false, то будут возвращены корневые разделы); ▪ **DEPTH_LEVEL** - по уровню вложенности (начинается с 1); ▪ **LEFT_MARGIN**, **RIGHT_MARGIN** - по положению в дереве (используется, когда необходима выборка дерева подразделов; ▪ **ID** - по коду раздела; ▪ **TIMESTAMP_X** - по времени последнего изменения; ▪ **DATE_CREATE** - по времени создания; ▪ **MODIFIED_BY** - по коду пользователя изменившему раздел; ▪ **CREATED_BY** - по содателю; Все фильтруемые поля могут содержать перед названием тип проверки фильтра.

	Необязательное. По умолчанию записи не фильтруются.

entity.section.add

Описание и пример

Добавление раздела хранилища. Пользователь должен обладать хотя бы правами на запись (**W**) в хранилище.

Пример

Вызов

```
BX24.callMethod('entity.section.add',
{ENTITY: 'menu_new', 'NAME': 'Тестовый
раздел'});
```

Запрос

```
https://my.bitrix24.ru/rest/entity.section.add.json?
ENTITY=menu_new&NAME=%D0%A2%D0%B5%D1%81%D1%82%D0%BE%D0%B2%D
1%8B%D0%B9%20%D1%80%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB&auth=9aff
e382af74d9c5caa588e28096e872
```

Ответ

```
{"result":220}
```

Параметры

Параметр	Описание

ENTITY	Обязательный. Строковой идентификатор хранилища.
NAME	Обязательный. Наименование раздела.
DESCRIPTION	Описание раздела.
ACTIVE	Флаг активности раздела (Y|N).
SORT	Сортировочный параметр раздела.
PICTURE	Картинка раздела.
DETAIL_PICTURE	Детальная картинка раздела.
SECTION	Идентификатор родительского раздела.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

entity.section.update

Описание и пример

Обновление раздела хранилища. Пользователь должен обладать хотя бы правами на запись (**W**) в хранилище.

Пример

Вызов

```
BX24.callMethod('entity.section.update',
{ENTITY: 'menu_new', ID: 220, NAME: 'Не
очень тестовый раздел'});
```

Запрос

```
https://my.bitrix24.ru/rest/entity.section.update.json?
auth=9affe382af74d9c5caa588e28096e872&ENTITY=menu_new&ID=22
0&NAME=%D0%9D%D0%B5%20%D0%BE%D1%87%D0%B5%D0%BD%D1%8C%20%D1%
82%D0%B5%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B9%20%D1%80%D0%B0
%D0%B7%D0%B4%D0%B5%D0%BB
```

Ответ

```
{"result":true}
```

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор хранилища.
ID	Обязательный. Идентификатор обновляемого раздела.
NAME	Наименование раздела.
DESCRIPTION	Описание раздела.
ACTIVE	Флаг активности раздела (Y|N).
SORT	Сортировочный параметр раздела.
PICTURE	Картинка раздела.
DETAIL_PICTURE	Детальная картинка раздела.
SECTION	Идентификатор родительского раздела.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

entity.section.delete

Удаление раздела хранилища. Пользователь должен обладать хотя бы правами на запись (**W**) в хранилище.

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор хранилища.
ID	Обязательный. Идентификатор удаляемого раздела.

Пример

Вызов

```
BX24.callMethod('entity.section.delete',
{ENTITY: 'menu_new', ID: 220});
```

Запрос

```
https://my.bitrix24.ru/rest/entity.section.delete.json?
ENTITY=menu_new&ID=220&auth=9affe382af74d9c5caa588e28096e87
2
```

Ответ

```
{"result":true}
```

© «Битрикс», 2001-2008, «1С-

entity.item.get

Описание

Получение списка элементов хранилища. Списочный метод.

Пользователь должен обладать хотя бы правами на чтение (**R**) хранилища.

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор хранилища.
SORT	Аналогичны параметрам *arOrder* и *arFilter* PHP-метода CIBlockElement::GetList (включая операции фильтра и сложную логику).
FILTER	

Примеры

Вызов

```
BX24.callMethod('entity.item.get', {
        ENTITY: 'menu',
        SORT: {DATE_ACTIVE_FROM: 'ASC', ID:
'ASC'},
        FILTER: {
                '>=DATE_ACTIVE_FROM':
```

```
dateStart,
                '<DATE_ACTIVE_FROM':
dateFinish
        }
}, $.proxy(this.buildData, this));
```

Запрос

```
https://my.bitrix24.ru/rest/entity.item.get.json?
=&ENTITY=menu&FILTER%5B%3CDATE_ACTIVE_FROM%5D=2013-07-
01T00%3A00%3A00.000Z&FILTER%5B%3E%3DDATE_ACTIVE_FROM%5D=201
3-06-
24T00%3A00%3A00.000Z&SORT%5BDATE_ACTIVE_FROM%5D=ASC&SORT%5B
ID%5D=ASC&auth=723867cdb1ada1de7870de8b0e558679
```

Ответ


```
{"result":[{"ID":"838","TIMESTAMP_X":"2013-06-
25T15:06:47+03:00","MODIFIED_BY":"1","DATE_CREATE":"2013-
06-
25T15:06:47+03:00","CREATED_BY":"1","ACTIVE":"Y","DATE_ACTI
VE_FROM":"2013-07-
01T03:00:00+03:00","DATE_ACTIVE_TO":"","SORT":"500","NAME":
"\u0413\u0440\u0435\u0447\u043a\u0430 \u0432
\u043c\u0443\u043d\u0434\u0438\u0440\u0435","PREVIEW_PICTUR
E":null,"PREVIEW_TEXT":null,"DETAIL_PICTURE":null,"DETAIL_T
EXT":null,"CODE":null,"ENTITY":"menu","SECTION":null,"PROPE
RTY_VALUES":{"dish":"813","price":"16"}}],"total":1}
```


Пример вызова со сложным фильтром:

```
BX24.callMethod('entity.item.get', {
    ENTITY: 'menu',
    SORT: {DATE_ACTIVE_FROM: 'ASC', ID:
'ASC'},
    FILTER: {
        '1':{
```

```
                'LOGIC':'OR',
                'PROPERTY_MYPROP1':'value1',
                'PROPERTY_MYPROP2':'value2'
                }
        }
});
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

entity.item.add

Описание и пример

Добавление элемента хранилища. Пользователь должен обладать хотя бы правами на запись (**W**) в хранилище.

Пример

Вызов

```
BX24.callMethod('entity.item.add', {
        ENTITY: 'menu_new',
        DATE_ACTIVE_FROM: new Date(),
        DETAIL_PICTURE: '',
        NAME: 'Hello, world!',
        PROPERTY_VALUES: {
                test: 11,
                test1: 22,
                test_file: ''
        },
        SECTION: 219
});
```

Запрос

```
https://my.bitrix24.ru/rest/entity.item.add.json?
DATE_ACTIVE_FROM=2013-06-
26T11%3A54%3A30.421Z&DETAIL_PICTURE=&ENTITY=menu_new&NAME=H
ello%2C%20world!&PROPERTY_VALUES%5Btest1%5D=22&PROPERTY_VAL
```

```
UES%5Btest%5D=11&PROPERTY_VALUES%5Btest_file%5D=&SECTION=21
9&auth=9affe382af74d9c5caa588e28096e872
```

Ответ

```
{"result":842}
```

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор хранилища.
NAME	Обязательный. Наименование элемента.
ACTIVE	Флаг активности элемента (Y|N).
DATE_ACTIVE_FROM	Дата начала активности элемента.
DATE_ACTIVE_TO	Дата окончания активности элемента.
SORT	Сортировочный вес элемента.
PREVIEW_PICTURE	Картинка анонса элемента.
PREVIEW_TEXT	Анонс элемента.
DETAIL_PICTURE	Детальная картинка элемента.
DETAIL_TEXT	Детальный текст элемента.
CODE	Символьный код элемента.
SECTION	Идентификатор раздела хранилища.
PROPERTY_VALUES	Ассоциативный список значений свойств элемента. Свойства хранилища создаются при помощи entity.item.property.add.

702455

entity.item.update

Описание и пример

Обновление элемента хранилища. Пользователь должен обладать хотя бы правами на запись (**W**) в хранилище.

Пример

Вызов

```
BX24.callMethod('entity.item.update', {
        ENTITY: 'menu_new',
        ID: 842,
        DATE_ACTIVE_FROM: new Date(),
        DETAIL_PICTURE: '',
        NAME: 'Goodbye Cruel World',
        PROPERTY_VALUES: {
                test: 11,
                test1: 22,
                test_file: ''
        },
        SECTION: 219
});
```

Запрос

```
https://my.bitrix24.ru/rest/entity.item.update.json?
DATE_ACTIVE_FROM=2013-06-
26T12%3A03%3A31.653Z&DETAIL_PICTURE=&ENTITY=menu_new&ID=842
```

```
&NAME=Goodbye%20Cruel%20World&PROPERTY_VALUES%5Btest1%5D=22
&PROPERTY_VALUES%5Btest%5D=11&PROPERTY_VALUES%5Btest_file%5
D=&SECTION=219&auth=9affe382af74d9c5caa588e28096e872
```

Ответ

```
{"result":true}
```

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор хранилища.
ID	Обязательный. Идентификатор элемента.
NAME	Наименование элемента.
ACTIVE	Флаг активности элемента (Y|N).
DATE_ACTIVE_FROM	Дата начала активности элемента.
DATE_ACTIVE_TO	Дата окончания активности элемента.
SORT	Сортировочный вес элемента.
PREVIEW_PICTURE	Картинка анонса элемента.
PREVIEW_TEXT	Анонс элемента.
DETAIL_PICTURE	Детальная картинка элемента.
DETAIL_TEXT	Детальный текст элемента.
CODE	Символьный код элемента.
SECTION	Идентификатор раздела хранилища.

PROPERTY_VALUES	Обязательный. ассоциативный список значений свойств элемента. Свойства хранилища создаются при помощи entity.item.property.add.

entity.item.delete

Удаление элемента хранилища. Пользователь должен обладать хотя бы правами на запись (**W**) в хранилище.

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор хранилища.
ID	Обязательный. Идентификатор элемента.

Пример

Вызов

```
BX24.callMethod('entity.item.delete', {
        ENTITY: 'menu_new',
        ID: 842
});
```

Запрос

```
https://my.bitrix24.ru/rest/entity.item.delete.json?
ENTITY=menu_new&ID=842&auth=340bf57f35ee95e0debf98399632999
c
```

Ответ

```
{"result":true}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

entity.item.property.get

Получение списка дополнительных свойств элементов хранилища.

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор хранилища.
PROPERTY	Строковой идентификатор требуемого свойства.

Пример

Вызов

```
BX24.callMethod('entity.item.property.get',
{ENTITY: 'menu_new'}, function(r){
        console.log(r.data());
});
```

Запрос

```
https://my.bitrix24.ru/rest/entity.item.property.get.json?
ENTITY=menu_new&auth=340bf57f35ee95e0debf98399632999c
```

Ответ


```
{"result":
[{"PROPERTY":"test","NAME":"\u0422\u0435\u0441\u0442\u043e\
u0432\u043e\u0435
\u0441\u0432\u043e\u0439\u0441\u0442\u0432\u043e","TYPE":"S
"},
{"PROPERTY":"test1","NAME":"\u0412\u0442\u043e\u0440\u043e\
u0435 \u0442\u0435\u0441\u0442\u043e\u0432\u043e\u0435
\u0441\u0432\u043e\u0439\u0441\u0442\u0432\u043e","TYPE":"N
"},
{"PROPERTY":"test_file","NAME":"\u0424\u0430\u0439\u043b\u0
43e","TYPE":"F"}]}
```


© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

entity.item.property.add

Добавление дополнительного свойства элементов хранилища. Пользователь должен обладать правами на управление (**X**) хранилищем.

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор хранилища.
PROPERTY	Обязательный. Строковой идентификатор свойства.
NAME	Обязательный. Наименование свойства.
TYPE	Обязательный. Тип свойства (**S** - строка, **N** - число, **F** - файл).

Пример

Вызов

```
BX24.callMethod('entity.item.property.add',
{ENTITY: 'menu_new', PROPERTY: 'new_prop',
NAME: 'Новое свойство', TYPE: 'S'});
```

Запрос

```
https://my.bitrix24.ru/rest/entity.item.property.add.json?
ENTITY=menu_new&NAME=%D0%9D%D0%BE%D0%B2%D0%BE%D0%B5%20%D1%8
1%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE&PROPERTY=new_pr
op&TYPE=S&auth=e690b44d2b3827d2eb9d4dbe59406dbb
```

Ответ

```
{"result":true}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

entity.item.property.update

Обновление дополнительного свойства элементов хранилища. Пользователь должен обладать правами на управление (**X**) хранилищем.

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор хранилища.
PROPERTY	Обязательный. Строковой идентификатор свойства.
PROPERTY_NEW	Новый строковой идентификатор свойства.
NAME	Наименование свойства.
TYPE	Тип свойства (**S** - строка, **N** - число, **F** - файл).

Пример

Вызов

```
BX24.callMethod('entity.item.property.update
', {ENTITY: 'menu_new', PROPERTY:
'new_prop', NAME: 'Уже не новое свойство'});
```

Запрос

```
https://my.bitrix24.ru/rest/entity.item.property.update.jso
n?
ENTITY=menu_new&NAME=%D0%A3%D0%B6%D0%B5%20%D0%BD%D0%B5%20%D
0%BD%D0%BE%D0%B2%D0%BE%D0%B5%20%D1%81%D0%B2%D0%BE%D0%B9%D1%
81%D1%82%D0%B2%D0%BE&PROPERTY=new_prop&auth=ad5a6f34f14f644
136830eb8a936f07f
```

Ответ

```
{"result":true}
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

entity.item.property.delete

Удаление дополнительного свойства элементов хранилища. Пользователь должен обладать правами на управление (**X**) хранилищем.

Параметры

Параметр	Описание
ENTITY	Обязательный. Строковой идентификатор хранилища.
PROPERTY	Обязательный. Строковой идентификатор свойства.

Пример

Вызов

```
BX24.callMethod('entity.item.property.delete
', {ENTITY: 'menu_new', PROPERTY:
'new_prop'});
```

Запрос

```
https://my.bitrix24.ru/rest/entity.item.property.delete.jso
n?
```

```
ENTITY=menu_new&PROPERTY=new_prop&auth=d92dd12b9b9b90425477
6104eed2bb76
```

Ответ

```
{"result":true}
```

© «Битрикс», 2001-2008, «1С-

imopenlines.operator.answer

Метод для приема диалога текущим оператором.

Параметры

Параметр	Описание	С версии
CHAT_ID	Идентификатор чата, на который отвечает текущий оператор.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

imopenlines.operator.finish

Метод для завершения диалога текущим оператором.

Параметры

	Описание	С версии
CHAT_ID	Идентификатор чата, который текущий оператор завершает.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

imopenlines.operator.skip

Метод для пропуска диалога текущим оператором.

Параметры

	Описание	С версии
CHAT_ID	Идентификатор чата, который пропускает текущий оператор.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

imopenlines.operator.spam

Метод для пометки диалога как спам текущим оператором.

Параметры

	Описание	С версии
CHAT_ID	Идентификатор чата, который текущий оператор помечает как спам.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

imopenlines.operator.transfer

Метод для перевода диалога текущим оператором на другого оператора/линию.

Параметры

	Описание	С версии
CHAT_ID	Идентификатор чата, который текущий оператор завершает.	
TRANSFER_ID	Идентификатор сущности, на которую переводится диалог. Если нужно перевести диалог на оператора, в качестве значения передается ID оператора. Если на линию - код вида "`queue#ID линии#`"	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

imopenlines.session.intercept

Метод для того, чтобы текущий оператор забрал диалог у которого уже есть другой оператор

Параметры

	Описание	С версии
CHAT_ID	Идентификатор чата, который текущий оператор забирает.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Сценарий встройки WebRTC

Регистрация встройки

Регистрация встройки для «клиента» на каждой странице.

Имеется специальное место встройки, которое в виде невидимого фрейма формируется на каждой странице **Битрикс24**. Регистрация делается так:

```
'placement.bind',
[
    'PLACEMENT' => 'PAGE_BACKGROUND_WORKER',
    'HANDLER' =>
'http://example.com/placement/?ty=1',
    'OPTIONS' => [
        'errorHandlerUrl' =>
'http://example.com/logg.php?ty=1',
    ],
    'LANG_ALL' => [
        'ru' => [
            'TITLE' => 'test',
        ]
    ]
]
```

Важное отличие от обычной встройки – параметр `Options[errorHandlerUrl]`. В этот обработчик передается сигнал,

что мы отключаем встройку на конкретном портале **Битрикс24**, в случае, если обработчик, указанный в `Handler`, отвечает слишком медленно. Поскольку встройка формируется на каждой странице, важно, чтобы обработчик встройки вызывался быстро (сейчас считаем, что «не быстро» - это когда обработчик вызывался дольше 5 сек три раза). В случае отключения нужно будет заново зарегистрировать обработчик в этом **Битрикс24**.

Сценарий использования

Работа с телефонией остается той же, что и была. Регистрация звонка осуществляется методом telephony.externalcall.register. Этот же метод «поднимает» карточку звонка. Очевидно, что это должно происходить, если WebRTC-клиент во встройке, описанной выше, начал обработку звонка.

Далее, встройка может взаимодействовать с открытой карточкой звонка, управляя кнопками и событиями нажатия на кнопки. Для работы с карточкой звонка через плейсмент PAGE_BACKGROUND_WORKER было добавлено 9 методов для получения и изменения данных карточки и 17 событий для обработки пользовательской деятельности.

Ключевым является событие BackgroundCallCard::initialized. Оно выбрасывается при создании карточки звонка и после него становится возможным управлять данной карточкой. Поэтому настоятельно рекомендуется все вызовы методов со стороны приложения производить именно в функции-обработчике данного события.

Свойства отгрузки

Методы работы со свойствами отгрузки:

Метод	Описание
sale.shipmentproperty.add	Добавляет свойство отгрузки.
sale.shipmentproperty.delete	Удаляет свойство отгрузки.
sale.shipmentproperty.get	Метод для доступа к полям и настройкам свойства отгрузки.
sale.shipmentproperty.getFieldsByType	Возвращает поля и настройки свойства отгрузки для определенного типа свойства.
sale.shipmentproperty.list	Получает список свойств отгрузки.
sale.shipmentproperty.update	Метод для обновления полей свойства отгрузки.

Примеры работы со свойствами отгрузки смотрите в уроке Процесс создания и настройки службы доставки.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Импорт

Импорт через REST поддерживает все основные сущности CRM (идентификаторы типов можно найти тут):

- Лиды
- Сделки
- Контакты
- Компании
- Коммерческие предложения
- Новые счета
- Смарт-процессы

Для импорта доступны два метода:

- `crm.item.import` - Импорт одной записи
- `crm.item.batchImport` - Групповой импорт записей

Добавление элементов через импорт, в отличие от добавления через `crm.*.add`, имеет следующие особенности:

- Проверка прав производится на импорт, а не на добавление элементов
- Роботы и бизнес-процессы не будут запущены на добавленном элементе
- При выполнении под администратором портала, есть возможность установить значения некоторых системных полей:

Поле	Описание
createdTime	Дата и время создания записи
updatedTime	Дата и время обновления записи

movedTime	Дата и время изменения стадии (если сущность поддерживает стадии)
createdBy	Пользователь, создавший запись
updatedBy	Пользователь, изменивший запись
movedBy	Пользователь, изменивший стадию (если сущность поддерживает стадии)

На значения этих полей накладываются некоторые ограничения:

- Требуется монотонное возрастание значения поля `createdTime`. То есть нельзя будет создавать записи с датой создания меньшей, чем у какой-либо из уже существующих записей.
- `createdTime` и `updatedTime` не могут быть в будущем
- `createdTime` не может быть меньше `updatedTime`
- `movedTime` должно быть в диапазоне между `createdTime` и `updatedTime`

Управление привязками дел к сущностям CRM

Метод	Описание
crm.activity.binding.list	Получить список привязок
crm.activity.binding.add	Добавление привязки
crm.activity.binding.delete	Удаление привязки

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Счета (новые)

Новые счета — это отдельный тип сущности, заменяющий собой существовавшие до этого счета.

Как смарт-процесс

Новые счета — это зафиксированный тип смарт-процесса, за исключением деталей. Поэтому те же самые методы, которые применяются при работе со смарт-процессами, используются и с новыми счетами. Отличие лишь в значениях входных параметров.

Многие методы требуют указать `entityTypeId`. В качестве значения для этого параметра необходимо передать `entityTypeId` для новых счетов. Значение `entityTypeId` для новых счетов, а так же других специфичных для типа сущности констант, можно посмотреть списке констант.

Так же значения `entityTypeId`, `entityTypeName` и `ownerType` можно получить через метод `crm.enum.ownertype` (см. поля `ID`, `SYMBOL_CODE`, `SYMBOL_CODE_SHORT`).

crm.type.*

Через методы `crm.type.*` прочитать/изменить/удалить новые счета **невозможно**. При попытке это сделать будет возвращена ошибка.

Управление элементами новых счетов - crm.item.*

Через методы crm.item.* можно читать/изменять/удалять элементы новых счетов. Работа с новыми счетами через них ничем не отличается от того, что описано в документации. В качестве параметра entityTypeId необходимо передать значение, актуальное для новых счетов.

Управление товарными позициями - crm.item.productrow.*

Через методы crm.item.productrow.* можно управлять товарными позициями, привязанными к счету. Работа с ними аналогична таковой для элементов смарт-процесса. В качестве параметра ownerType необходимо передавать краткий символьный код для новых счетов (SI).

Управление стадиями - crm.status.*

Через методы crm.status.* можно управлять стадиями новых счетов. Значение поля ENTITY_ID и STATUS_ID формируется по тому же принципу, что и для смарт-процессов.

Чтобы получить значение готовое ENTITY_ID для новых счетов, можно воспользоваться методом crm.status.entity.types и найти там подмассив, у которого ENTITY_TYPE_ID совпадает с entityTypeId для новых счетов. Под ключом ID будет нужное значение, которое можно подставлять как ENTITY_ID в другие методы.

crm.category.*

Так как новые счета основаны на смарт-процессах, то они имеют техническую возможность работать с направлениями. Однако **не рекомендуется** использовать методы `crm.category.*` для изменения направлений новых счетов, так как направления в них не используются. В интерфейсе любые изменения отображены не будут.

Настройки карточки счета - crm.item.details.configuration.*

Через методы `crm.item.details.configuration.*` можно управлять настройками карточки нового счета. В качестве `entityTypeId` необходимо передавать актуальное для нового счета значение.

События над элементами нового счета

При добавлении/изменении/удалении элемента нового счета, будут выбрасываться события, идентичные таковым у смарт-процесса. В качестве значения для `ENTITY_TYPE_ID` будет передаваться идентификатор CRM нового счета (`entityTypeId`).

Встройки в интерфейс

Коды встроек генерируются таким же образом, как и для других сущностей, но в качестве строкового названия используется `SMART_INVOICE`

- `CRM_SMART_INVOICE_LIST_TOOLBAR` - кнопка в тулбаре в списке счетов
- `CRM_SMART_INVOICE_DETAIL_TOOLBAR` - кнопка в тулбаре в карточке счета
- `CRM_SMART_INVOICE_DOCUMENTGENERATOR_BUTTON` - пункт меню в кнопке "Документ" в карточке счета

... и другие

Скрам

Описанный в данном разделе [ds]Скрам[/ds][di] Скрам (англ. scrum «схватка») — методология управления проектами.

Подробнее...[/di] технически является проектом *Битрикс24*. Под "id Скрама" подразумевается id проекта/группы и передаётся в параметрах/полях как **groupId**.

Для создания Скрама можно воспользоваться методом для рабочих групп соцсети sonet_group.create, заполнив поле `SCRUM_MASTER_ID`.

Пространства имен для rest-методов работы со Скрамом:

Пространство	Описание
Бэклог	Методы для работы с бэклогом.
Спринты	Методы для работы со спринтами.
Эпики	Методы для работы с эпиками.
Задачи Скрама	Методы для работы с задачами Скрама.
Канбан	Методы для работы с канбаном.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Складской учёт

Методы работы со складским учётом:

Метод	Описание
catalog.document.add	Метод для добавления документа складского учёта.
catalog.document.confirm	Метод для проведения документа.
catalog.document.delete	Метод для удаления документа.
catalog.document.fields	Метод возвращает список полей документов.
catalog.document.list	Метод для получения списка документов.
catalog.document.mode.status	Метод для получения информации о том, включен ли складской учёт.
catalog.document.unconfirm	Метод для отмены проведения документа.
catalog.document.update	Метод для обновления документа складского учёта.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Товары документа складского учёта

Методы работы с товарами документа складского учёта:

Метод	Описание
catalog.document.element.add	Метод для добавления товара документа складского учёта.
catalog.document.element.delete	Метод для удаления товара документа складского учёта.
catalog.document.element.fields	Метод возвращает список полей товаров документа складского учёта.
catalog.document.element.list	Метод для получения списка товаров в документах складского учёта.
catalog.document.element.update	Метод для обновления товара документа складского учёта.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Склад

Методы работы со складами:

Метод	Описание
catalog.store.add	Метод для добавления склада.
catalog.store.delete	Метод для удаления склада.
catalog.store.get	Метод для получения значений полей склада по ID.
catalog.store.getFields	Метод возвращает поля склада.
catalog.store.list	Метод получает список складов по фильтру.
catalog.store.update	Метод для обновления склада.

© «Битрикс», 2001-2008, «1С-

Товарные позиции

Описание

REST-методы из семейства **crm.item.productrow.*** позволяют работать с товарными позициями, привязанными к различным элементам CRM. Методы являются универсальными и поддерживают любой тип владельца, который может работать с новым API CRM.

ВАЖНО! На данный момент полностью поддерживается работа только с товарными позициями, привязанными к смарт-процессу или предложению, так как только эти типы сущностей сейчас работают в новом API. При попытке воспользоваться методами для других типов владельцев будет возвращена ошибка.

Права доступа

При обращении к методам REST учитываются права доступа пользователя, от которого осуществляется вызов методов. Товарные позиции не являются самостоятельной сущностью CRM и всегда привязаны к какому-либо элементу, выступающему как их владелец. Поэтому при всех операциях проверяются права пользователя на доступ и управление элементом-владельцем (предложением, смарт-процессом и т.п.). Например, если у пользователя нет доступа к чтению какого-либо элемента, то и прочитать его товарные позиции он не сможет.

Автоматические действия после любого изменения

После любого изменения, вносимого в товарные позиции, будут выполнены все стандартные проверки и процедуры, происходящие при изменении элемента CRM, в том числе пересчет суммы и запуск роботов после сохранения. Это относится ко всем методам, создающим/изменяющим товарные позиции: crm.item.productrow.add, crm.item.productrow.update, crm.item.productrow.set.

Булевы значения

Значения некоторых полей имеют булевый тип (например, поле **taxIncluded**). В этом случае для изменения значения надо передавать "Y" или "N". В ответах на запросы они будут отображаться аналогично.

Методы

Методы работы с товарными позициями **crm.item.productrow.***.

Метод	Описание	С версии
crm.item.productrow.fields	Получение информации о полях товарных позиций	
crm.item.productrow.get	Получение информацию о товарной позиции с идентификатором id.	
crm.item.productrow.add	Метод создает новую товарную	

	позицию.	
crm.item.productrow.update	Метод обновит товарную позицию с идентификатором id	
crm.item.productrow.set	Метод привяжет товарные позиции к элементу CRM.	
crm.item.productrow.delete	Метод удалит товарную позицию.	
crm.item.productrow.list	Фильтрация списка товарных позиций.	

События

С версии CRM 21.900.0 доступно три вида событий:

- `onCrmDocumentGeneratorDocumentAdd` - создание нового документа.
- `onCrmDocumentGeneratorDocumentUpdate` - изменение документа.
- `onCrmDocumentGeneratorDocumentDelete` - удаление документа.

В обработчик события придут данные в следующем виде:

```
[
    'FIELDS' => [
        'ID' => $documentId,
        'ENTITY_TYPE_ID' => $entityTypeId,
        'ENTITY_ID' => $entityId,
    ],
]
```

где

- `$documentId` - идентификатор документа.
- `$entityTypeId` - идентификатор типа CRM.
- `$entityId` - идентификатор элемента.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Общее

Каждый **смарт-процесс** - это новая сущность в рамках CRM, для которой создается новый раздел со своим интерфейсом, набором функционала, полями, элементами и т.д. Подробнее можно прочитать в документации.

Порядок работы со смарт-процессам следующий:

1. Создается новый смарт-процесс crm.type.add.
2. При создании смарт-процесса независимо от настроек будет создано направление по умолчанию. При желании его можно изменить/добавить.
3. При создании направления независимо от настроек к нему создается набор стадий по умолчанию. При желании их можно изменить/удалить.
4. Создаются пользовательские поля для этого смарт-процесса.
5. Можно работать с элементами.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Импорт отраслевых решений

Разрешение **Импорт отраслевых решений** (configuration.import).

Метод	Описание	С версии
configuration.import.register	Регистрация импорта.	21.400.0
configuration.import.unregister	Отмена зарегистрированного импорта.	21.400.0
configuration.import.get	Получение информации о текущем шаге импорта.	21.400.0

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

История движения по стадиям

Описание

```
crm.stagehistory.list(
   {
      entityTypeId: number,
      order: ?{} = null,
      filter: ?{} = null,
      select: ?{} = null,
      start: ?number = 0
   }
)
```

Метод поддерживает извлечение записей из истории движения по стадиям для лидов, сделок и счетов.

Параметры

Параметр	Описание	С версии
entityTypeId	Идентификатор типа сущности. Может принимать значения: ▪ **1** - лид	

	▪ **2** - сделка ▪ **5** - счет	
order	Список для сортировки, где ключ - поле, а значение - ASC или DESC	
filter	Список для фильтрации. Фильтр поддерживает использование точных значений, массивов значений, а также модификаторы =, !=, <, >, >=, <=. Поиск по like не поддерживается.	
start	Сдвиг для постраничной навигации. Логика работы с постраничкой стандартная для списочных рест методов.	

Возвращаемое значение

Метод вернет массив записей из истории:

```
{
    "items": []
}
```

Каждый элемент массива - массив с ключами:

- **ID** - идентификатор записи
- **TYPE_ID** - тип записи. Может принимать значения: **1** - создание сущности, **2** - перевод на промежуточную стадию, **3** - перевод на финальную стадию
- **OWNER_ID** - идентификатор сущности, в которой изменилась стадия
- **CREATED_TIME** - дата и время попадания на стадию

Помимо этого, имеются специфичные для разных типов сущностей поля:

- для лидов и счетов это
  - **STATUS_SEMANTIC_ID** - [dw]семантика статуса[/dw]
    [di]**P** - промежуточная стадия,
    S - успешная стадия,
    F - провальная стадия[/di] (стадии).
  - **STATUS_ID** - идентификатор статуса (стадии)

- для сделок это
  - **CATEGORY_ID** - идентификатор направления (воронки)
  - **STAGE_SEMANTIC_ID** - [dw]семантика статуса[/dw]
    [di]**P** - промежуточная стадия,
    S - успешная стадия,
    F - провальная стадия[/di] (стадии).
  - **STAGE_ID** - идентификатор стадии

Пример вызова

Получение истории движения по стадиям для сделки с ID=1

```
BX24.callMethod(
    "crm.stagehistory.list",
    {
        entityTypeId: 2,
        order: { "ID": "ASC" },
        filter: { "OWNER_ID": 1 },
        select: [ "ID", "STAGE_ID",
"CREATED_TIME" ]
    },
    function(result)
    {
        if(result.error())
            console.error(result.error());
        else
        {
```

```
                console.dir(result.data());
                if(result.more())
                    result.next();
            }
        }
);
```

Кассы

Методы работы с кассами и чеками. Разрешение **cashbox**.

Метод	Описание
Методы для работы с обработчиками	
sale.cashbox.handler.add	Добавляет REST-обработчик кассы.
sale.cashbox.handler.delete	Удаляет REST-обработчик кассы.
sale.cashbox.handler.update	Обновляет данные REST-обработчика кассы.
sale.cashbox.handler.list	Метод для получения списка доступных REST-обработчиков касс.
Методы для работы с кассами	
sale.cashbox.add	Добавляет кассу.
sale.cashbox.update	Обновляет существующую кассу.
sale.cashbox.delete	Удаляет кассу.
sale.cashbox.list	Возвращает список настроенных касс.
Методы для работы с чеками	
sale.cashbox.check.apply	Сохраняет результат печати чека, напечатанного на REST-

	кассе.

Пользовательские поля

Метод	Описание	С версии
user.userfield.add	Добавляет пользовательское поле	
user.userfield.update	Обновляет пользовательское поле	
user.userfield.delete	Удаляет пользовательское поле	
user.userfield.list	Получает список пользовательских полей	
user.userfield.file.get	Получает файл из пользовательского поля	

© «Битрикс», 2001-2008, «1С-

Торговый каталог

Пространства имен для rest-методов работы с Торговым каталогом.

Пространство	Описание
Торговый каталог	Методы работы с торговым каталогом.
Перечисления	Методы работы с перечислениями.
Наценка	Методы работы с наценкой.
Единица измерения	Методы работы с единицами измерений.
Цена	Методы работы с ценами.
Тип цены	Методы работы с типами цен.
Товар	Методы работы с товарами.
Коэффициент единицы измерения	Методы работы с коэффициентами единиц измерений.
Правила округления цен	Методы работы с правилами округления цен.
Секция каталога	Методы работы с секциями каталога.
Склад	Методы работы со складами.
Складской учёт	Методы работы со складским учётом.
Налоги	Методы работы с налогами.

© «Битрикс», 2001-2008, «1С-

События разделов товаров

Событие	Вызывается после	С версии
onCrmProductSectionAdd	добавления раздела	
onCrmProductSectionDelete	удаления раздела	
onCrmProductSectionUpdate	изменения раздела	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События единиц измерения

Событие	Вызывается после	С версии
onCrmMeasureAdd	добавления новой единицы измерения на портале.	
onCrmMeasureUpdate	изменения единицы измерения на портале.	
onCrmMeasureDelete	удаления единицы измерения на портале.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

Событие	Вызывается
onCrmRequisiteUserFieldAdd	при добавлении пользовательского поля
onCrmRequisiteUserFieldUpdate	при изменении пользовательского поля
onCrmRequisiteUserFieldDelete	при удалении пользовательского поля
onCrmRequisiteUserFieldSetEnumValues	при изменении набора значений для пользовательского поля списочного типа
onCrmRequisiteAdd	при добавлении реквизита
onCrmRequisiteDelete	при удалении реквизита
onCrmRequisiteUpdate	при обновлении реквизита
onCrmBankDetailAdd	при добавления банковского реквизита
onCrmBankDetailDelete	при удалении банковского реквизита.

onCrmBankDetailUpdate	при обновлении банковского реквизита
onCrmAddressRegister	при регистрации адреса
onCrmAddressUnregister	при удалении адреса

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

Событие	Вызывается
onCrmQuoteAdd	при создании предложения
onCrmQuoteUpdate	при обновлении предложения
onCrmQuoteDelete	при удалении предложения
onCrmQuoteUserFieldAdd	при добавлении пользовательского поля
onCrmQuoteUserFieldDelete	при удалении пользовательского поля
onCrmQuoteUserFieldUpdate	при изменении пользовательского поля
onCrmQuoteUserFieldSetEnumValues	при изменении набора значений для пользовательского поля списочного типа

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Шаблоны Бизнес-процессов

Метод	Описание
bizproc.workflow.template.add	Метод добавляет шаблон Бизнес-процесса.
bizproc.workflow.template.delete	Метод удаляет шаблон Бизнес-процесса.
bizproc.workflow.template.update	Метод изменяет шаблон Бизнес-процесса.
bizproc.workflow.template.list	Метод возвращает список шаблонов Бизнес-процессов, установленных на сайте.

© «Битрикс», 2001-2008, «1С-

Роботы приложений

Методы для работы с роботами приложений.

Метод	Описание
bizproc.robot.add	Метод регистрирует нового робота.
bizproc.robot.delete	Метод удаляет зарегистрированного робота.
bizproc.robot.list	Возвращает список зарегистрированных приложением роботов.
bizproc.robot.update	Метод обновляет поля робота.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Действия приложений

Методы работы с действиями приложений.

Метод	Описание	С версии
bizproc.activity.add	Добавляет новое действие в бизнес-процесс.	
bizproc.activity.delete	Метод удаляет действие.	
bizproc.activity.list	Метод возвращает список установленных приложением действий.	
bizproc.activity.update	Метод позволяет обновить поля действия.	
bizproc.activity.log	Метод записывает информацию в лог бизнес-процесса	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Роли и их права доступа

Методы	Описание
documentgenerator.role.get	Возвращает информацию о роли и её правах доступа по идентификатору.
documentgenerator.role.list	Возвращает список ролей без их прав доступа.
documentgenerator.role.delete	Удаляет роль.
documentgenerator.role.add	Добавляет новую роль.
documentgenerator.role.update	Обновляет существующую роль.
documentgenerator.role.fillaccesses	Устанавливает набор ролей и их привязок.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Задачи

Метод	Описание
tasks.task.add	Создает задачу.
tasks.task.approve	Позволяет принять задачу.
tasks.task.complete	Переводит задачу в статус «завершена».
tasks.task.counters.get	Получает счетчики пользователя.
tasks.task.deger	Переводит задачу в статус «отложена».
tasks.task.delegate	Метод для делегирования задачи.
tasks.task.delete	Удаляет задачу.
tasks.task.disapprove	Позволяет отклонить задачу.
tasks.task.favorite.add	Добавляет задачи в "Избранное".
tasks.task.favorite.delete	Удаляет задачи из "Избранного".
tasks.task.files.attach	Прикрепляет загруженный на диск файл к задаче.
tasks.task.get	Возвращает информацию о конкретной задаче.
tasks.task.getFields	Возвращает все доступные поля.
tasks.task.getaccess	Метод для проверки доступа к

	задаче.
tasks.task.history.list	Получает историю задачи.
tasks.task.list	Возвращает массив задач, каждая из которых содержит массив полей.
tasks.task.pause	Останавливает выполнение задачи, переводя ее в статус "ждет выполнения".
tasks.task.renew	Возобновляет задачу после ее завершения.
tasks.task.start	Переводит задачу в статус «выполняется».
tasks.task.startwatch	Позволяет наблюдать за задачей.
tasks.task.stopwatch	Останавливает наблюдение за задачей.
tasks.task.update	Обновляет задачу.

Частые кейсы

Несколько примеров частых кейсов при работе с Бизнес-процессами.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Пользовательские страны

Методы	Описание
documentgenerator.region.get	Возвращает информацию о регионе по его идентификатору.
documentgenerator.region.list	Возвращает список регионов, как установленных по умолчанию, так и пользовательских.
documentgenerator.region.delete	Удаляет регион.
documentgenerator.region.add	Добавляет новый регион.
documentgenerator.region.update	Обновляет существующий регион.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Генератор документов

Описание модуля Генератор документов приведено в документации по D7.

Архитектурно REST-api большей частью соответствует PHP-api модуля. Сам набор REST-методов покрывает все возможности модуля.

На данный момент есть **два scope** для работы с генератором документов:

- Методы **crm.documentgenerator.***. Результаты работы этих методов отображаются в интерфейсе CRM;
- Методы **documentgenerator.***. Результат работы этих методов доступен только на уровне REST.

Из методов одного scope нельзя получить доступ к данным другого:

- Нельзя создать документ CRM с шаблоном для REST;
- Нельзя использовать данные CRM при работе с методами documentgenerator.*.

Поддерживаются следующие **типы полей** и их **модификаторы**:

- IMAGE - изображения;
- STAMP - печати и подписи;
- DATE - даты;
- NAME - имена.

Типы полей **Деньги** и **Адрес** реализованы внутри модуля *crm*, поэтому использовать их в REST этого модуля не получится. Если надо вывести такие данные - придётся передавать их в уже сформированном виде.

Есть возможность использовать массивы для вставки в таблицы и повторяющиеся блоки.

Отличие параметров методов разных scope

“Изнутри” методы идентичны. По факту методы **crm.documentgenerator.*** после пред-обработки параметров вызывают методы **documentgenerator.***. Но есть ряд отличий:

- На вход методов **crm.documentgenerator.*** необходимо вместо имен провайдеров передавать **ID** типа сущности CRM (параметр `entityTypeId`);
- На вход методов **crm.documentgenerator.*** необходимо вместо параметра **value** передавать параметр `entityId` - **ID** сущности CRM

Шаблоны

Все создаваемые этим api шаблоны и документы привязаны к модулю REST. Через scope `documentgenerator` нельзя обращаться к шаблонам и документам других модулей. Поэтому `moduleId` в данных о шаблоне всегда будет `rest`. Даже если в `add` или `update` указать другой модуль, он не будет изменён.

Для работы REST доступны только два провайдера:

- `Bitrix\DocumentGenerator\DataProvider\Rest` - всегда должен быть указан в качестве провайдера для шаблона
- `Bitrix\DocumentGenerator\DataProvider\HashDataProvider` - используется для передачи данных в таблицы / повторяющиеся блоки

Привязка шаблона к пользователю самими REST-методами никак не учитывается. Но её можно использовать на стороне приложения.

Нумераторы

Для работы с нумераторами есть методы documentgenerator.numerator.*, описанные тут.

Следует учесть, что через данный скоуп есть возможность получить доступ ко всем нумераторам для документов. В том числе к тем, которые работают в CRM. Но через REST обновить / удалить можно только тот нумератор, который был создан через REST.

Список регионов

Каждый шаблон привязан к определенной стране. Список стран фиксирован и на данный момент состоит из:

ru - Россия

by - Беларусь

kz - Казахстан

ua - Украина

br - Бразилия

mx - Мексика

de - Германия

uk - Великобритания

pl - Польша

Начиная с версии documentgenerator 18.6.1 появилась возможность добавлять свои регионы. Для управления ими появился отдельный раздел.

Итоги

Что сделать можно?

- Создать документы на основе шаблонов в формате .docx файлов;
- В шаблон можно вставить списки с произвольным количеством элементов через таблицы или повторяющиеся блоки;
- В шаблон можно вставить изображения, в том числе из списков;
- Вставить поля в виде html с частичным сохранением форматирования;
- Создать документы, отправить их и отследить просмотр без участия пользователя (через роботов).

Что сделать нельзя?

- Вставить множественное значение поля типа «файл»;
- Вставить таблицы и изображения из html;
- Вставить векторные изображения;
- Передача форматирование выполняется не полностью.

Отгрузки

Методы работы с Отгрузками:

Метод	Описание
sale.shipment.add	Добавляет элемент коллекции отгрузок.
sale.shipment.delete	Удаляет элемент коллекции отгрузок.
sale.shipment.get	Метод для доступа к полям отгрузки заказа и полям связанных сущностей.
sale.shipment.getFields	Возвращает поля отгрузки.
sale.shipment.list	Получает список элементов коллекции отгрузок.
sale.shipment.update	Метод для обновления полей элемента коллекции отгрузок.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Табличная часть отгрузки

Методы работы с табличной частью отгрузки:

Метод	Описание
sale.shipmentitem.add	Добавляет элемент коллекции табличной части отгрузки.
sale.shipmentitem.delete	Удаляет элемент коллекции табличной части отгрузки.
sale.shipmentitem.get	Метод для доступа к полям элемента табличной части отгрузки.
sale.shipmentitem.getFields	Возвращает поля табличной части отгрузки.
sale.shipmentitem.list	Получает список элементов табличной части отгрузки.
sale.shipmentitem.update	Метод для обновления элемента коллекции табличной части отгрузки.

© «Битрикс», 2001-2008, «1С-

Статусы

Методы работы со статусами:

Метод	Описание
sale.status.add	Создает статус.
sale.status.delete	Удаляет статус.
sale.status.get	Метод для доступа к полям статуса.
sale.status.getFields	Возвращает поля статуса.
sale.status.list	Получает список статусов.
sale.status.update	Метод для обновления статуса.

© «Битрикс», 2001-2008, «1С-

Локализация статусов

Методы работы с локализацией статусов:

Метод	Описание
sale.statusLang.add	Добавляет локализацию.
sale.statusLang.deleteByFilter	Удаляет локализацию.
sale.statusLang.getListLangs	Метод для получения списка языков для локализации.
sale.statusLang.getFields	Возвращает поля локализации статусов.
sale.statusLang.list	Метод для получения списка локализаций статусов.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

Событие	Вызывается
onCrmDealAdd	при создании сделки
onCrmDealUpdate	при обновлении сделки
onCrmDealDelete	при удалении сделки
onCrmDealUserFieldAdd	при добавлении пользовательского поля
onCrmDealUserFieldUpdate	при изменении пользовательского поля
onCrmDealUserFieldDelete	при удалении пользовательского поля
onCrmDealUserFieldSetEnumValues	при изменении набора значений для пользовательского поля списочного типа
onCrmDealRecurringAdd	при создании новой регулярной сделки
onCrmDealRecurringUpdate	при обновлении регулярной сделки
onCrmDealRecurringDelete	при удалении регулярной сделки
onCrmDealRecurringExpose	при создании новой сделки из регулярной сделки

События

Событие	Вызывается
onCrmInvoiceAdd	при создании счёта
onCrmInvoiceDelete	при удалении счёта
onCrmInvoiceSetStatus	при изменении статуса счёта
onCrmInvoiceUpdate	при обновлении счёта
onCrmInvoiceUserFieldAdd	при добавлении пользовательского поля
onCrmInvoiceUserFieldUpdate	при изменении пользовательского поля
onCrmInvoiceUserFieldDelete	при удалении пользовательского поля
onCrmInvoiceUserFieldSetEnumValues	при изменении набора значений для пользовательского поля списочного типа
onCrmInvoiceRecurringAdd	при создании нового регулярного счета
onCrmInvoiceRecurringUpdate	при обновлении регулярного счета
onCrmInvoiceRecurringDelete	при удалении регулярного счета

onCrmInvoiceRecurringExpose	при выставлении нового счета из регулярного счета

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

Событие	Вызывается
onCrmTimelineCommentAdd	при добавлении нового комментария в таймлайне
onCrmTimelineCommentUpdate	при обновлении нового комментария в таймлайне
onCrmTimelineCommentDelete	при удалении нового комментария в таймлайне

© «Битрикс», 2001-2008, «1С-

Операторы

Методы работы с операторами Открытых линий.

Метод	Описание	С версии
imopenlines.session.intercept	Метод для того, чтобы текущий оператор забрал диалог у которого уже есть другой оператор.	
imopenlines.operator.transfer	Метод для перевода диалога текущим оператором на другого оператора/ линию.	
imopenlines.operator.spam	Метод для пометки диалога как спам текущим оператором.	
imopenlines.operator.skip	Метод для пропуска диалога текущим оператором.	
imopenlines.operator.finish	Метод для завершения диалога текущим оператором.	

imopenlines.operator.answer	Метод для приема диалога текущим оператором.	

Настройка открытой линии

Методы для настройки Открытых линий

Метод	Описание	С версии
imopenlines.config.add	Метод для добавления линии.	
imopenlines.config.delete	Метод для удалении линии.	
imopenlines.config.get	Метод позволяет получить линию по её ID.	
imopenlines.config.list.get	Метод для получения списка линий.	
imopenlines.config.update	Метод для обновления линии.	
imopenlines.config.path.get	Метод получения путей к публичной части открытых линий и домену портала.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Корзина

Методы работы с Корзиной:

Метод	Описание
sale.basketitem.add	Добавляет в коллекцию элемент корзины.
sale.basketitem.getFields	Возвращает поля элемента корзины.
sale.basketitem.catalogProductAdd	Добавляет в коллекцию элемент корзины.
sale.basketItem.getCatalogProductFields	Возвращает поля элемента корзины.
sale.basket.list	Получает список элементов коллекции корзины.
sale.basketitem.delete	Удаляет элемент коллекции корзины.
sale.basketitem.get	Метод для доступа к полям элемент коллекции корзины и полям связанных сущностей.
sale.basketitem.update	Метод для обновления полей элемента коллекции корзины.

Свойства корзины

Методы работы со свойствами корзины:

Метод	Описание
sale.basketProperties.get	Метод для доступа к полям элемента коллекции свойств табличной части корзины.
sale.basketproperties.add	Добавляет элемент коллекции свойств табличной части корзины.
sale.basketproperties.delete	Удаляет элемент коллекции свойств табличной части корзины.
sale.basketproperties.getFields	Возвращает поля элемента свойств корзины.
sale.basketproperties.list	Получает список элементов свойств коллекции корзины.
sale.basketproperties.update	Метод для обновления элемента коллекции свойств табличной части корзины.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Оплаты

Методы работы с Оплатами:

Метод	Описание
sale.payment.add	Добавляет элемент коллекции оплат.
sale.payment.delete	Удаляет элемент коллекции оплат.
sale.payment.get	Метод для доступа к полям элемента коллекции оплат.
sale.payment.getFields	Возвращает поля оплаты.
sale.payment.list	Получает список элементов коллекции оплат.
sale.payment.update	Метод для обновления полей элемента коллекции оплат.

© «Битрикс», 2001-2008, «1С-

Типы плательщиков

Методы работы с Типами плательщиков:

Метод	Описание
sale.persontype.add	Добавляет тип плательщика.
sale.persontype.delete	Удаляет тип плательщика.
sale.persontype.get	Метод для доступа к полям типа плательщика.
sale.persontype.getFields	Возвращает поля типа плательщика.
sale.persontype.list	Получает список типов плательщика.
sale.persontype.update	Метод для обновления полей типа плательщика.

© «Битрикс», 2001-2008, «1С-

Соответствие физ. и юр. лицам

Методы работы с элементами соответствия физическим и юридическим лицам:

Метод	Описание
sale.businessValuePersonDomain.add	Добавляет элемент соответствия физическим и юридическим лицам.
sale.businessValuePersonDomain.deleteByFilter	Удаляет элемент соответствия физическим и юридическим лицам.
sale.businessValuePersonDomain.getFields	Возвращает поля элемента соответствия физическим и юридическим лицам.
sale.businessValuePersonDomain.list	Получает список элементов соответствия физическим и юридическим лицам.

Заказ

Методы работы с Заказом:

Метод	Описание
sale.order.add	Добавляет заказ.
sale.order.delete	Удаляет заказ и связанные сущности.
sale.order.get	Метод для доступа к полям заказа и полям связанных сущностей.
sale.order.getFields	Возвращает поля заказа.
sale.order.list	Получает список заказов.
sale.order.tryadd	Добавляет заказ без сохранения заказа.
sale.order.tryupdate	Метод для обновления полей заказа без сохранения заказа.
sale.order.update	Метод для обновления полей заказа.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Заказы из внешних источников

Методы работы с Заказами из внешних источников:

Метод	Описание
sale.tradeBinding.getFields	Возвращает поля заказов из внешних источников.
sale.tradeBinding.list	Метод для получения списка заказов из внешних источников.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Торговые платформы

Методы работы с Торговыми платформами:

Метод	Описание
sale.tradePlatform.getFields	Возвращает поля торговых платформ статусов.
sale.tradePlatform.list	Метод для получения списка торговых платформ.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Таймлайн

Функция	Описание
crm.timeline.bindings.bind	Привязывает запись в таймлайне к элементу crm.
crm.timeline.bindings.fields	Возвращает список полей привязки элементов crm к записям в таймлайне.
crm.timeline.bindings.list	Возвращает список привязок к записи в таймлайне.
crm.timeline.bindings.unbind	Снимает привязку записи таймлайна с элемента crm.
crm.timeline.comment.add	Добавляет новый комментарий в таймлайн.
crm.timeline.comment.delete	Удаляет сообщение.
crm.timeline.comment.fields	Возвращает список полей комментария таймлайна.
crm.timeline.comment.get	Возвращает информацию о сообщении.
crm.timeline.comment.list	Возвращает список всех сообщений для определенного crm элемента.
crm.timeline.comment.update	Обновляет сообщение.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Свойства заказа

Методы работы со свойствами заказа:

Метод	Описание
sale.property.add	Добавляет свойство заказа.
sale.property.delete	Удаляет свойство заказа.
sale.property.get	Метод для доступа к полям и настройкам свойства заказа.
sale.property.getFieldsByType	Возвращает поля и настройки свойства заказа для определенного типа свойства.
sale.property.list	Получает список свойств заказа.
sale.property.update	Метод для обновления полей свойства заказа.

© «Битрикс», 2001-2008, «1С-

Группы свойств

Методы работы с группами свойств:

Метод	Описание
sale.propertygroup.add	Добавляет группу свойств.
sale.propertygroup.delete	Удаляет группу свойств.
sale.propertygroup.get	Метод для доступа к полям группы свойств.
sale.propertygroup.getFields	Возвращает поля группы свойств.
sale.propertygroup.list	Получает список групп свойств.
sale.propertygroup.update	Метод для обновления полей группы свойств.

© «Битрикс», 2001-2008, «1С-

Привязка свойства

Методы работы с привязкой свойств:

Метод	Описание
sale.propertyRelation.add	Добавляет привязку свойства.
sale.propertyRelation.deleteByFilter	Удаляет группу свойств.
sale.propertyRelation.getFields	Возвращает поля привязки свойства.
sale.propertyRelation.list	Получает список привязок свойства.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Значения свойства

Методы работы со значениями свойств:

Метод	Описание
sale.propertyvalue.delete	Удаляет значения свойства.
sale.propertyvalue.get	Метод для доступа к полям значений свойств заказа.
sale.propertyvalue.getFields	Возвращает поля значения свойства.
sale.propertyvalue.list	Получает список значений свойств.
sale.propertyvalue.modify	Метод для изменения значения свойства.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Вариант свойства

Методы работы с вариантами свойств:

Метод	Описание
sale.propertyVariant.add	Добавляет вариант свойства.
sale.propertyVariant.delete	Удаляет вариант свойства.
sale.propertyVariant.get	Метод для доступа к полям варианта свойства.
sale.propertyVariant.getFields	Возвращает поля варианта свойства.
sale.propertyVariant.list	Получает список вариантов свойств.
sale.propertyVariant.update	Метод для обновления полей варианта свойства.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

item Предоставляет работу с PHP-классом CTaskItem.

Вместо **item** можно также использовать его синоним **ctaskitem**.

Метод	Описание
task.item.add	Создает новую задачу.
task.item.delete	Удаляет задачу.
task.item.getdata	Возвращает массив данных о задаче.
task.item.getmanifest	Показ списка методов.
task.item.list	Возвращает массив задач, каждая из которых содержит массив полей.
task.item.update	Обновляет данные по задаче.
task.item.getdescription	Возвращает описание задачи.
task.item.getfiles	Возвращает массив, содержащий ссылки на файлы, прикрепленные к задаче.
task.item.getdependson	Возвращает массив, содержащий идентификаторы

	задач, от которых зависит задача.
task.item.getallowedactions	Возвращает массив, идентификаторов допустимых действий над задачей.
task.item.getallowedtaskactionsasstrings	Возвращает массив, ключи которого являются названиями действий , а значения показывают, допустимо действие или нет.
task.item.isactionallowed	Проверяет, разрешено ли действие.
task.item.delegate	Делегирует задачу новому пользователю.
task.item.startexecution	Переводит задачу в статус «выполняется».
task.item.defer	Переводит задачу в статус «отложена».
task.item.complete	Переводит задачу в статус «завершена» или «условно завершена (ждет контроля исполнителя)».
task.item.renew	Переводит задачу в статус «не выполняется».
task.item.approve	Переводит задачу, ожидающую

	контроля, в статус «завершена».
task.item.disapprove	Переводит задачу, ожидающую контроля, в статус «не выполняется».
task.item.addtofavourite	Добавляет задачу в Избранное.
task.item.deletefromfavorite	Удаляет задачу из Избранного.
task.item.addfile	Загружает к задаче файл.
task.item.deletefile	Удаляет привязку файла к задаче.

Примечание

Для получения тэгов конкретной задачи необходимо передать параметр `/rest/task.item.gettags.xml?TASK_ID=3&auth=18tci5kga6v12g8okzm5r26sv0n9is84`. Запрос может быть как ID, так и TASK_ID. Принципиально, чтобы этот параметр был первым. В ответ вернётся `{"result": ["TAG1","TAG2","ETC..."]}`.

Сайты

Rest-методы, доступные при работе с Сайтами (значений SCOPE: **landing**).

Работа с разделами списка

Метод	Описание	С версии
lists.section.add	Метод создаёт раздел списка.	
lists.section.get	Метод возвращает даные о разделах.	
lists.section.update	Метод обновляет раздел списка.	
lists.section.delete	Метод удаляет раздел списка.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Сделки

Функция	Описание
crm.deal.add	Создаёт новую сделку.
crm.deal.contact.add	Добавляет контакт к указанной сделке.
crm.deal.contact.delete	Удаляет контакт из указанной сделки.
crm.deal.contact.fields	Возвращает описание полей для связи сделка-контакт.
crm.deal.contact.items.delete	Очищает набор контактов, связанных с указанной сделкой.
crm.deal.contact.items.get	Возвращает набор контактов, связанных с указанной сделкой.
crm.deal.contact.items.set	Устанавливает набор контактов, связанных с указанной сделкой.
crm.deal.delete	Удаляет сделку и все связанные с ней объекты.
crm.deal.fields	Возвращает описание полей сделки.
crm.deal.get	Возвращает сделку по идентификатору.
crm.deal.list	Возвращает список сделок по фильтру.

crm.deal.productrows.get	Возвращает товарные позиции сделки.
crm.deal.productrows.set	Устанавливает (создаёт или обновляет) товарные позиции сделки.
crm.deal.recurring.add	Добавляет новую настройку для регулярной сделки.
crm.deal.recurring.delete	Удаляет существующую настройку для шаблона регулярной сделки.
crm.deal.recurring.expose	Создает новую сделку из шаблона.
crm.deal.recurring.fields	Возвращает список полей настройки шаблона регулярной сделки с описанием.
crm.deal.recurring.get	Возвращает поля настройки шаблона регулярной сделки по идентификатору.
crm.deal.recurring.list	Возвращает список настроек шаблонов регулярных сделок по фильтру.
crm.deal.recurring.update	Обновляет существующую настройку для шаблона регулярной сделки.
crm.deal.update	Обновляет существующую сделку.
crm.deal.userfield.add	Создаёт новое пользовательское поле для сделок.
crm.deal.userfield.get	Возвращает пользовательское поле сделок по идентификатору.

crm.deal.userfield.list	Возвращает список пользовательских полей сделок по фильтру.
crm.deal.userfield.update	Обновляет существующее пользовательское поле сделок.
crm.deal.userfield.delete	Удаляет пользовательское поле сделок.

Общий список **событий сделки** приведен здесь.

Счета (старые)

Функция	Описание
crm.invoice.add	Создаёт новый счёт.
crm.invoice.delete	Удаляет счёт.
crm.invoice.fields	Возвращает описание полей счёта и товаров, входящих в него.
crm.invoice.get	Возвращает счёт по идентификатору.
crm.invoice.list	Возвращает список счетов.
crm.invoice.recurring.add	Добавляет новую настройку для регулярного счета.
crm.invoice.recurring.delete	Удаляет существующую настройку для шаблона регулярного счета.
crm.invoice.recurring.expose	Создает новый счет из шаблона.
crm.invoice.recurring.fields	Возвращает список полей настройки шаблона регулярного счета с описанием.
crm.invoice.recurring.get	Возвращает поля настройки шаблона регулярного счета по идентификатору.
crm.invoice.recurring.list	Возвращает список настроек шаблонов регулярных счетов по

	фильтру.
crm.invoice.recurring.update	Обновляет существующую настройку для шаблона регулярного счета.
crm.invoice.update	Обновляет существующий счёт.
crm.invoice.userfield.add	Создаёт новое пользовательское поле для счетов.
crm.invoice.userfield.delete	Удаляет пользовательское поле счетов.
crm.invoice.userfield.get	Возвращает пользовательское поле счетов по идентификатору.
crm.invoice.userfield.list	Возвращает список пользовательских полей счетов по фильтру.
crm.invoice.userfield.update	Обновляет существующее пользовательское поле счетов.
crm.paysystem.fields	Возвращает описание полей для способов оплаты.
crm.paysystem.list	Возвращает список способов оплаты.
crm.persontype.fields	Возвращает описание полей для типов плательщиков.
crm.persontype.list	Возвращает список типов плательщиков.
crm.invoice.getexternallink	Возвращает ссылку на счёт.

Коннекторы для внешних мессенджеров

Описание REST-методов и событий, работающих с коннекторами для внешних мессенджеров. Разрешение **imopenlines**.

© «Битрикс», 2001-2008, «1С-

Встраивание в календарь

Плейсмент **CALENDAR_GRIDVIEW** позволяет встроиться в календарный вид (вверху - там где день/неделя/месяц/список).

Пример

Как можно забиндить (привязать) приложение к календарной встройке::

```
 BX24.callMethod('placement.bind', {
   PLACEMENT:'CALENDAR_GRIDVIEW',
   HANDLER: 'http://svd.org/svdapp.php',
   TITLE: 'Custom tab'
}, (result) => {console.log(result)});
```

Если в самом приложении вызвать

```
echo "<pre>";
print_r($_REQUEST);
echo "</pre>";
```

то увидим, что туда приходят определенные параметры. В частности:

```
[PLACEMENT_OPTIONS] => {
   "viewRangeFrom":"2018-09-30",
```

```
    "viewRangeTo":"2018-11-04"
}
```

Также при работе во встройке есть определенный интерфейс: методы и события.

Методы (js методы)

Метод	Описание

`getEvents`	Получение событий
	`var dateFrom = new Date();`
	`var dateTo = new Date(dateFrom.getTime() + 86400 * 30);`
	`dateFrom.setHours(0, 0, 0, 0);`
	`dateTo.setHours(0, 0, 0, 0);`
	
	`BX24.placement.call('getEvents', {`
	`  dateFrom: dateFrom,`
	`  dateTo: dateTo`
	`  }, function(events)`
	`{`
	`  console.log('getEvents responce:');`
	`  console.dir(events);`
	`});`

`viewEvent`	Просмотр события (открытие карточки просмотра)
	`BX24.placement.call('viewEvent', {`
	`  id: "1431170", - id события`
	`  dateFrom: "11.07.2018" - дата события (не обязательно, но важно для регулярных)`
	`},`
	`function(){});`

`addEvent`	Добавление нового события (открытие карточки)

	`BX24.placement.call('addEvent', function(){});`
`editEvent`	Редактирование события (открытие карточки) `BX24.placement.call('editEvent', {uid: "1431171|19.07.2018"}, function(){});`
`deleteEvent`	Удаление события `BX24.placement.call('deleteEvent', { id: "1431169" }, function(){});`

События, которые можно отслеживать в плейсменте

События	Описание
Calendar.customView:refreshEntries	Обновление событий.
Calendar.customView:decreaseViewRangeDate	Нажимаем на стрелочку назад, т.е. отматываем календарь на предыдущие даты (на какие точно - может решать автор плейсмента).
Calendar.customView:increaseViewRangeDate	Нажимаем на

	стрелочку вперед, т.е. отматываем календарь на следующие даты (на какие точно - может решать автор плейсмента).
Calendar.customView:adjustToDate	Переход к конкретной дате (она передается в параметре).

Дополнительно

Встраивание приложений (REST).

Встраивание приложений в курсе Маркетплейс Битрикс24.

Встраивание приложений в виде пользовательских типов полей в учебном курсе.

Добавление своих методов REST API в учебном курсе.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

Для всех событий входящий параметр - идентификатор (ID) сущности, по которой сработало событие.

Событие	Вызывается
OnCalendarEntryAdd	при добавлении события.
OnCalendarEntryUpdate	при изменении события.
OnCalendarEntryDelete	при удалении события.
OnCalendarSectionAdd	при добавлении [dw]секции календаря[/dw] [di]CALENDAR_SECTION_ID. Подробнее[/di]. Также будет вызываться при добавлении ресурса. **Примечание.** Технически ресурс равен секции. Т.к. создаваемые ресурсы помещаются в особый тип календарей и для каждого ресурса создается секция, то при удалении или добавлении ресурсов будут срабатывать эти события.
OnCalendarSectionUpdate	при изменении секции/ресурса.
OnCalendarSectionDelete	при удалении секции/ресурса.

© «Битрикс», 2001-2008, «1С-

Методы для работы с сущностью Сайт

Метод	Описание	С версии
landing.site.add	Метод для добавления сайта.	
landing.site.addFolder	Метод добавляет папку в сайт.	21.800.0
landing.site.delete	Метод для удаления сайта.	
landing.site.fullExport	Метод экспортирует сайт и всего его страницы в специальный массив.	
landing.site.getFolders	Метод получает папки сайта.	21.800.0
landing.site.getList	Метод для получения списка сайтов	
landing.site.getPreview	Метод возвращает URL изображения-	21.800.0

	превью сайта (превью индексной страницы).	
landing.site.getPublicUrl	Метод возвращает полный URL сайта (сайтов).	18.7.500
landing.site.getadditionalfields	Метод для получения дополнительных полей сайта.	
landing.site.markDelete	Метод помечает сайт как удаленный.	
landing.site.markFolderDelete	Метод помечает папку как удаленную (помещенную в корзину).	21.800.0
landing.site.markFolderUnDelete	Метод помечает папку как не удаленную (возвращает из корзины).	21.800.0
landing.site.markUnDelete	Метод помечает сайт как не удаленный.	
landing.site.publication	Метод публикует сайт (и все его страницы).	
landing.site.publicationFolder	Метод публикует папку сайта.	21.800.0
landing.site.unPublicFolder	Метод снимает с	21.800.0

	публикации папку сайта.	
landing.site.unpublic	Метод снимает с публикации сайт (и все его страницы).	
landing.site.update	Метод для изменения сайта.	
landing.site.updateFolder	Метод изменяет папку в сайте.	21.800.0

Контроль времени

Метод	Описание
timeman.timecontrol.report.add	Метод для отправки отчета о выявленном отсутствии.
timeman.timecontrol.reports.get	Метод для получения отчета о выявленных отсутствиях.
timeman.timecontrol.reports.settings.get	Метод для получения пользовательских настроек для построения интерфейса отчетов инструмента контроля времени.
timeman.timecontrol.reports.users.get	Метод для получения списка пользователей, относящихся к указанному подразделению.
timeman.timecontrol.settings.get	Метод для получения настроек инструмента контроля времени.
timeman.timecontrol.settings.set	Метод для установки настроек инструмента контроля времени.

Офисные сети

Метод	Описание
timeman.networkrange.check	Метод для проверки IP-адреса на вхождение в диапазоны сетевых адресов офисной сети.
timeman.networkrange.get	Метод для получения диапазонов сетевых адресов, входящих в офисную сеть.
timeman.networkrange.set	Метод для установки диапазонов сетевых адресов, входящих в офисную сеть.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Партнерские шаблоны

Партнерские шаблоны дают возможность добавить свой шаблон в мастер создания сайта или страницы.

Метод	Описание	С версии
landing.demos.register	Метод регистрирует шаблон в мастере создания сайта и страницы.	
landing.demos.unregister	Метод удаляет зарегистрированный партнерский шаблон.	
landing.demos.getList	Метод для получения списка доступных партнерских шаблонов текущего приложения.	
landing.demos.getSiteList	Метод для получения списка доступных шаблонов для создания сайтов.	
landing.demos.getPageList	Метод для получения списка доступных шаблонов для создания страниц.	

Компании

Функция	Описание
crm.company.add	Создаёт новую компанию.
crm.company.delete	Удаляет компанию и все связанные с ней объекты.
crm.company.fields	Возвращает описание полей компании.
crm.company.get	Возвращает компанию по идентификатору.
crm.company.list	Возвращает список компаний по фильтру.
crm.company.update	Обновляет существующую компанию.
crm.company.userfield.add	Создаёт новое пользовательское поле для компаний.
crm.company.userfield.get	Возвращает пользовательское поле компаний по идентификатору.
crm.company.userfield.list	Возвращает список пользовательских полей компаний по фильтру.
crm.company.userfield.update	Обновляет существующее пользовательское поле

	компаний.
crm.company.userfield.delete	Удаляет пользовательское поле компаний.
crm.company.contact.add	Добавляет контакт к указанной компании.
crm.company.contact.delete	Удаляет контакт из указанной компании.
crm.company.contact.fields	Возвращает описание полей для связи компания-контакт.
crm.company.contact.items.delete	Очищает набор коонтактов, связанных с указанной компанией.
crm.company.contact.items.get	Возвращает набор контактов, связанных с указанной компанией.
crm.company.contact.items.set	Устанавливает набор контактов, связанных с указанной компанией.

Общий список **событий компании** приведен здесь.

Встраивание приложений

Разработчику приложений для Маркетплейса Битрикс24 доступен простой механизм встраивания приложения в интерфейс Битрикс24. В разделе приведены справочные данные по методам и местам встраивания. Детальное описание примеров работы представлено в курсе Приложения Битрикс24.

Встраиваемые приложения, к которым пользователь доступа не имеет, не будут отображаться в списках, пунктах меню или в других местах, где могут быть видны приложения. Исключение - встройка через тип пользовательского поля. Но в фрейм не будет передаваться авторизация, и разработчик приложения должен это учитывать.

Доступны следующие места для встраивания:

- scope CRM
- scope intranet
- scope telephony
- scope landing
- scope worgroups
- task
- calendar
- contact_center
- произвольное место

scope CRM	
Код	**Л**
CRM_LEAD_LIST_MENU	[dw]Контекстное меню[/
CRM_LEAD_DETAIL_TAB	Пункт в [dw]верхнем ме карточке лида
CRM_LEAD_DETAIL_ACTIVITY	Пункт в [dw]меню таймл

CRM_LEAD_DETAIL_TOOLBAR	Пункт в [dw]списке при карточки лида
Код	**Сд**
CRM_DEAL_LIST_MENU	[dw]Контекстное меню[/
CRM_DEAL_DETAIL_TAB	Пункт в [dw]верхнем ме карточке сделки
CRM_DEAL_DETAIL_ACTIVITY	Пункт в [dw]меню таймл сделки
CRM_DEAL_DETAIL_TOOLBAR	Пункт в [dw]списке при карточки сделки.
Код	**Кон**
CRM_CONTACT_LIST_MENU	[dw]Контекстное меню[/
CRM_CONTACT_DETAIL_TAB	Пункт в [dw]верхнем ме карточке контакта
CRM_CONTACT_DETAIL_ACTIVITY	Пункт в [dw]меню таймл контакта
CRM_CONTACT_DETAIL_TOOLBAR	Пункт в [dw]списке при карточки контакта.
Код	**Ком**
CRM_COMPANY_LIST_MENU	[dw]Контекстное меню[/
CRM_COMPANY_DETAIL_TAB	Пункт в [dw]верхнем ме карточке компании
CRM_COMPANY_DETAIL_ACTIVITY	Пункт в [dw]меню таймл компании
CRM_COMPANY_DETAIL_TOOLBAR	Пункт в [dw]списке при

	карточки компании.
Код	**Сч**
CRM_INVOICE_LIST_MENU	[dw]Контекстное меню[/
Код	**Предл**
CRM_QUOTE_LIST_MENU	[dw]Контекстное меню[/ предложений
Код	**Д**
CRM_ACTIVITY_LIST_MENU	[dw]Контекстное меню[/
Код	**CRM-ан**
CRM_ANALYTICS_MENU	[dw]Меню[/dw][di][/di
scope intranet	
Код	**Встраивани**
CRM	
CRM_FUNNELS_TOOLBAR	[dw]Кнопка[/dw][di][/ продаж
CRM_ANALYTICS_TOOLBAR	[dw]Кнопка[/dw][di][/ аналитики
CRM_*_LIST_TOOLBAR	[dw]Кнопка[/dw][di][/ Допустимые объекты: D COMPANY, INVOICE, QUC
CRM_*_DETAIL_TOOLBAR	[dw]Кнопка[/dw][di][/ Допустимые объекты: D COMPANY, INVOICE, QUC
CRM_*_ACTIVITY_TIMELINE_MENU	[dw]Кнопка[/dw][di][/ объекта (Только для объ

CRM_*_DOCUMENTGENERATOR_BUTTON	Кнопка в документах. Д LEAD, CONTACT, COMPAN
CRM_*_ROBOT_DESIGNER_TOOLBAR	[dw]Кнопка[/dw][di][/ (Только для объектов DI
Задачи	
TASK_USER_LIST_TOOLBAR	[dw]Кнопка[/dw][di][/
TASK_GROUP_LIST_TOOLBAR	[dw]Кнопка[/dw][di][/
TASK_ROBOT_DESIGNER_TOOLBAR	[dw]Кнопка[/dw][di][/
Рабочие группы	
SONET_GROUP_ROBOT_DESIGNER_TOOLBAR	[dw]Кнопка[/dw][di][/
SONET_GROUP_TOOLBAR	[dw]Кнопка[/dw][di][/ группе.
Профиль пользователя	
USER_PROFILE_MENU	[dw]Кнопка в главном м портала.
USER_PROFILE_TOOLBAR	[dw]Кнопка в профиле[/
scope telephony	
Код	**Карточ**
CALL_CARD	[dw]Карточка звонка[/d
Код	**Статисти**
TELEPHONY_ANALYTICS_MENU	[dw]Меню статистики зв
scope landing	

Код	Наст
LANDING_SETTINGS	Меню настроек (Страни
Код	**Редакт**
LANDING_BLOCK	Пункт редактирования л
scope worgroups	
Код	**Наст**
SONET_GROUP_DETAIL_TAB	[dw]Закладка[/dw][di]
scope task	
Код	**Наст**
TASK_LIST_CONTEXT_MENU	[dw]Контекстное меню[/ задач.
TASK_VIEW_TAB	[dw]Вкладка[/dw][di][задачи
TASK_VIEW_SIDEBAR	[dw]Боковая панель[/dv просмотра задачи
TASK_VIEW_TOP_PANEL	Пункт в верхнем[dw]мен просмотра задачи
scope calendar	
Код	**Наст**
CALENDAR_GRIDVIEW	Список [dw]видов отобр календаря
scope contact_center	
Код	**Наст**

CONTACT_CENTER	[dw]"Квадратик"[/dw][d центра.
Произвольное место	
Код	**Наст**

REST_APP_URI	На каждом портале кажд
	позволяется зарегистри
	встройку методом placer
	
	Эта встройка не имеет в
	которая позволяет поль
	самостоятельно. Прилож
	ссылку на свою встройк
	данной встройки ссылка
	/marketplace/view/#APP
	код вашего приложения
	
	Встройка может принима
	параметров в get ключе
	/marketplace/view/#APP
	В этом случае в PLACEM
	равен:
	
	`[ 'test' = 'y' ]`
	
	Пример: как с помощью
	ссылку в чат:
	
	`[url=/marketplace/vi params[test]=y]uri[/`
	
	`<? $placement = 'REST_A $url = 'https://exam CRest::call( 'placement.bind', [ 'PLACEMENT' => 'HANDLER' => $ ]`

```
);
?>
```

Пример выше регистрир
встройку. Для данного Р
портале зарегистрирова
после этого она станови
В вашем приложении вы
встройку:

```
<a href="/marketplac
=$myApp['CODE']?>/">
```

И разместить её в посте

```
<?
    CRest::call('log
        [
            'POST_TI
            'POST_ME
[url=/marketplace/vi
params[test]=y]uri[/
            'DEST' =
        ]);
?>
```

И отправить её с помощ
пользователю методом:

```
<?
CRest::call(
    'imbot.message.a
    [
        'DIALOG_ID'
        'MESSAGE' =>
[url=/marketplace/vi
params[test]=y]uri[/

    ]
);
?>
```

Ссылку можно вставить
которое поддерживает В
других встроек как обыч
GET параметра **params**
отображаемые данные в

	эту возможность в абсол передавая любое колич ключе params, наприме

PAGE_BACKGROUND_WORKER	На каждом портале каж
	позволяется зарегистри
	встройку методом placer
	
	▪ `errorHandlerUrl` -
	обработчика ошиб
	грузится (дольше 3
	приходит событие
	а сама встройка уд
	
	Пример регистрации:
	
	`CRest::call(`
	`  'placement.bind'`
	`  [`
	`  'PLACEMENT' =`
	`'PAGE_BACKGROUND_WOR`
	`  'HANDLER' =>`
	`'http://portal.vlads`
	`ty=1',`
	`  'OPTIONS' =>`
	`  'errorHa`
	`'http://portal.vlads`
	`ty=1',`
	`  ],`
	`  'LANG_ALL' =`
	`  'ru' =>`
	`  'TIT`
	`  ]`
	`  ]`
	`  ]`
	`);`

Дополнительно

- Взаимодействие с пользовательским интерфейсом Битрикс24

Места встраивания

В модуле Сайтов встраивание происходит через внутренний rest-метод самого модуля landing.repo.bind и landing.repo.unbind (удаление места встраивания). Подробнее смотрите на страницах описания мест встраивания.

Места встраивания для scope landing	
Код	**Настройки**
LANDING_SETTINGS	Меню настроек (Страницы / Сайта)
Код	**Редактирование**
LANDING_BLOCK	Пункт редактирования любого блока.

Смотри также

- Взаимодействие с пользовательским интерфейсом Битрикс24

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Методы работы с Живой лентой

Разрешение **Живая лента** (log)

Методы

Метод	Описание
log.blogpost.add	Добавляет в Живую Ленту сообщение от имени текущего пользователя.
log.blogpost.getusers.important	Отдает массив ID пользователей, прочитавших Важное сообщение.
log.blogpost.get	Возвращает массив с доступными текущему пользователю сообщениями Живой ленты. Каждое из сообщений представляет собой массив значений полей (включая пользовательские поля).
log.blogpost.delete	Удаляет сообщение Живой Ленты.
log.blogpost.share	Добавляет получателей в сообщение Живой Ленты.
log.blogpost.update	Изменяет сообщение Живой Ленты.

log.blogcomment.add	Добавляет комментарий к сообщению Живой ленты.

Специальные блоки

Блоки, имеющие особое назначение.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Использование методов REST

Для использования методов REST необходимо либо создать локальное приложение в рамках конкретного Битрикс24 или вебхук, либо разработать тиражное решение и добавить его в партнерский кабинет. Эти варианты описаны в курсе Приложения для Битрикс24. В общем виде вызов метода REST для локальных и тиражных приложений выглядит как HTTPS-запрос следующего вида

```
https://домен_Б24.bitrix24.{ru|com|de}/rest/
имя_метода.транспорт?параметры_метода&auth=ключ_авторизации
```

"транспорт" - необязательный параметр, который может иметь значения json или xml. По умолчанию - json.

Запрос может отправляться как методом GET, так и POST.

Значения параметров методов принимаются в кодировке UTF-8.

Для вебхуков синтаксис вызова отличается, однако все равно содержит имя метода, параметры метода и транспорт.

Фактически, это означает, что вы можете обращаться к методам REST API Битрикс24 используя любые языки программирования и средства разработки, которые поддерживают работу с HTTPS.

Помните об [dw]ограничениях на интенсивные запросы к REST[/dw][di]Существует лимит на число запросов. Разрешается два запроса в секунду. Если лимит превышается, то ограничение начинает срабатывать после **50 запросов**.

Подробнее в курсе Приложения для Битрикс24.Маркет[/di].

Если Ваше приложение/вебхук создаёт аномальную нагрузку на портал, может сработать блокировка:

```
[
'error' => 'OVERLOAD_LIMIT',
'error_description' => 'REST API is blocked due to overload.'
]
```

Для разблокировки необходимо обратиться в техподдержку.

Внимание: все ключи входящих массивов должны передаваться в верхнем регистре.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Бизнес-процесс

Методы работы с Бизнес-процессами

Метод	Описание
bizproc.workflow.instance.list	Метод возвращает список запущенных бизнес-процессов. Алиас bizproc.workflow.instances.
bizproc.workflow.instances	Метод возвращает список запущенных бизнес-процессов.
bizproc.workflow.terminate	Метод останавливает активный Бизнес-процесс.
bizproc.workflow.start	Метод запускает Бизнес-процесс.
bizproc.workflow.kill	Метод удаляет запущенный бизнес-процесс.

Пример

Пример rest приложения для работы с запуском/остановкой БП:

```
1 <?php
2 header('Content-Type: text/html; charset=UTF-8');
3 ?>
4 <!DOCTYPE html>
```

```
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title></title>
 <style type="text/css">
        .wf-list {
                list-style-type: none;
        }
        .wf-list li {
                margin: 25px 0;
        }
        .wf-title {
                font-size: 16px;
                font-weight: bold;
                color: grey;
        }
        .wf-complete {
                padding: 5px;
                border: 1px dotted;
        }
        .wf-complete .wf-title
        {
                color: green;
                margin-bottom: 10px;
                display: block;
        }
        .wf-complete textarea
        {
                margin-bottom: 10px;
                display: block;
        }
        #user-name
        {
                font-weight: bold;
                color: green;
        }
```

```
    41  </style>
    42 </head>
    43 <body style="display: none">
    44 <script
src="//api.bitrix24.com/api/v1/"></script>
    45 <h1>Hello, <span id="user-name">
</span></h1>
    46 <h2>This is example of Bizproc
workflows API usage</h2>
    47
    48 <button
onclick="selectCRMEntity();">Select LEAD
    49 <span id="selected-entity" data-
entity-id=""></span>
    50 <br>Select template:
    51 <select id="tpl-id"><option
value="">-- choose --</option></select>
    52 <button
onclick="startWf(document.getElementById('se
lected-entity').getAttribute('data-entity-
id'), document.getElementById('tpl-
id').value, getList);">Start BP</button>
    53 <hr>
    54 <button onclick="getList();">Get
workflow instances list</button>
    55 <ul id="inst-list" class="inst-list">
</ul>
    56
    57 <script type="text/javascript">
    58  var currentUser = null;
    59  BX24.init(function()
    60  {
    61
BX24.callMethod('user.current', {},
function(res){
    62                  currentUser =
res.data();
```

```
    63
document.body.style.display = '';
    64
document.getElementById('user-
name').textContent = currentUser['NAME'] + '
' + currentUser['LAST_NAME']
    65          });
    66
    67          getTemplates();
    68  });
    69
    70  function getList()
    71  {
    72          var listNode =
document.getElementById('inst-list');
    73
    74          listNode.innerHTML = '';
    75
    76          BX24.callMethod(
    77
'bizproc.workflow.instance.list',
    78                  {
    79                          select: [
    80
'ID',
    81
'STARTED',
    82
'STARTED_BY',
    83
'TEMPLATE_ID',
    84
'DOCUMENT_ID'
    85                          ],
    86                          order:
{MODIFIED: 'DESC'},
    87                          filter:
```

```
{MODULE_ID: 'crm', ENTITY:
'CCrmDocumentLead'}
    88                  },
    89                  function(result)
    90                  {
    91
if(result.error())
    92
alert("Error: " + result.error());
    93                          else
    94                          {
    95                                  var
wfs = result.data();
    96
    97
wfs.forEach(function(wf)
    98                                  {
    99
renderWfi(wf, listNode);
   100                                  });
   101                          }
   102                  }
   103          );
   104  }
   105
   106  function getTemplates()
   107  {
   108          var selectNode =
document.getElementById('tpl-id');
   109
   110          BX24.callMethod(
   111
'bizproc.workflow.template.list',
   112                  {
   113                          select: [
   114
'ID',
```

```
   115                                    //
'MODULE_ID',
   116                                    //
'ENTITY',
   117                                    //
'DOCUMENT_TYPE',
   118                                    //
'AUTO_EXECUTE',
   119
'NAME',
   120                                    //
'TEMPLATE',
   121                                    //
'PARAMETERS',
   122                                    //
'VARIABLES',
   123                                    //
'CONSTANTS',
   124
'MODIFIED',
   125
'IS_MODIFIED',
   126
'USER_ID',
   127
'SYSTEM_CODE'
   128                            ],
   129                            filter:
{MODULE_ID: 'crm', ENTITY:
'CCrmDocumentLead'}
   130                    },
   131                    function(result)
   132                    {
   133
if(result.error())
   134
alert("Error: " + result.error());
```

```
135                             else
136                             {
137                                     var
templates = result.data();
138
139
console.log(templates);
140
141
templates.forEach(function(tpl)
142                                     {
143
var option =
document.createElement('OPTION');
144
option.setAttribute('value', tpl['ID']);
145
option.textContent = tpl['NAME'];
146
147
selectNode.appendChild(option);
148
149                                     });
150                             }
151                     }
152             );
153     }
154
155     function renderWfi(wf, parentNode)
156     {
157             var li =
document.createElement('LI');
158             var span =
document.createElement('SPAN');
159             span.textContent = wf['ID'];
160             span.classList.add('wf-
title');
```

```
   161
   162          li.appendChild(span);
   163
   164          li.classList.add('wf-
complete');
   165
   166          var wfContainer =
document.createElement('DIV');
   167
   168          var doTerminate = function()
   169          {
   170
terminateWf(wf['ID'], function()
   171                          {
   172
parentNode.removeChild(li);
   173                          }
   174                  );
   175          };
   176
   177          var terminateButton =
document.createElement('BUTTON');
   178          terminateButton.textContent
= 'Terminate';
   179
terminateButton.addEventListener('click',
doTerminate.bind(window), false);
   180
wfContainer.appendChild(terminateButton);
   181
   182          li.appendChild(wfContainer);
   183
   184          parentNode.appendChild(li);
   185  }
   186
   187  function terminateWf(id, cb)
   188  {
```

```
189         var params = {ID: id, STATUS: 'Terminated by rest app.'};
190
191         BX24.callMethod(
192 'bizproc.workflow.terminate',
193                 params,
194                 function(result)
195                 {
196 if(result.error())
197 alert("Error: " + result.error());
198                         else if (cb)
199 cb();
200                 }
201         );
202 }
203
204 function startWf(leadId, tplId, cb)
205 {
206         if (!leadId)
207         {
208                 alert('Lead not selected');
209                 return;
210         }
211
212         var params = {
213                 TEMPLATE_ID: tplId,
214                 DOCUMENT_ID: ['crm', 'CCrmDocumentLead', leadId],
215                 PARAMETERS: null
216         };
217
218         BX24.callMethod(
```

```
   219
'bizproc.workflow.start',
   220                 params,
   221                 function(result)
   222                 {
   223
if(result.error())
   224
alert("Error: " + result.error());
   225                         else if (cb)
   226
cb();
   227                 }
   228          );
   229  }
   230
   231
   232  function selectCRMEntity()
   233  {
   234
document.getElementById('selected-
entity').textContent = '';
   235          BX24.selectCRM({
   236                 entityType: ['lead']
   237          }, function(selected)
   238          {
   239                 if (selected['lead']
&& selected['lead'][0])
   240                 {
   241
document.getElementById('selected-
entity').textContent = selected['lead'][0]
['title'];
   242                         var     id =
selected['lead'][0]['id'];
   243
   244
```

```
document.getElementById('selected-
entity').setAttribute('data-entity-id',
id.substring(2));
   245                    }
   246            })
   247  }
   248
   249 </script>
   250 </body>
   251 </html>
```

© «Битрикс», 2001-2008, «1С-

Календарь

Методы

Метод	Описание
calendar.accessibility.get	Возвращает занятость пользователей из списка.
calendar.event.add	Добавляет новое событие.
calendar.event.delete	Удаляет событие.
calendar.event.get	Возвращает список событий календаря.
calendar.event.get.nearest	Возвращает список будущих событий для текущего пользователя.
calendar.event.update	Редактирует существующее событие.
calendar.meeting.params.set	Устанавливает параметры события для текущего пользователя.
calendar.meeting.status.get	Возвращает статус участия текущего пользователя в событии.
calendar.meeting.status.set	Устанавливает статус участия в событии для текущего пользователя.

calendar.resource.list	Возвращает список (массив) всех ресурсов.
calendar.resource.add	Добавляет новый ресурс.
calendar.resource.update	Изменяет ресурс.
calendar.resource.delete	Удаляет ресурс.
calendar.resource.booking.list	Предоставляет возможность выбрать бронирования ресурсов.
calendar.section.add	Добавляет новый календарь.
calendar.section.delete	Удаляет календарь.
calendar.section.get	Возвращает список календарей.
calendar.section.update	Обновляет календарь.
calendar.settings.get	Возвращает основные настройки календаря.
calendar.user.settings.set	Сохраняет пользовательские настройки календаря.
calendar.event.getbyid	Возвращает событие календаря по идентификатору.

Торговый каталог

Торговый каталог

Методы работы с торговым каталогом:

Метод	Описание
catalog.catalog.add	Метод для добавления торгового каталога.
catalog.catalog.delete	Метод для удаления торгового каталога.
catalog.catalog.get	Метод для получения значений полей торгового каталога по ID.
catalog.catalog.getFields	Метод возвращает поля торгового каталога.
catalog.catalog.isOffers	Метод проверки, является ли указанный торговый каталог торговым каталогом предложений.
catalog.catalog.list	Метод получает список торговых каталогов по фильтру.
catalog.catalog.update	Метод обновляет поля торгового каталога.

Сущности CRM

- Автоматизация
- Сделки
- Валюты
- Внешние каналы
- Дела
- Единицы измерения
- Каталог
- Коммерческие предложения
- Компании
- Контакты
- Лента CRM
- Лиды
- Направления сделок (устаревшее)
- Направления
- Пользовательские поля
- Разделы товаров
- Реквизиты
- Статусы счетов
- Счета (старые)
- Счета (новые)
- Смарт-процессы
- Товарные позиции (старые)
- Товарные позиции
- Товары
- Вспомогательные сущности

Примечание. Во всех примерах кода, приведённых в разделе **CRM**, данные выводятся в консоль. Если нужно вывести данные по другому, то реализуйте свою обработку данных возвращенных вызовами **result.data()** и **result.error()**.

Методы работы с подразделениями

Разрешение **Структура компании** (department)

Методы

Метод	Описание
department.add	Создает подразделение.
department.update	Изменяет подразделение.
department.get	Получение фильтрованного списка подразделений.
department.fields	Получение списка названий полей подразделения.
department.delete	Удаляет подразделение.

Диск

REST-методы, доступные при работе с Диском. Разрешение **disk**.

Метод	Описание	С версии
Версия	Методы работы с версиями файлов	
Папка	Методы работы с папками	
Прикреплённый файл	Методы работы с прикреплёнными файлами	
Файл	Методы работы с файлами	
Хранилище	Методы работы с хранилищами	
Права доступа	Методы работы с правами доступа	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Право "Хранилище данных"

Данное право позволяет создавать на сервере **Битрикс24** собственные хранилища произвольных данных (инфоблоки).

Метод	Описание
entity.add	Создает хранилище данных.
entity.update	Обновляет параметры хранилища данных.
entity.rights	Получение или изменение прав доступа к хранилищу.
entity.get	Получение параметров хранилища или списка всех хранилищ приложения.
entity.delete	Удаление хранилища.
entity.section.get	Получение списка разделов хранилища (секций инфоблока). Списочный метод.
entity.section.add	Добавление раздела хранилища.
entity.section.update	Обновление раздела хранилища.
entity.section.delete	Удаление раздела хранилища.
entity.item.get	Получение списка элементов хранилища. Списочный метод.
entity.item.add	Добавление элемента хранилища.

entity.item.update	Обновление элемента хранилища.
entity.item.delete	Удаление элемента хранилища.
entity.item.property.get	Получение списка дополнительных свойств элементов хранилища.
entity.item.property.add	Добавление дополнительного свойства элементов хранилища.
entity.item.property.update	Обновление дополнительного свойства элементов хранилища.
entity.item.property.delete	Удаление дополнительного свойства элементов хранилища.

Методы работы с Уведомлениями

Разрешение **Уведомления** (im)

Документацию по Chat API смотрите в курсе Бот платформа Битрикс24

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Чат-боты, сообщения и Открытые линии

Разрешение **Чат-боты** (imbot)

Чат-бот - это виртуальный собеседник, программа, которая создана для имитации поведения человека при общении с одним или несколькими собеседниками.

Что могут делать чат-боты?

- **Замена рутины** - позволяет выполнять определенные функции, не привлекая людей, а работа будет выполнена моментально и безупречно;
- **Поиск и агрегация** новостей, аналитики, данных (Data-Driven Collaboration), данные доступны в месте принятия решений - мессенджерах и всем участникам, которым они нужны;
- **E-commerce** - для спонтанных покупок без долгого поиска, mobile ecommerce + visual search + chatbots, для общения с клиентами;
- **Первая линия** работы с клиентами, помощники, консультанты, типовые вопросы, телефония;
- **Just for Fun** - просто для развлечения.

В курсе Бот платформа Битрикс24 дана исчерпывающая информация по API Чат-ботов, даны примеры и описание всего процесса создания чат-ботов.

Работа со списками

Rest-методы для работы с самими универсальными списками.

Метод	Описание	С версии
lists.add	Метод создаёт список.	
lists.delete	Метод удаляет список.	
lists.get	Метод возвращает данные по спискам.	
lists.update	Метод обновляет существующий список.	
lists.get.iblock.type.id	Метод возвращает id типа инфоблока.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Почтовые сервисы

Методы работы с почтовыми сервисами.

Метод	Описание	С версии
mailservice.fields	Возвращает описание полей почтового сервиса.	
mailservice.list	Возвращает список всех почтовых сервисов.	
mailservice.get	Возвращает параметры указанного почтового сервиса.	
mailservice.add	Добавляет почтовый сервис.	
mailservice.update	Обновляет параметры почтового сервиса.	
mailservice.delete	Удаляет почтовый сервис.	

© «Битрикс», 2001-2008, «1С-

Служба SMS сообщений

Метод	Описание	С верс
messageservice.sender.add	Метод регистрирует новый SMS-провайдер	
messageservice.sender.delete	Метод удаляет зарегистрированного SMS-провайдера	
messageservice.sender.list	Метод возвращает список зарегистрированных приложением SMS-провайдеров	
messageservice.message.status.update	Метод для управления статусами SMS для приложений.	17.5.0

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Мобильное приложение

Разрешение приложения - Мобильное приложение (**mobile**).

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Платёжные системы

Рест методы работы с платёжными системами и счетами.
Разрешение **pay_system**.

Метод	Описание	С версии
sale.paysystem.handler.add	Метод добавляет рест-обработчик.	
sale.paysystem.handler.delete	Метод удаляет рест-обработчик.	
sale.paysystem.handler.list	Метод для получение списка рест-обработчиков.	
sale.paysystem.handler.update	Метод выполняет обновление рест-обработчика.	
sale.paysystem.add	Метод добавляет платежную систему.	
sale.paysystem.pay.payment	Метод для оплаты заказа с использованием конкретной платёжной системы.	

sale.paysystem.delete	Метод удаляет платежную систему.	
sale.paysystem.list	Метод для получения списка платежных систем.	
sale.paysystem.update	Метод для редактирования платежной системы.	
sale.paysystem.pay.invoice	Метод для оплаты счета с использованием конкретной платёжной системы.	
sale.paysystem.settings.invoice.get	Метод для получения настроек платежной системы для конкретного счета.	
sale.paysystem.settings.payment.get	Метод для получения настроек платежной системы для конкретной оплаты.	

Pull&Push

REST-методы, доступные при работе с уведомлениями. Разрешение **pull**.

Смотри также

- Интерактивность в приложениях

© «Битрикс», 2001-2008, «1С-

Интернет-магазин

Пространства имен для rest-методов работы с Интернет-магазином.

Пространство	Описание
Вариант свойства	Методы работы с вариантами свойств.
Группы свойств	Методы работы с группами свойств.
Заказ	Методы работы с Заказом.
Заказы из внешних источников	Методы работы с Заказами из внешних источников.
Значения свойства	Методы работы со значениями свойств.
Кассы	Методы работы с кассами и чеками.
Корзина	Методы работы с Корзиной.
Локализация статусов	Методы работы с локализацией статусов.
Оплаты	Методы работы с Оплатами.
Отгрузки	Методы работы с Отгрузками.
Привязка свойства	Методы работы с привязкой свойств.
Свойства заказа	Методы работы со свойствами заказа.
Свойства корзины	Методы работы со свойствами корзины.
Свойства отгрузки	Методы работы со свойствами отгрузки.

Службы доставки	Методы и веб-хуки работы со службами доставки.
Соответствие физ. и юр. лицам	Методы работы с элементами соответствия физическим и юридическим лицам.
Статусы	Методы работы со статусами.
Табличная часть отгрузки	Методы работы с табличной частью отгрузки.
Типы плательщиков	Методы работы с Типами плательщиков.
Торговые платформы	Методы работы с Торговыми платформами.

Методы работы с группами соцсети

Разрешение **Рабочие группы соцсети** (sonet_group)

Методы

Метод	Описание
sonet_group.create	Создает группу соцсети, используя м CSocNetGroup::CreateGroup(), указы группы текущего пользователя.
sonet_group.delete	Удаляет группу соцсети.
sonet_group.feature.access	Проверяет, имеет ли текущий пользо совершение операции в группе соцсе вызов функции CSocNetFeaturesPerms::CurrentUserC
sonet_group.get	Возвращает массив групп соцсети, ка содержит массив полей, осуществляя CSocNetGroup::GetList(), при этом во те группы, которые доступны пользо
sonet_group.setowner	Изменяет владельца группы.
sonet_group.update	Изменяет параметры группы соцсети API CSocNetGroup::Update()
sonet_group.user.add	Добавляет пользователей в качестве группы (без приглашения и подтверж
sonet_group.user.delete	Удаляет пользователей из рабочей гр

sonet_group.user.get	Возвращает массив участников групп осуществляя вызов CSocNetUserToGr этом проверяются права на доступ те пользователя к группе.
sonet_group.user.groups	Возвращает массив групп соцсети те пользователя, осуществляя вызов CSocNetUserToGroup::GetList().
sonet_group.user.invite	Выполняет приглашение пользовател от лица текущего пользователя, при права на доступ текущего пользовате
sonet_group.user.request	Отправляет запрос текущего пользов вступление в группу соцсети, при это права на доступ текущего пользовате
sonet_group.user.update	Изменяет роль пользователей в рабо
socialnetwork.api.workgroup.list	Метод возвращает список групп.
socialnetwork.api.workgroup.get	Метод возвращает данные по рабоче
События при работе с группами СоцСети	Список событий при добавлении, изм группы.

Задачи

Описание классов для работы с задачами. Разрешение **task**.

Примечание: в связи с особенностями REST API задач, нельзя менять местами аргументы в методах, а также пропускать какие-либо из них.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Телефония

Методы REST для телефонии (scope telephony) позволяют обращаться к возможностям встроенной телефонии на базе VoxImplant, а также реализовывать интеграцию с любой внешней телефонией по основным пользовательским сценариям: автоматическое создание лида при входящем звонке с неизвестного номера, показ стандартной карточки звонка Битрикс24 и так далее.

Используя данное REST-API и набор интерфейсов, оператор телефонии может создать собственное приложение, позволяющее осуществить прозрачную интеграцию с порталом Битрикс24. Существует две возможности интеграции телефонии - использование REST API с использованием встроенной телефонии (voixplant), а также универсальный REST API для телефонии (telephony).

Клиенту не потребуется читать инструкции по настройке и разбираться, как это работает, он получит готовое решение из Маркетплейса и продолжит пользоваться телефонией в удобном для него окружении - сервисе Битрикс24.

В наборе интерфейсов содержится psd-файл для разработчиков приложения интеграции телефонии.

Смотри также

- Работа с js-библиотекой
- Общее описание REST

Учет рабочего времени

REST-методы для модуля **Учёт рабочего времени**.

Метод	Описание	С верс
Базовые методы		
timeman.settings	Получение настроек рабочего времени пользователя	17.0.2
timeman.status	Получение информации о текущем рабочем дне пользователя	17.0.2
timeman.open	Начать новый рабочий день либо возобновить закрытый или приостановленный	17.0.2
timeman.close	Закрыть рабочий день	17.0.2
timeman.pause	Приостановить рабочий день	17.0.2
Офисные сети		
timeman.networkrange.check	Метод для проверки IP-адреса на вхождение в	18.5.0

	диапазоны сетевых адресов офисной сети.	
timeman.networkrange.get	Метод для получения диапазонов сетевых адресов, входящих в офисную сеть.	18.5.0
timeman.networkrange.set	Метод для установки диапазонов сетевых адресов, входящих в офисную сеть.	18.5.0
Контроль времени		
timeman.timecontrol.report.add	Метод для отправки отчета о выявленном отсутствии..	18.5.0
timeman.timecontrol.reports.get	Метод для получения отчета о выявленных отсутствиях.	18.5.0
timeman.timecontrol.reports.settings.get	Метод для получения пользовательских настроек для построения интерфейса отчетов инструмента контроля времени.	18.5.0
timeman.timecontrol.reports.users.get	Метод для получения списка пользователей, относящихся к	18.5.0

	указанному подразделению.	
timeman.timecontrol.settings.get	Метод для получения настроек инструмента контроля времени.	18.5.0
timeman.timecontrol.settings.set	Метод для установки настроек инструмента контроля времени.	18.5.0
Рабочий график		
timeman.schedule.get	Метод позволяет получить рабочий график по его идентификатору.	

Работа с пользователями

Разрешение **Пользователи** (user)

Описание

Методы работы с пользователями Битрикс24 позволяют приглашать новых пользователей, изменять данные существующих пользователей и выбирать пользователей при помощи условий. Приложения, которые используют эти методы в своих сценариях, должны обеспечивать максимальную безопасность пользовательских данных и получать только ту информацию о пользователях, которая действительно необходима для работы приложения.

Чтобы гарантировать пользователям безопасность их персональной информации, существует несколько уровней доступа через методы работы с пользователями:

- Ограниченные версии доступа:
  - **user_brief**, который позволяет получать базовую информацию о пользователях, без их контактных данных и пользовательских полей. Этот скоуп необходим и достаточен для сценариев, в которых требуется отобразить ФИО пользователя в интерфейсе приложения.
  - **user_basic**, который позволяет получать не только базовую информацию, но и контактные данные пользователей Битрикс24. Этот скоуп нужен для сценариев, связанных с совершением звонков, или отправкой e-mail сообщений при помощи вашего приложения.
- Полные версии доступа:
  - **user**, который позволяет получить все стандартные поля, а кроме того, делает доступной возможность приглашения новых пользователей и изменение данных существующих пользователей.

- **user.userfield**, который открывает доступ к методам для работы с пользовательскими полями пользователей (расширяет перечень доступных полей в методах чтения, доступных в скоупах выше) для получения, добавления, изменения и удаления пользовательских полей.

  Внимание! Это максимальный уровень доступа к персональной информации, запрашивать его нужно очень ответственно.

Ограниченные версии скоупа user

В этих скоупах нельзя добавлять/обновлять пользователей: не доступны методы user.add и user.update. Во всех остальных методах получения информации о пользователе доступны только эти поля (с версии **Rest 21.600.0**):

user_basic	user_brief
ID	ID
XML_ID	XML_ID
ACTIVE	ACTIVE
NAME	NAME
LAST_NAME	LAST_NAME
SECOND_NAME	SECOND_NAME
TITLE	TITLE
EMAIL	IS_ONLINE
PERSONAL_PHONE	TIME_ZONE
WORK_PHONE	TIME_ZONE_OFFSET
WORK_POSITION	TIMESTAMP_X
WORK_COMPANY	DATE_REGISTER
IS_ONLINE	PERSONAL_PROFESSION
TIME_ZONE	PERSONAL_GENDER
TIMESTAMP_X	PERSONAL_BIRTHDAY
TIME_ZONE_OFFSET	PERSONAL_PHOTO
DATE_REGISTER	PERSONAL_CITY
LAST_ACTIVITY_DATE	PERSONAL_STATE
PERSONAL_PROFESSION	PERSONAL_COUNTRY
PERSONAL_GENDER	WORK_POSITION
PERSONAL_BIRTHDAY	WORK_CITY

- PERSONAL_PHOTO
- PERSONAL_PHONE
- PERSONAL_FAX
- PERSONAL_MOBILE
- PERSONAL_PAGER
- PERSONAL_STREET
- PERSONAL_MAILBOX
- PERSONAL_CITY
- PERSONAL_STATE
- PERSONAL_ZIP
- PERSONAL_COUNTRY
- PERSONAL_NOTES
- WORK_COMPANY
- WORK_DEPARTMENT
- WORK_POSITION
- WORK_WWW
- WORK_PHONE
- WORK_FAX
- WORK_PAGER
- WORK_STREET
- WORK_MAILBOX
- WORK_CITY
- WORK_STATE
- WORK_ZIP
- WORK_COUNTRY
- WORK_PROFILE
- WORK_LOGO
- WORK_NOTES
- UF_DEPARTMENT
- UF_DISTRICT
- UF_SKYPE
- UF_SKYPE_LINK
- UF_ZOOM
- UF_TWITTER
- UF_FACEBOOK*
- UF_LINKEDIN
- UF_XING
- UF_WEB_SITES
- WORK_STATE
- WORK_COUNTRY
- LAST_ACTIVITY_DATE
- UF_EMPLOYMENT_DATE
- UF_TIMEMAN
- UF_SKILLS
- UF_INTERESTS
- UF_DEPARTMENT
- UF_PHONE_INNER

- UF_PHONE_INNER
- UF_EMPLOYMENT_DATE
- UF_TIMEMAN
- UF_SKILLS
- UF_INTERESTS

* деятельность организации запрещена в Российской Федерации.

Полная версия скоупа user

В полной версии доступны поля (с версии **Rest 21.600.0**):

user
▪ ID ▪ XML_ID ▪ ACTIVE ▪ NAME ▪ LAST_NAME ▪ SECOND_NAME ▪ TITLE ▪ EMAIL ▪ LAST_LOGIN ▪ DATE_REGISTER ▪ TIME_ZONE ▪ IS_ONLINE ▪ TIME_ZONE_OFFSET ▪ TIMESTAMP_X ▪ LAST_ACTIVITY_DATE ▪ PERSONAL_PROFESSION ▪ PERSONAL_GENDER ▪ PERSONAL_WWW ▪ PERSONAL_BIRTHDAY ▪ PERSONAL_PHOTO

- PERSONAL_ICQ
- PERSONAL_PHONE
- PERSONAL_FAX
- PERSONAL_MOBILE
- PERSONAL_PAGER
- PERSONAL_STREET
- PERSONAL_MAILBOX
- PERSONAL_CITY
- PERSONAL_STATE
- PERSONAL_ZIP
- PERSONAL_COUNTRY
- PERSONAL_NOTES
- WORK_COMPANY
- WORK_DEPARTMENT
- WORK_POSITION
- WORK_WWW
- WORK_PHONE
- WORK_FAX
- WORK_PAGER
- WORK_STREET
- WORK_MAILBOX
- WORK_CITY
- WORK_STATE
- WORK_ZIP
- WORK_COUNTRY
- WORK_PROFILE
- WORK_LOGO
- WORK_NOTES
- UF_DEPARTMENT
- UF_DISTRICT
- UF_SKYPE
- UF_SKYPE_LINK
- UF_ZOOM
- UF_TWITTER
- UF_FACEBOOK*
- UF_LINKEDIN
- UF_XING
- UF_WEB_SITES
- UF_PHONE_INNER

- UF_EMPLOYMENT_DATE
- UF_TIMEMAN
- UF_SKILLS
- UF_INTERESTS

* деятельность организации запрещена в Российской Федерации.

Методы

Метод	Описание
user.fields	Получение списка названий полей пользователя.
user.current	Получение информации о текущем пользователе.
user.add	Приглашает пользователя.
user.update	Обновляет данные пользователя.
user.get	Получение фильтрованного списка пользователей.
user.search	Получение списка пользователей с ускоренным поиском по персональным данным.

Работа с соглашениями

Разрешение **Соглашение пользователя** (userconsent).

Методы

Метод	Описание
userconsent.agreement.list	Получение списка соглашений.
userconsent.agreement.text	Получение текста соглашения.
userconsent.consent.add	Сохранение полученного согласия пользователя.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

Список событий.

Примечание. При работе с методами telephony можно использовать события класса voximplant.

Событие	Описание
OnExternalCallStart	Вызывается, когда пользователь нажимает н объектах CRM для совершения исходящего з Чтобы событие срабатывало надо зайти в Тел телефонии и в поле **Номер для исходящег** выбрать свое приложение. Либо выбрать это приложение как номер по пользователя. Событие будет срабатывать то приложения.

Параметр	Описание
USER_ID	Идентификатор пользователя.
PHONE_NUMBER	Номер исходящего зв
CRM_ENTITY_TYPE	Тип объекта CRM, из карточки которого совершается звонок - CONTACT | COMPANY
CRM_ENTITY_ID	Идентификатор объек

	тип которого указан в CRM_ENTITY_TYPE.
CALL_LIST_ID	Идентификатор списк обзвона, в случае есл звонок совершается и списка обзвона.
LINE_NUMBER	Номер внешней лини которую запрошен зв
CALL_ID	Идентификатор звонк метода telephony.externalcall.
IS_MOBILE	(Y|N) Признак того, ч звонок инициирован мобильного приложен

OnExternalCallBackStart

Вызывается, когда посетитель заполняет crm В настройках формы должно быть выбрано в линия, через которую будет совершаться обр

Параметр	Описание
PHONE_NUMBER	Номер телефона.
TEXT	Текст, который долже произнесен пользоват при начале звонка (и настроек формы).
VOICE	Идентификатор голос которым должен быть произнесен текст (из настроек формы). Дл получения списка идентификаторов гол voximplant.tts.voices.
CRM_ENTITY_TYPE	Тип связанной сущно

	CRM.
CRM_ENTITY_ID	Идентификатор сущно CRM, тип которого ука CRM_ENTITY_TYPE.
LINE_NUMBER	Номер внешней линии которую запрошен об звонок

© «Битрикс», 2001-2008, «1С-

События

Интерфейс REST позволяет устанавливать свои обработчики серверных событий.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Подключение библиотеки

Если ваше приложение реализует пользовательский интерфейс внутри Битрикс24 (выводится на специальной странице или использует инструменты встройки), то вы можете воспользоваться возможностями специальной JS-библиотеки. Она представляет собой,
во-первых, JS SDK для REST, что позволяет обращаться к REST прямо из front-end вашего приложения не погружаясь в реализацию авторизации по OAuth 2.0,

а во-вторых, предлагает ряд дополнительных функций для взаимодействия пользовательского интерфейса вашего приложения с интерфейсом Битрикс24:

- управление размером фрейма,
- вызов стандартных диалогов,
- и т.п.

Подключается библиотека следующим образом:

```
<script src="//api.bitrix24.com/api/v1/"></script>
```

Для внешних приложений и вебхуков библиотека использоваться не может.

Системные функции

Функция	Описание
BX24.init	Добавляет в список обработчик события "библиотека готова к работе".
BX24.install	Возможность установить обработчик события "приложение запускается первый раз для текущего пользователя".
BX24.installFinish	Функция, сигнализирующая об окончании работы инсталлятора или настройщика приложения.
BX24.getAuth	Получение текущих данных для авторизации через [link=700007]OAuth 2.0[/link].
BX24.refreshAuth	Принудительное обновление ключа авторизации.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Вызов методов REST

Вызов метода **REST** представляет собой кросс-доменный запрос к REST-сервису текущего **Битрикс24** от имени текущего пользователя. Авторизация запроса осуществляется автоматически на основе протокола OAuth 2.0.

- Отправка запроса
- Обработка результата запроса
- Пакетное выполнение запросов
- Обработка файлов

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

Событие	Вызывается
OnAppInstall	при установке приложения
OnAppUninstall	при удалении приложения
OnAppPayment	при оплате приложения
OnAppMethodConfirm	при получении решения администратора портала по запросу на использование методов, требующих подтверждения
OnUserAdd	при добавлении пользователя в Битрикс24

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Настройки приложения

Для работы с настройками приложения используются два объекта.

Объект	Описание
BX24.userOption	Работа с настройками текущего пользователя.
BX24.appOption	Работа с общими настройками приложения.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Показ системных диалогов

Системные диалоги работают и отображаются вне приложения. Приложение имеет к ним только косвенный доступ через обработчик события выбора значения.

Функция	Описание
BX24.selectUser	Показать стандартный диалог одиночного выбора пользователя.
BX24.selectUsers	Показать стандартный диалог множественного выбора пользователей.
BX24.selectAccess	Показать стандартный диалог выбора прав доступа.
BX24.selectCRM	Показать стандартный диалог выбора сущности CRM.

Примеры

Открыть из приложения определенный чат

```
BX24.im.openMessenger(dialogId)
```

Открыть из приложения историю чата

```
BX24.im.openHistory(dialogId)
```

Позвонить пользователю портала

```
BX24.im.callTo(userId[, video=true])
```

Позвонить по номеру телефона

```
BX24.im.phoneTo(phone)
```

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Дополнительные методы

Метод	Описание
BX24.isAdmin	Определяет имеет ли текущий пользователь права на управление приложениями.
BX24.getLang	Возвращает идентификатор языка текущего портала.
BX24.resizeWindow	Изменяет размер фрейма с приложением.
BX24.fitWindow	Устанавливает размер фрейма с приложением в соответствии с размерами содержимого фрейма.
BX24.reloadWindow	Перезагружает страницу с приложением (всю страницу, не только фрейм).
BX24.setTitle	Устанавливает заголовок страницы.
BX24.ready	Устанавливает обработчик события "DOM-структура документа готова к работе".
BX24.isReady	Флаг "DOM-структура документа готова к работе".
BX24.proxy	Аналогична BX.proxy. Аналог $.proxy, но с одним отличием: при повторном вызове функции с теми же параметрами будет возвращена

	ссылка на ту же прокси-функцию, которая была результатом первого вызова, а не новая прокси-функция.
BX24.proxyContext	При вызове изнутри прокси-функцию выдаст ссылку на оригинальный контекст выполнения прокси-функции.
BX24.bind	Устанавливает функцию *func* в качестве обработчика события *eventName* объекта *element*.
BX24.unbind	Убирает функцию *func* в качестве обработчика события *eventName* объекта *element*.
BX24.getDomain	Возвращает **window.location.host** родительского окна, т.е. возвращает адрес портала *Битрикс24*.
BX24.getScrollSize	Функция возвращает внутренние размеры фрейма приложения.
BX24.loadScript	Загружает и выполняет клиентский javascript-файл.
BX24.im.callTo	Звонок по внутренней связи.
BX24.im.phoneTo	Звонок на телефонный номер.
BX24.im.openMessenger	Открытие окна мессенджера.
BX24.im.openHistory	Открытие окна истории.
BX24.openApplication	Метод открывает приложение.
BX24.closeApplication	Метод закрывает открытое модальное окно с приложением.
BX24.scrollParentWindow	Метод прокручивает родительское окно.

Общие методы

Несколько общих методов, доступных приложению независимо от запрашиваемых приложением разрешений.

Метод	Описание
methods	Показ списка методов.
scope	Показ разрешений.
app.info	Показ информации о приложении.
batch	Выполнение пакета запросов.
user.admin	Определяет обладает ли текущий пользователь правами на управление настройками приложений.
user.access	Определяет обладает ли текущий пользователь хотя бы одним из заданного параметром *ACCESS* набора прав.
access.name	Получение названий прав доступа.
server.time	Возвращает текущее время сервера в формате ATOM.
profile	Позволяет получить базовую информации о текущем пользователе без каких-либо скоупов.
events	Показ общего списка событий.
event.bind	Регистрация нового обработчика события.
event.unbind	Отмена зарегистрированного обработчика

	события.
events.get	Получение списка зарегистрированных обработчиков событий.
event.offline.get	Метод возвращает приложению первые в очереди офлайновые события.
event.offline.error	Сохранение записи в базе с пометкой об ошибке при использовании офлайновых событий.
event.offline.clear	Очистка записей в очереди офлайн событий.
event.offline.list	Чтение текущей очереди без внесения изменений в ее состояние.

Смотри также

- Авторизация

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Методы событий

Методы для работы с событиями.

Функция	Описание
events	Показ общего списка событий.
event.get	Получение списка зарегистрированных обработчиков событий.
event.bind	Регистрация нового обработчика события.
event.unbind	Отмена зарегистрированного обработчика события.
event.offline.get	Метод возвращает приложению первые в очереди офлайновые события.
event.offline.error	Сохранение записи в базе с пометкой об ошибке при использовании офлайновых событий.
event.offline.clear	Очистка записей в очереди офлайн событий.
event.offline.list	Чтение текущей очереди без внесения изменений в ее состояние.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Настройка приложений

Методы работы с приложениями

Метод	Описание	С версии
app.option.set	Метод привязывает данные к приложению.	
app.option.get	Получает данные, привязанные к приложению.	
user.option.*	Методы привязки данных к приложению и пользователю	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Частые кейсы

- Добавление лидов и других сущностей
- Редактирование
- Получение списка
- Сквозная аналитика

- Как отправить e-mail клиенту от имени сотрудника с указанием ящика сотрудника
- Получение воронки заданного направления с семантикой каждой стадии сделки
- Генератор документов в crm: создание "путевого листа" по данным сделки
- Встройка в лид в виде пользовательского свойства

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Добавление лидов и других сущностей

- Простое добавление лида
- Добавление лида с учетом режимов CRM (простой или классический режим)
- Добавление повторного лида
- Простое добавление контакта
- Добавление контакта с реквизитами
- Простое добавление компании
- Добавление компании с реквизитами
- Добавление сделки с выбором реквизитов существующей компании или контакта
- Как добавить клиенту событие календаря
- Добавление сделки (лида, счета, компреда) с товарами, с применением скидок и налогов
- Как добавить товар со значениями пользовательских полей

Редактирование

- Добавление картинок в поля лида
- Как сделать свою карточку редактирования лида
- Как сделать свою карточку редактирования контакта
- Как сделать свою карточку редактирования компании
- Как сделать свою карточку редактирования сделки
- Как изменить номера телефонов и e-mail на примере контакта
- Как изменить даты в деле-событии
- Как изменить значения пользовательских полей товара

© «Битрикс», 2001-2008, «1С-

Получение списка

- Поиск в CRM по телефону и e-mail
- Как получить список дел на примере контакта
- Получение списка статусов лидов с семантикой
- Получение списка статусов коммерческих предложений

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Сквозная аналитика

- Использование сквозной аналитики при создании лида
- Использование сквозной аналитики при создании сделки и контакта
- Как передать информацию в Сквозную аналитику

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Лиды

Функция	Описание
crm.lead.add	Создаёт новый лид.
crm.lead.delete	Удаляет лид и все связанные с ним объекты.
crm.lead.fields	Возвращает описание полей лида.
crm.lead.get	Возвращает лид по идентификатору.
crm.lead.list	Возвращает список лидов по фильтру.
crm.lead.productrows.get	Возвращает товарные позиции лида.
crm.lead.productrows.set	Устанавливает (создаёт или обновляет) товарные позиции лида.
crm.lead.update	Обновляет существующий лид.
crm.lead.userfield.add	Создаёт новое пользовательское поле для лидов.
crm.lead.userfield.get	Возвращает пользовательское поле лидов по идентификатору.
crm.lead.userfield.list	Возвращает список пользовательских полей лидов по фильтру.
crm.lead.userfield.update	Обновляет существующее

	пользовательское поле лидов.
crm.lead.userfield.delete	Удаляет пользовательское поле лидов.

Общий список **событий лидов** приведен здесь.

Дополнительно

- Как создать лид из других сервисов.

События

Событие	Вызывается
onCrmLeadAdd	при создании лида
onCrmLeadUpdate	при обновлении лида
onCrmLeadDelete	при удалении лида
onCrmLeadUserFieldAdd	при добавлении пользовательского поля
onCrmLeadUserFieldUpdate	при изменении пользовательского поля
onCrmLeadUserFieldDelete	при удалении пользовательского поля
onCrmLeadUserFieldSetEnumValues	при изменении набора значений для пользовательского поля списочного типа

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

Событие	Вызывается
onCrmCompanyAdd	при создании компании
onCrmCompanyUpdate	при обновлении компании
onCrmCompanyDelete	при удалении компании
onCrmCompanyUserFieldAdd	при добавлении пользовательского поля
onCrmCompanyUserFieldUpdate	при изменении пользовательского поля
onCrmCompanyUserFieldDelete	при удалении пользовательского поля
onCrmCompanyUserFieldSetEnumValues	при изменении набора значений для пользовательского поля списочного типа

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Контакты

Функция	Описание
crm.contact.add	Создаёт новый контакт.
crm.contact.company.add	Добавляет компанию к указанному контакту.
crm.contact.company.delete	Удаляет компанию из указанного контакта.
crm.contact.company.fields	Возвращает описание полей для связи контакт-компания.
crm.contact.company.items.delete	Очищает набор компаний, связанных с указанным контактом.
crm.contact.company.items.get	Возвращает набор компаний, связанных с указанным контактом.
crm.contact.company.items.set	Устанавливает набор компаний, связанных с указанным контактом.
crm.contact.delete	Удаляет контакт и все связанные с ним объекты.
crm.contact.fields	Возвращает описание полей контакта.
crm.contact.get	Возвращает контакт по идентификатору.

crm.contact.list	Возвращает список контактов по фильтру.
crm.contact.update	Обновляет существующий контакт.
crm.contact.userfield.add	Создаёт новое пользовательское поле для контактов.
crm.contact.userfield.get	Возвращает пользовательское поле контактов по идентификатору.
crm.contact.userfield.list	Возвращает список пользовательских полей контактов по фильтру.
crm.contact.userfield.update	Обновляет существующее пользовательское поле контактов.
crm.contact.userfield.delete	Удаляет пользовательское поле контактов.

Общий список **событий контакта** приведен здесь.

События

Событие	Вызывается
onCrmContactAdd	при создании контакта
onCrmContactUpdate	при обновлении контакта
onCrmContactDelete	при удалении контакта
onCrmContactUserFieldAdd	при добавлении пользовательского поля
onCrmContactUserFieldUpdate	при изменении пользовательского поля
onCrmContactUserFieldDelete	при удалении пользовательского поля
onCrmContactUserFieldSetEnumValues	при изменении набора значений для пользовательского поля списочного типа

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Направления

Методы	Описание
crm.category.fields	Отдает информацию о полях направлений.
crm.category.get	Отдает информацию о направлении с идентификатором `id`.
crm.category.list	Возвращает массив направлений, которые относятся к типу сущности с идентификатором `entityTypeId`.
crm.category.add	Создаёт у типа сущности с идентификатором `entityTypeId` новое направление с полями `fields`.
crm.category.update	Обновляет направление с идентификатором `id`, задав ему новые значения полей из `fields`.
crm.category.delete	Удаляет направление с идентификатором `id`, которое относится к типу сущности с идентификатором `entityTypeId`.

Дополнительно

- Настройки пользовательских полей

Направления сделок

Методы	Описание
crm.dealcategory.add	Создаёт новое направление сделок.
crm.dealcategory.delete	Удаляет направление сделок.
crm.dealcategory.fields	Возвращает описание полей направления сделок.
crm.dealcategory.get	Возвращает направление сделок по идентификатору.
crm.dealcategory.list	Возвращает список направлений сделок по фильтру. Является реализацией списочного метода для направлений сделок.
crm.dealcategory.stage.list	Возвращает список стадий сделок для направления по идентификатору.
crm.dealcategory.status	Возвращает идентификатор типа справочника для хранения стадий по идентификатору направления сделок.
crm.dealcategory.update	Обновляет существующее направление.
crm.dealcategory.default.set	Метод записи настроек общего направления сделок.

crm.dealcategory.default.get	Метод чтения настроек общего направления сделок.

Справочники

Методы

Метод	Описание
crm.status.add	Создаёт новый элемент в указанном справочнике.
crm.status.delete	Удаляет элемент справочника.
crm.status.entity.items	Возвращает элементы справочника по его символьному идентификатору
crm.status.entity.types	Возвращает описание типов справочников
crm.status.fields	Возвращает описание полей справочника.
crm.status.get	Возвращает элемент справочника по идентификатору.
crm.status.list	Возвращает список элементов справочника по фильтру.
crm.status.update	Обновляет существующий элемент справочника.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Статусы счетов

С версии 19.0.0 рекомендуется использовать общие методы работы со справочниками.

Функция	Описание
crm.invoice.status.fields	Возвращает описание полей статуса счёта.
crm.invoice.status.list	Возвращает список статусов счёта.
crm.invoice.status.get	Возвращает статус счёта по идентификатору.
crm.invoice.status.add	Создаёт новый статус счёта.
crm.invoice.status.update	Обновляет существующий статус счёта.
crm.invoice.status.delete	Удаляет статус счёта.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Предложения

Функция	Описание
crm.quote.fields	Возвращает описание полей коммерческого предложения.
crm.quote.add	Набор полей.
crm.quote.get	Возвращает коммерческое предложение по идентификатору.
crm.quote.list	Возвращает список предложений по фильтру. Является реализацией списочного метода для предложений.
crm.quote.update	Обновляет существующее предложение.
crm.quote.delete	Удаляет предложение и все связанные с ним объекты.
crm.quote.productrows.get	Возвращает товарные позиции предложения.
crm.quote.productrows.set	Устанавливает (создаёт или обновляет) товарные позиции предложения.
crm.quote.userfield.add	Создаёт новое пользовательское поле для предложений.
crm.quote.userfield.delete	Удаляет пользовательское поле предложений.

crm.quote.userfield.get	Возвращает пользовательское поле предложений по идентификатору.
crm.quote.userfield.list	Возвращает список пользовательских полей предложений по фильтру.
crm.quote.userfield.update	Обновляет существующее пользовательское поле предложений.

© «Битрикс», 2001-2008, «1С-

Настройки пользовательских полей

На данный момент это API доступно только для модулей rpa и crm.

Физически контроллер находится в модуле **main**. Но он вынесен как отдельный **scope** для REST (через AJAX можно обращаться как *main.userfieldconfig.**).

> Чтобы методы работали приложение должно иметь доступ не только к скоупу `userfieldconfig`, но и к скоупу, для которого выполняются изменения настроек полей. Например, чтобы изменить настройки полей в модуле Роботизация бизнеса, приложение должно иметь доступ и к **userfieldconfig**, и к **rpa**

При обращении к любому из методов необходимо передавать идентификатор модуля **moduleId**, по этому идентификатору определяется доступ пользователя к полям.

Дополнительно

- Типы пользовательских полей и их настройки в главном модуле Bitrix Framework.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Настройка карточек сущности

Метод	Описан
crm.lead.details.configuration.get	Метод для получения параметро настройки карточки лидов.
crm.lead.details.configuration.set	Метод установли настройки карточки лидов.
crm.lead.details.configuration.reset	Метод сбрасывае настройки карточки лидов.
crm.lead.details.configuration.forceCommonScopeForAll	Метод принудите устанавли общую карточку лидов для пользовате
crm.deal.details.configuration.get	Метод для получения

	настроек карточки сделок.
crm.deal.details.configuration.set	Метод позволяет установить настройки карточки сделок.
crm.deal.details.configuration.reset	Метод для сброса настроек карточки сделок.
crm.deal.details.configuration.forceCommonScopeForAll	Метод принудите устанавли общую карточку сделок для всех пользовате
crm.contact.details.configuration.get	Метод для получения настроек карточки контактов.
crm.contact.details.configuration.set	Метод устанавли настройки карточки контактов.
crm.contact.details.configuration.reset	Метод для сброса настроек карточки контактов.
crm.contact.details.configuration.forceCommonScopeForAll	Метод позволяет

	принудите установить общую карточку контакта д всех пользовате
crm.company.details.configuration.get	Метод для получения настроек карточки компаний.
crm.company.details.configuration.set	Метод устанавли настройки карточки компаний.
crm.company.details.configuration.reset	Метод для сброса настроек карточки компаний.
crm.company.details.configuration.forceCommonScopeForAll	Метод позволяет принудите установить общую карточку компаний всех пользовате

Методы настроек смарт-процессов

Методы	Описание
crm.type.fields	Отдает информацию о собственных полях настроек смарт-процесса.
crm.type.get	Отдает информацию о смарт-процессе.
crm.type.list	Возвращает массив настроек смарт-процессов.
crm.type.add	Создаёт новый смарт-процесс.
crm.type.update	Обновляет существующие настройки смарт-процесса.
crm.type.delete	Удаляет существующие настройки смарт-процесса.

Дополнительно

- Настройки пользовательских полей

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Элементы смарт-процессов

Описание

Так как элементы каждого смарт-процесса хранятся в отдельной таблице, то идентификаторы элементов разных смарт-процессов могут совпадать.

Поэтому во все методы необходимо передавать внешний идентификатор смарт-процесса `entityTypeId`.

Права доступа

При обращении к методам REST учитываются права доступа пользователя, от которого осуществляется вызов методов.

Булевы значения

Значения некоторых полей имеют булевый тип (например, поле `opened` - "Доступно для всех").

В этом случае для изменения значения надо передавать "Y" или "N". В ответах на запросы они будут отображаться аналогично.

Привязка к реквизитам

Для управления привязкой реквизитов можно воспользоваться методами crm.requisite.link.*.

Особенности работы с названиями полей

В базе данных имена полей элементов хранятся в `UPPER_CASE` формате. Но при работе через REST мы преобразуем их в `camelCase`. Например, поле `ASSIGNED_BY_ID` преобразуется в `assignedById`.

Имена пользовательских полей, помимо символа _ и букв, могут содержать цифры. Обычно они идут раздельно, например, поле `UF_CRM_10_5186744711` преобразуется в `ufCrm10_5186744711`.

Чтобы код выглядел более читаемым, подчеркивания между цифрами остаются, а остальные - убираются.

Проблемы начинаются, когда буквы и цифры в поле идут вперемешку. Например, `UF_CRM_10_DIGIT10` преобразуется в `ufCrm10Digit10`. При обратном преобразовании нет никакой возможности определить, исходное поле называлось `UF_CRM_10_DIGIT10` или `UF_CRM_10_DIGIT_10`.

Чтобы разрешить подобные конфликты, с версии **CRM 21.1800.0** был внедрен механизм предварительного анализа названий полей, и если в них обнаружены конфликты, то название поля не преобразуется в `camelCase`, а остается как есть.

Кроме того, любые названия полей во всех методах можно передавать как в `UPPER_CASE`, так и в `camelCase`.

Например, у элемента есть два поля - `UF_CRM_10_DIGIT_10` и `UF_CRM_10_DIGIT10`.

При вызове метода `crm.item.fields` название первого поля будет преобразовано в `ufCrm10Digit10`, а второе останется `UF_CRM_10_DIGIT10`. При этом по ключу `upperName` можно прочитать название поля в `UPPER_CASE` и во всех методах использовать его.

Методы

Методы	Описание
crm.item.fields	Отдает информацию о полях смарт-процесса.
crm.item.get	Отдает информацию об элементе смарт-процесса.
crm.item.list	Возвращает массив элементов смарт-процесса.

crm.item.add	Создаёт создает новый элемент смарт-процесса.
crm.item.update	Обновляет элемент смарт-процесса.
crm.item.delete	Удаляет элемент смарт-процесса.

С версии crm 21.800.0

Появляется поддержка полей-родителей при чтении и изменении элементов смарт-процессов.

Каждое поле имеет код `parentId + {parentEntityTypeId}`.

- Метод `fields` в списке полей будет отдавать информацию о полях с родителями.
- Методы `get` и `list` будут отдавать значения полей родителей.
- Методы `add` и `update` будут поддерживать изменения значений этих полей.

Дополнительно

- Настройки пользовательских полей

Дела

Дела описывают выполненную, текущую и запланированную работу по лидам, контактам, компаниям и сделкам. Делятся на звонки, встречи и e-mail сообщения.

Функция	Описание
crm.activity.add	Создаёт новое дело.
crm.activity.communication.fields	Возвращает описание коммуникации для дела.
crm.activity.delete	Удаляет дело.
crm.activity.fields	Возвращает описание полей дела.
crm.activity.get	Возвращает дело по идентификатору.
crm.activity.list	Возвращает список дел по фильтру.
crm.activity.update	Обновляет существующую дело.
crm.activity.type.add	Метод для регистрации своего подтипа дел с указанием ему названия и иконки.
crm.activity.type.delete	Метод для удаления подтипа дел.
crm.activity.type.list	Метод для получения списка

	подтипов дел.

Общий список **событий дел** приведен здесь.

Встраивание приложений

Внешние приложения могут добавлять свои элементы в интерфейс Дел. См. также раздел Встраивание приложений.

Места встраивания

Код	Описание
	Создание дел из приложений
CRM_ACTIVITY_LIST_MENU	Контекстное меню дел

© «Битрикс», 2001-2008, «1С-

События

Событие	Вызывается
onCrmActivityAdd	при создании дела
onCrmActivityUpdate	при обновлении дела
onCrmActivityDelete	при удалении дела

© «Битрикс», 2001-2008, «1С-

Реквизиты

- Методы
- [link=105293]События[/link]

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Методы

Методы	Описание
crm.address.add	Создаёт новый адрес для реквизита.
crm.address.delete	Удаляет адрес для реквизита.
crm.address.fields	Возвращает описание полей адреса.
crm.address.list	Возвращает список адресов по фильтру.
crm.address.update	Обновляет адрес для реквизита.
crm.requisite.add	Создаёт новый реквизит.
crm.requisite.bankdetail.add	Создаёт новый банковский реквизит.
crm.requisite.bankdetail.delete	Удаляет банковский реквизит.
crm.requisite.bankdetail.fields	Возвращает описание полей банковских реквизитов.
crm.requisite.bankdetail.get	Возвращает банковский реквизит по идентификатору.

crm.requisite.bankdetail.list	Возвращает список банковских реквизитов по фильтру.
crm.requisite.bankdetail.update	Обновляет существующий банковский реквизит.
crm.requisite.delete	Удаляет реквизит и все связанные с ним объекты.
crm.requisite.fields	Возвращает описание полей реквизита.
crm.requisite.get	Возвращает реквизит по идентификатору.
crm.requisite.list	Возвращает список реквизитов по фильтру.
crm.requisite.link.fields	Возвращает описание полей для связей реквизитов.
crm.requisite.link.get	Возвращает связь реквизитов с сущностью.
crm.requisite.link.list	Возвращает список связей реквизитов.
crm.requisite.link.register	Регистрирует связь реквизитов с сущностью.
crm.requisite.link.unregister	Удаляет связь реквизитов с сущностью.
crm.requisite.preset.add	Создаёт новый шаблон.

crm.requisite.preset.countries	Возвращает возможный список стран для шаблонов реквизита.
crm.requisite.preset.delete	Удаляет шаблон реквизита по идентификатору.
crm.requisite.preset.field.add	Добавляет поле в набор полей шаблона, связанного с реквизитом.
crm.requisite.preset.field.availabletoadd	Возвращает доступные поля из набора шаблона для определенного реквизита.
crm.requisite.preset.field.delete	Удаляет поле из набора полей шаблона для определенного реквизита.
crm.requisite.preset.field.fields	Возвращает описание набора полей шаблона.
crm.requisite.preset.field.get	Возвращает описание поля из набора полей шаблона для определенного реквизита.
crm.requisite.preset.field.list	Возвращает список полей из набора полей шаблона для определенного реквизита.
crm.requisite.preset.field.update	Обновляет поле из набора полей

	шаблона, связанного с реквизитом.
crm.requisite.preset.fields	Возвращает описание полей шаблона реквизитов.
crm.requisite.preset.get	Возвращает шаблон реквизита по идентификатору.
crm.requisite.preset.list	Возвращает список шаблонов по фильтру.
crm.requisite.preset.update	Обновляет шаблон реквизита.
crm.requisite.userfield.add	Создаёт новое пользовательское поле для реквизита.
crm.requisite.userfield.delete	Удаляет пользовательское поле реквизита.
crm.requisite.userfield.get	Возвращает пользовательское поле реквизита по идентификатору.
crm.requisite.userfield.list	Возвращает список пользовательских полей реквизита по фильтру.
crm.requisite.userfield.update	Обновляет существующее пользовательское поле реквизита..
crm.requisite.update	Обновляет существующий реквизит.

Генератор документов

Описание модуля Генератор документов приведено в документации по D7.

REST-методы CRM для документов реализованы через аякс-контроллеры. При этом сами контроллеры и их экшны реализованы в модуле **Генератор документов**, а в модуле CRM только обвязка для них. Эта обвязка преобразует входные и выходные параметры в соответствии с новым стандартом REST, а также учитывая особенности модуля CRM.

При работе с методами документов через скоуп CRM есть возможность работать только с шаблонами / документами, привязанными к модулю CRM.

Особенности передаваемых значений

Провайдеры

Модуль **Генератор документов** в качестве идентификатора провайдера данных использует полное наименование класса. Т.к. в модуле CRM используется именованные / числовые идентификаторы для идентификации сущностей, в RESTe для документов был использован аналогичный синтаксис. Если входной параметр называется *entityTypeId*, то он принимает числовой индекс сущностей CRM. На данный момент есть следующие идентификаторы:

1 - лид

2 - сделка

3 - контакт

4 - компания

5 - счет

6 - предложение

14 - заказ

Фильтрация по направлениям сделок

Методы REST не учитывают настройку привязки шаблона к провайдерам. Т.е. если был отправлен запрос на генерацию документа по шаблону с провайдером, к которому этот шаблон не привязан, ошибки не будет. Отсюда же следует, что не учитывается привязка шаблона к определенному направлению сделок. Если вы хотите указать сделку в качестве провайдера, всегда указывается только числовой идентификатор (2).

Список регионов

Каждый шаблон привязан к определенной стране. Список стран фиксирован и на данный момент состоит из:

ru - Россия

by - Беларусь

kz - Казахстан

ua - Украина

br - Бразилия

mx - Мексика

de - Германия

uk - Великобритания

pl - Польша

Управление регионами осуществляется в разделе documentgenerator .

Документы

Методы	Описание
crm.documentgenerator.document.get	Возвращает информацию о документе по его идентификатору.
crm.documentgenerator.document.getfields	Возвращает список полей документа с их описанием.
crm.documentgenerator.document.list	Возвращает список документов по фильтру.
crm.documentgenerator.document.delete	Удаляет документ.
crm.documentgenerator.document.add	Создает новый документ на основании шаблона и данных из соответствующей сущности.
crm.documentgenerator.document.update	Обновляет существующий документ с новыми значениями.
crm.documentgenerator.document.enablepublicurl	Включает /

	выключает публичную ссылку на документ.
crm.documentgenerator.document.upload	Загружает сформированный документ и прикрепляет его к указанной сущности.

Шаблоны документов

Методы	Описание
crm.documentgenerator.template.get	Возвращает информацию о шаблоне по его идентификатору.
crm.documentgenerator.template.getfields	Возвращает список полей шаблона с их описанием.
crm.documentgenerator.template.list	Возвращает список шаблонов по фильтру.
crm.documentgenerator.template.delete	Удаляет шаблон.
crm.documentgenerator.template.add	Добавляет новый шаблон.
crm.documentgenerator.template.update	Обновляет существующий шаблон.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Нумераторы

Методы	Описание
crm.documentgenerator.numerator.get	Возвращает информацию о нумераторе по его идентификатору.
crm.documentgenerator.numerator.list	Возвращает список нумераторов.
crm.documentgenerator.numerator.add	Добавляет новый нумератор.
crm.documentgenerator.numerator.update	Обновляет существующий нумератор с новыми значениями.
crm.documentgenerator.numerator.delete	Удаляет нумератор.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Пользовательские поля

Функция	Описание
crm.userfield.fields	Возвращает описание полей для пользовательских полей.
crm.userfield.types	Возвращает список типов пользовательских полей.
crm.userfield.enumeration.fields	Возвращает описание полей для пользовательского поля типа "enumeration" (список).
crm.userfield.settings.fields	Возвращает описание полей настроек для типа пользовательского поля.

Дополнительно

- Настройки пользовательских полей

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Встраивание в виде пользовательских типов полей

Приложения, имеющие доступ к скоупу **placement** могут регистрировать свои типы пользовательских полей.

На данный момент облачные порталы поддерживают работу таких полей в новой и старой карточке сущностей CRM. Приложения могут создавать пользовательские поля стандартных типов и тех, которые зарегистрированы этим приложением. Администраторы портала могут создавать поля любых зарегистрированных типов, включая типы полей, зарегистрированных приложениями. При регистрации типа приложение указывает адрес обработчика, который будет открываться в фрейме по месту вывода поля, и дальнейшая работа практически ничем не отличается от работы обычной встройки.

Метод	Описание	С версии
userfieldtype.add	Регистрация нового типа пользовательских полей.	
userfieldtype.list	Получение списка зарегистрированных приложением типов пользовательских полей.	
userfieldtype.update	Изменение настроек зарегистрированного приложением типа пользовательских полей.	

userfieldtype.delete	Удаление зарегистрированного приложением типа пользовательских полей.	

Дополнительно

1. Подробнее о встройке в виде пользовательских типах полей, читайте в учебном курсе

2. Вы зарегистрировали новый тип пользовательских полей. Если при попытке создать поле с новым типом:

1. userfieldtype.list возвращает ваш новый тип пользовательского поля
2. и, тем не менее, у вас возникает ошибка: `Error! 400: ERROR_CORE: Указан неверный пользовательский тип. (400)`

, то это значит, что [ds]приложение[/ds][di]Под локальными приложениями понимаются приложения, которые описываются и добавляются непосредственно на конкретный Битрикс24.

Подробнее ...[/di] не установлено полностью. Вызовите метод app.info и убедитесь что в результате вернулось `INSTALLED = true`. Если приложение с интерфейсом, то для завершения установки необходимо на странице приложения выполнить js код:

```
BX24.init(function(){
   BX24.installFinish();
});
```

Автоматизация

Функция	Описание
crm.automation.trigger	Активирует триггер Webhook, настроенный в автоматизации CRM.

© «Битрикс», 2001-2008, «1С-

Триггеры приложений

В автоматизации CRM есть возможность регистрировать триггеры приложений.

Метод	Описание	С версии
crm.automation.trigger.add	Метод добавляет триггер.	
crm.automation.trigger.delete	Метод удаляет триггер.	
crm.automation.trigger.list	Метод для получения списка приложений и триггеров	
crm.automation.trigger.execute	Метод, запускающий выполнение триггера	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Внешние каналы

Функция	Описание
crm.externalchannel.activity.company	Создает дело "Документ от компании".
crm.externalchannel.activity.contact	Создает дело "Документ от контакта".
crm.externalchannel.company	Импортирует Компанию и создает дело "Импорт компании".
crm.externalchannel.contact	Импортирует Контакт и создает дело "Импорт контакта".
crm.externalchannel.connector.register	Регистрирует коннектор внешнего канала.
crm.externalchannel.connector.unregister	Удаляет коннектор внешнего канала.

Параметры функций

Параметр	Описание

"обновляемое поле"=>"значение"	Набор полей для компании|контакта - массив вида array("обновляемое поле"=>"значение"[, ...]), где "обновляемое поле" может принимать значения из возвращаемых методом crm.company.fields|crm.contact.fields для агентов и crm.activity.fields для дела. Для дела, поле IS_MY_COMPANY - Y (компания продавец) | N.
MESSAGE	Текст сообщения.
ORIGIN_ID	Внешний код агента.
COMPANY_ORIGIN_ID	Актуально только для контакта. Обрабатывается, но использовать не рекомендуется т.к. логика работы с этим полем может быть изменена без предупреждения. При передаче данного параметра осуществляется поиск компании по origin_id, в случае если она найдена контакт привязывается к компании.
RQ_ADDR	Множественный список с предустановленными типами: ▪ PRIMARY - Фактический адрес, ▪ HOME - Адрес регистрации, ▪ REGISTERED - Юридический адрес, ▪ BENEFICIARY - Адрес бенефициара
RESULT_CURRENCY_ID	Коды валюты, берутся из справочника crm.currency.list.
RESULT_VALUE	Актуально только для дела подтипа SELLING: 1 - проведен, 0 - не проведен.
START_TIME	Дата дела в формате ISO-8601.

VERSION	Версия агента. Смена версии агента приводит к обновлению контакта в системе и созданию дела Импорт #агент#.
CHANNEL_ID	ID канала, где значение принимает значение из возвращаемого методом crm.externalchanell.connector.register.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Каталог

Функция	Описание
crm.catalog.fields	Возвращает описание полей каталога товаров.
crm.catalog.get	Возвращает товарный каталог по идентификатору.
crm.catalog.list	Возвращает список товарных каталогов.

© «Битрикс», 2001-2008, «1С-

Валюты

Функция	Описание
crm.currency.add	Создаёт новую валюту.
crm.currency.delete	Удаляет валюту.
crm.currency.fields	Возвращает описание полей валюты.
crm.currency.get	Возвращает валюту по символьному идентификатору.
crm.currency.list	Возвращает список валют.
crm.currency.localizations.delete	Удаляет выбранные локализации для валюты, указанной по символьному идентификатору.
crm.currency.localizations.get	Возвращает локализации для валюты, указанной по символьному идентификатору.
crm.currency.localizations.set	Устанавливает локализации для валюты, указанной по символьному идентификатору.
crm.currency.localizations.fields	Возвращает описание локализаций для валюты.
crm.currency.update	Обновляет существующую

	валюту.
crm.currency.base.get	Метод позволяет получить символьный идентификатор базовой валюты.
crm.currency.base.set	Метод устанавливает валюту в качестве базовой.

Общий список **событий валют** приведен здесь.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

Событие	Вызывается
onCrmCurrencyAdd	после создании валюты
onCrmCurrencyUpdate	после обновлении валюты
onCrmCurrencyDelete	после удалении валюты

© «Битрикс», 2001-2008, «1С-

Единицы измерения

Функция	Описание
crm.measure.add	Добавляет новую единицу измерения.
crm.measure.delete	Удаляет единицу измерения.
crm.measure.fields	Возвращает описание полей для единиц измерений.
crm.measure.get	Возвращает единицу измерения по идентификатору.
crm.measure.list	Возвращает список единиц измерений.
crm.measure.update	Обновляет существующую единицу измерения.

© «Битрикс», 2001-2008, «1С-

Разделы товаров

Функция	Описание
crm.productsection.add	Создаёт новый раздел товаров.
crm.productsection.delete	Удаляет раздел товаров.
crm.productsection.fields	Возвращает описание полей раздела товара.
crm.productsection.get	Возвращает раздел товаров по идентификатору.
crm.productsection.list	Возвращает список разделов товаров по фильтру.
crm.productsection.update	Обновляет существующий раздел товаров.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Товарные позиции (старые)

Функция	Описание
crm.productrow.fields	Возвращает описание полей товарных позиций.
crm.productrow.list	Возвращает список товарных позиций по фильтру.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Товары

Функция	Описание
crm.product.add	Создаёт новый товар.
crm.product.delete	Удаляет товар.
crm.product.fields	Возвращает описание полей товара.
crm.product.get	Возвращает товар по идентификатору.
crm.product.list	Возвращает список товаров по фильтру.
crm.product.update	Обновляет существующий товар.
crm.product.property.types	Возвращает список типов свойств товаров.
crm.product.property.fields	Возвращает описание полей для свойств товаров.
crm.product.property.settings.fields	Возвращает описание полей дополнительных настроек свойства товаров пользовательского типа.

crm.product.property.enumeration.fields	Возвращает описание полей элемента свойства товаров списочного типа.
crm.product.property.add	Создаёт новое свойство товаров.
crm.product.property.get	Возвращает свойство товаров по идентификатору.
crm.product.property.list	Возвращает список свойств товаров.
crm.product.property.update	Обновляет существующее свойство товаров.
crm.product.property.delete	Удаляет свойство товаров.

Общий список **событий товаров** приведен здесь.

События

Событие	Вызывается
onCrmProductAdd	при создании товара
onCrmProductUpdate	при обновлении товара
onCrmProductDelete	при удалении товара
onCrmProductPropertyAdd	при создании свойства товара
onCrmProductPropertyUpdate	при обновлении свойства товара
onCrmProductPropertyDelete	при удалении свойства товара

© «Битрикс», 2001-2008, «1С-

Вспомогательные сущности

- Дубликаты
- Множественные поля
- Перечисления
- Справочники
- Ставки НДС

© «Битрикс», 2001-2008, «1С-

Дубликаты

Настройки поиска дубликатов по любым полям

Внимание! Методы этого раздела будут работать с версии CRM 22.200.0.

- crm.duplicate.findbycomm
- crm.entity.mergeBatch

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Настройки поиска дубликатов по любым полям

Внимание! Методы этого раздела будут работать с версии CRM 22.200.0.

- crm.duplicate.volatileType.fields
- crm.duplicate.volatileType.list
- crm.duplicate.volatileType.register
- crm.duplicate.volatileType.unregister

© «Битрикс», 2001-2008, «1С-

Множественные поля

- crm.multifield.fields

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Перечисления

Метод	Описание
crm.enum.fields	Возвращает описание полей перечисления
crm.enum.ownertype	Возвращает элементы перечисления "Тип владельца".
crm.enum.contenttype	Возвращает элементы перечисления "Тип содержания".
crm.enum.activitytype	Возвращает элементы перечисления "Тип активности".
crm.enum.activitypriority	Возвращает элементы перечисления "Приоритет активности".
crm.enum.activitydirection	Возвращает элементы перечисления "Направление активности" (для писем и звонков).
crm.enum.activitynotifytype	Возвращает элементы перечисления "Тип уведомления о начале активности" (для встреч и звонков).
crm.enum.addresstype	Возвращает элементы перечисления "Тип адреса".
crm.enum.activitystatus	Возвращает элементы перечисления "Статус" (STATUS).

crm.enum.settings.mode	Возвращает описание режимов работы CRM.

Ставки НДС

Функция	Описание
crm.vat.fields	Возвращает описание полей ставки НДС.
crm.vat.list	Возвращает список ставок НДС.
crm.vat.get	Возвращает ставку НДС по идентификатору.
crm.vat.add	Создаёт новую ставку НДС.
crm.vat.update	Обновляет существующую ставку НДС.
crm.vat.delete	Удаляет ставку НДС.

© «Битрикс», 2001-2008, «1С-

Лента CRM

Функция	Описание
crm.livefeedmessage.add	Добавляет сообщение в ленту CRM.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Режим работы CRM

- crm.settings.mode.get

© «Битрикс», 2001-2008, «1С-

Бизнес-процессы

Пространство имён для rest методов работы с Бизнес-процессами.

Пространство	Описание
Бизнес-процесс	Методы работы с Бизнес-процессами
Действия приложений	Методы для работы с действиями
Задания	Методы для работы с заданиями
Провайдеры	Методы для работы с провайдерами.
Роботы приложений	Методы для работы с роботами приложений.
События	Методы для работы с событиями.
Шаблоны Бизнес-процессов	Методы для работы с шаблонами Бизнес-процессов.

При работе с Бизнес-процессами через API важно помнить:

- роботы запускаются всегда, если была смена стадии, либо добавлен элемент (на первой стадии);
- изменения документа из БП (действием или роботом) не запускают БП с автозапуском на изменение;
- создание сущностей действиями БП не запускают БП с автозапуском на создание;
- из реста БП запускаются только на платных тарифах, на demo лицензиях и лицензиях NFR.

Методы для встраивания приложений

Методы для встраивания внешних приложений в интерфейс *Битрикс24*.

Функция	Описание
placement.bind	Метод установит обработчик места встраивания. Работает только при авторизации под пользователем с правами администрирования портала.
placement.unbind	Метод удалит обработчик места встраивания. Работает только при авторизации под пользователем с правами администрирования портала.
placement.list	Получение списка доступных мест встраивания.
placement.get	Получение списка своих мест встраивания. Работает только при авторизации под пользователем с правами администрирования портала.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Встраивание приложений

Методы, обеспечивающие встраивание сторонних приложений в интерфейс *Битрикс24*.

Метод	Описание
BX24.placement.info	Получение информации о контексте вызова
BX24.placement.getInterface(Function callback)	Получение информации о js-интерфейсе текущего места встраивания: списке возможных команд и событий
BX24.placement.call(command, parameters, callback)	Вызов зарегистрированной команды интерфейса
BX24.placement.bindEvent(event, callback)	Установка обработчика события интерфейса

Дополнительно

Встраивание приложений в курсе Маркетплейс Битрикс24

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Задания

Методы для работы с заданиями

Метод	Описание
bizproc.task.complete	Метод осуществляет выполнение заданий БП.
bizproc.task.list	Метод возвращает список заданий бизнес-процессов

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Провайдеры

Методы для работы с провайдерами.

Внимание! Методы устарели, рекомендуется к использованию Служба сообщений

Метод	Описание	С версии
bizproc.provider.add	Регистрирует новый SMS-провайдер в бизнес-процессе.	17.0.3
bizproc.provider.delete	Удаляет зарегистрированного SMS-провайдера.	17.0.3
bizproc.provider.list	Возвращает список зарегистрированных приложением SMS-провайдеров.	17.0.3

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

Методы для работы с событиями.

Метод	Описание
bizproc.event.send	Метод возвращает действию выходные параметры, заданные в описании действия

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Работа с пользовательскими полями

Метод	Описание	С версии
userfieldconfig.add	Метод добавит новое пользовательское поле.	
userfieldconfig.delete	Метод удалит настройки поля с идентификатором id.	
userfieldconfig.get	Метод вернет данные о настройках пользовательского поля с идентификатором id.	
userfieldconfig.getTypes	Метод вернет набор доступных типов пользовательских полей для модуля moduleId.	
userfieldconfig.list	Метод вернет список настроек пользовательских полей.	
userfieldconfig.update	Метод изменяет значение поля.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Документы

Методы	Описание
documentgenerator.document.get	Возвращает информацию о документе по его идентификатору.
documentgenerator.document.getfields	Возвращает список полей документа с их описанием.
documentgenerator.document.list	Возвращает список документов по фильтру.
documentgenerator.document.delete	Удаляет документ.
documentgenerator.document.add	Создает новый документ на основании шаблона.
documentgenerator.document.update	Обновляет существующий документ с новыми значениями.
documentgenerator.document.enablepublicurl	Включает / выключает публичную

	ссылку на документ.

Шаблоны документов

Методы	Описание
documentgenerator.template.get	Возвращает информацию о шаблоне по его идентификатору.
documentgenerator.template.getfields	Возвращает список полей шаблона с их описанием.
documentgenerator.template.list	Возвращает список шаблонов по фильтру.
documentgenerator.template.delete	Удаляет шаблон.
documentgenerator.template.add	Добавляет новый шаблон.
documentgenerator.template.update	Обновляет существующий шаблон.

© «Битрикс», 2001-2008, «1С-

Нумераторы

Структура настроек нумератора перечислена здесь.

Методы	Описание
documentgenerator.numerator.get	Возвращает информацию о нумераторе по его идентификатору.
documentgenerator.numerator.list	Возвращает список нумераторов.
documentgenerator.numerator.add	Добавляет новый нумератор.
documentgenerator.numerator.update	Обновляет существующий нумератор с новыми значениями.
documentgenerator.numerator.delete	Удаляет нумератор.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Версии

Метод	Описание	С версии
disk.version.get	Возвращает версию по идентификатору	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Папка

Функция	Описание
disk.folder.getfields	Возвращает описание полей папки.
disk.folder.get	Возвращает папку по идентификатору.
disk.folder.getchildren	Возвращает список файлов и папок, которые находятся непосредственно в папке.
disk.folder.addsubfolder	Создает дочернюю папку.
disk.folder.copyto	Копирует папку в указанную папку.
disk.folder.moveto	Перемещает папку в указанную папку.
disk.folder.rename	Переименовывает папку.
disk.folder.deletetree	Уничтожает папку и всё её дочерние элементы навсегда.
disk.folder.markdeleted	Перемещает папку в корзину.
disk.folder.restore	Восстанавливает папку из корзины.
disk.folder.uploadfile	Загружает новый файл в указанную папку.
disk.folder.getExternalLink	Метод возвращает публичную ссылку.

Права доступа

Метод	Описание	С версии
disk.rights.getTasks	Метод позволяет получить список уровней доступов, которые можно использовать в назначении прав.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Прикреплённый файл

Метод	Описание	С версии
disk.attachedObject.get	Возвращает информацию о прикрепленном файле.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Файл

Функция	Описание
disk.file.getfields	Возвращает описание полей файла.
disk.file.get	Возвращает файл по идентификатору.
disk.file.rename	Переименовывает файл.
disk.file.copyto	Копирует файл в указанную папку.
disk.file.moveto	Перемещает файл в указанную папку.
disk.file.delete	Уничтожает файл навсегда.
disk.file.markdeleted	Перемещает файл в корзину.
disk.file.restore	Восстанавливает файл из корзины.
disk.file.uploadversion	Загружает новую версию файла.
disk.file.getVersions	Возвращает список версий файла.
disk.file.restoreFromVersion	Восстанавливает файл из конкретной версии.
disk.file.getExternalLink	Возвращает публичную ссылку на файл.

Хранилище

Функция	Описание
disk.storage.getfields	Возвращает описание полей хранилища.
disk.storage.get	Возвращает хранилище по идентификатору.
disk.storage.rename	Переименовывает хранилище. Допустимо переименование только хранилища приложения (см. disk.storage.getforapp).
disk.storage.getlist	Возвращает список доступных хранилищ.
disk.storage.gettypes	Возвращает список типов хранилищ.
disk.storage.addfolder	Создает папку в корне хранилища.
disk.storage.getchildren	Возвращает список файлов и папок, которые находятся непосредственно в корне хранилища.
disk.storage.uploadfile	Загружает новый файл в корне хранилища.
disk.storage.getforapp	Возвращает описание хранилища, с которым может работать приложение для хранения своих данных (файлов и папок).

© «Битрикс», 2001-2008, «1С-

Описание

Пространство имен, содержащее REST-методы модуля Задач.

Название	Описание
task.task.*	Методы работы с задачами.
task.item.*	Методы по работе с сущностью задачи.
task.commentitem.*	Методы по работе с комментариями в задачах.
task.dependence.*	Методы для работы с зависимостями задач.
task.stages.*	Методы для работы со стадиями Канбана.
task.stages.*	Методы для работы со стадиями Канбана.
task.checklistitem.*	Методы по работе с элементами чек-листа в задачах.
task.elapseditem.*	Методы по работе с учетом затраченного на задачи времени.
task.planner.*	Методы по работе со списком задач ("план на день") в окне инструмента учета рабочего времени ("timeman").
task.item.userfield.*	Методы для работы с пользовательскими полями.
tasks.api.scrum.*	Методы для работы со Скрамом.

commentitem

Предоставляет работу с комментариями в задачах.

Метод	Описание
task.commentitem.getmanifest	Возвращает список методов и их описание.
task.commentitem.getlist	Возвращает список комментариев к задаче.
task.commentitem.get	Возвращает комментарий к задаче.
task.commentitem.add	Создает новый комментарий к задаче.
task.commentitem.update	Обновляет данные комментария.
task.commentitem.delete	Удаляет комментарий.
task.commentitem.isactionallowed	Проверяет, разрешено ли действие.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Зависимости

Методы для работы с зависимостями задач.

Метод	Описание	С версии
task.dependence.add	Метод добавляет зависимость	
task.dependence.delete	Метод удаляет зависимость	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Kanbanitem

Методы для работы со стадиями Канбана.

Таблица стадий

Параметр	Описание
ID	Идентификатор стадии
TITLE	Заголовок
SORT	Индекс сортировки (только для чтения)
COLOR	Цвет в формате RGB, пример: FFAAEE.
SYSTEM_TYPE	Системный код, пока может быть или null или NEW - означающий, что данная стадия является стадией по-умолчанию (только для чтения)
ENTITY_TYPE	Тип сущности, к которой относится стадия (G - группа, U - пользователь) (только для чтения, а также указывается при добавлении)
ENTITY_ID	ID сущности, к которой относится стадия (группа или пользователь) (только для чтения)

Методы

Метод	Описание	С версии
task.stages.add	Метод добавляет стадии Канбана / Моего плана.	
task.stages.canmovetask	Метод определяет, может ли текущий пользователь перемещать задачи в указанной сущности	
task.stages.delete	Метод удаляет стадии Канбана / Моего плана.	
task.stages.get	Метод получает стадии Канбана / Моего плана.	
task.stages.movetask	Метод перемещает задачи из одной стадии в другую.	
task.stages.update	Метод обновляет стадии Канбана / Моего плана.	

Выборка задач

Выборка задач из стадий осуществляется методом task.task.get. Есть особенности.

1. Выборка из стадий Канбана группы. В массиве фильтра передаем ключ STAGE_ID с ID нужной вам стадии. Но если вы выбираете задачи из стадии по-умолчанию (она идет первой), вам требуется передать для STAGE_ID массив значений, состоящий из 0 и ID вашей стадии ([0,]). Дополнительно надо отфильтровать еще и по группе данную выборку.

2. Выборка из Моего плана. В массиве фильтра передаётся ключ STAGES_ID с ID нужной вам стадии. Передавать массив значений не требуется.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

checklistitem

Предоставляет работу с элементами чек-листа в задачах.

Метод	Описание
task.checklistitem.getmanifest	Возвращает список методов и их описание.
task.checklistitem.getlist	Возвращает список элементов чек-листа в задаче.
task.checklistitem.get	Возвращает элемент чек-листа по его идентификатору.
task.checklistitem.add	Добавляет новый элемент чек-листа к задаче.
task.checklistitem.update	Обновляет данные элемента чек-листа.
task.checklistitem.delete	Удаляет элемент чек-листа.
task.checklistitem.complete	Отмечает элемент чек-листа как выполненный.
task.checklistitem.renew	Отмечает выполненный элемент чек-листа как вновь активный.
task.checklistitem.moveafteritem	Помещает элемент чек-листа в списке после указанного.

task.checklistitem.isactionallowed	Проверяет, разрешено ли действие для элемента чек-листа.

elapseditem

Предоставляет работу с PHP-классом CTaskElapsedItem.

Метод	Описание
task.elapseditem.getmanifest	Возвращает список методов и их описание.
task.elapseditem.getlist	Возвращает список записей о затраченном времени по задаче.
task.elapseditem.get	Возвращает запись о затраченном времени по ее идентификатору.
task.elapseditem.add	Добавляет затраченное время к задаче.
task.elapseditem.delete	Удаляет запись о затраченном времени.
task.elapseditem.isactionallowed	Проверяет разрешено ли действие.
task.elapseditem.update	Изменяет параметры записи о затраченном времени.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

planner

Вместо **planner** можно также использовать его синоним **ctaskplannermaintance**.

Метод	Описание
task.planner.getlist	Возвращает массив, содержащий идентификаторы задач в плане на день.

© «Битрикс», 2001-2008, «1С-Битрикс»: 2008-2022

task.item.userfield

Методы для работы с пользовательскими полями.

Название	Описание
task.item.userfield.getfields	Получение всех доступных полей свойства.
task.item.userfield.gettypes	Получение всех доступных типов данных.
task.item.userfield.add	Создание нового свойства.
task.item.userfield.get	Получение свойства по идентификатору.
task.item.userfield.getlist	Получение списка свойств.
task.item.userfield.update	Редактирование параметров свойства.
task.item.userfield.delete	Удаление свойства.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Бэклог

Методы для работы с бэклогом:

Метод	Описание
tasks.api.scrum.backlog.add	Метод добавляет бэклог в Скрам.
tasks.api.scrum.backlog.delete	Метод удаляет бэклог.
tasks.api.scrum.backlog.get	Метод возвращает значения полей бэклога по id Скрама.
tasks.api.scrum.backlog.getFields	Метод возвращает доступные поля бэклога.
tasks.api.scrum.backlog.update	Метод изменяет бэклог.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Спринты

Методы для работы со спринтами:

Метод	Описание
tasks.api.scrum.sprint.add	Метод добавляет спринт в Скрам.
tasks.api.scrum.sprint.complete	Метод завершает активный спринт выбранного Скрама.
tasks.api.scrum.sprint.delete	Метод удаляет спринт.
tasks.api.scrum.sprint.get	Метод возвращает значения полей спринта по его id.
tasks.api.scrum.sprint.getFields	Метод возвращает доступные поля спринта.
tasks.api.scrum.sprint.list	Метод возвращает список спринтов.
tasks.api.scrum.sprint.start	Метод запускает спринт.
tasks.api.scrum.sprint.update	Метод изменяет спринт.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Эпики

Методы для работы с эпиками:

Метод	Описание
tasks.api.scrum.epic.add	Метод добавляет эпик в Скрам.
tasks.api.scrum.epic.delete	Метод удаляет эпик.
tasks.api.scrum.epic.get	Метод возвращает значения полей эпика по его id.
tasks.api.scrum.epic.getFields	Метод возвращает доступные поля эпика.
tasks.api.scrum.epic.list	Метод возвращает список эпиков.
tasks.api.scrum.epic.update	Метод изменяет эпик в Скраме.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Канбан

Методы позволяют создавать/изменять/удалять стадии канбана, а также добавлять задачи Скрама в канбан. Канбан Скрама обязательно должен содержать в себе стадии с типом `NEW` и `FINISH`.

Методы для работы с канбаном:

Метод	Описание
tasks.api.scrum.kanban.addStage	Метод создаёт стадию канбана Скрама.
tasks.api.scrum.kanban.addTask	Метод добавляет задачу в канбан Скрама.
tasks.api.scrum.kanban.deleteStage	Метод удаляет стадию.
tasks.api.scrum.kanban.deleteTask	Метод удаляет задачу из канбана Скрама.
tasks.api.scrum.kanban.getFields	Метод возвращает доступные поля стадии канбана.
tasks.api.scrum.kanban.getStages	Метод возвращает стадии канбана по id спринта.
tasks.api.scrum.kanban.updateStage	Метод изменяет стадию канбана Скрама.

Устаревшее

ВНИМАНИЕ! В данном разделе содержится информация об устаревших методах. Их использование в работе крайне не рекомендуется.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

items

Вместо **items** можно также использовать его синоним **ctasks**.

Метод	Описание
task.items.getlist	Возвращает массив задач, каждая из которых содержит массив полей.

Примечание: Вместо данного метода следует использовать метод task.item.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

comment

Используется для работы с комментариями в задачах.

Метод	Описание
task.comment.add	Добавление комментария к задаче.

Примечание: Вместо данного метода следует использовать метод task.commentitem.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

Событие	Вызывается
OnTaskAdd	при добавлении задачи.
OnTaskUpdate	при обновлении задачи.
OnTaskDelete	при удалении задачи.
OnTaskCommentAdd	при добавлении комментария к задаче.
OnTaskCommentUpdate	при проведении операций над комментарием к задаче.

Ссылки по теме:

- Общее описание REST - События

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Службы доставки

- Обработчики служб доставки
- Службы доставки
- Дополнительные услуги
- Транспортные заявки
- События

© «Битрикс», 2001-2008, «1С-

Обработчики служб доставки

Методы работы с рест-обработчиками служб доставок.

Метод	Описание	С версии
sale.delivery.handler.add	Метод добавляет обработчик служб доставки.	
sale.delivery.handler.delete	Метод удаляет обработчик служб доставки.	
sale.delivery.handler.list	Метод для получения списка обработчиков служб доставки.	
sale.delivery.handler.update	Метод выполняет обновление обработчика служб доставки.	

О работе с REST служб доставки читайте в соответствующей главе.

Службы доставки

Методы работы со службами доставок.

Метод	Описание	С версии
sale.delivery.add	Метод добавляет службу доставки.	
sale.delivery.config.get	Метод позволяет получить информацию о службе доставки по её ID.	
sale.delivery.config.update	Метод для обновления службы доставки.	
sale.delivery.delete	Метод удаляет службу доставки.	
sale.delivery.getList	Метод для получения списка доступных служб доставки.	
sale.delivery.update	Метод для редактирования службы доставки.	

О работе с REST служб доставки читайте в соответствующей главе.

Дополнительные услуги

Методы работы с дополнительными услугами служб доставок.

Метод	Описание	С версии
sale.delivery.extra.service.add	Метод для добавления дополнительной услуги службы доставки.	
sale.delivery.extra.service.delete	Метод удаляет дополнительную услугу службы доставки.	
sale.delivery.extra.service.get	Метод позволяет получить дополнительную услугу службы доставки по ID этой услуги.	
sale.delivery.extra.service.update	Метод для обновления дополнительной услуги службы доставки.	

Примеры настройки дополнительных услуг службы доставки смотрите в уроке Процесс создания и настройки службы доставки .

Транспортные заявки

Методы работы с транспортными заявками.

Метод	Описание	С версии
sale.delivery.request.delete	Метод удаляет транспортную заявку.	
sale.delivery.request.sendmessage	Метод для отправки сообщений о доставке.	
sale.delivery.request.update	Метод для обновления транспортной заявки.	

О работе с транспортными заявками читайте в уроке Процесс использования службы доставки в сценариях центра продаж.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

- Расчет стоимости доставки
- Создание заказа на доставку
- Отмена заказа на доставку

© «Битрикс», 2001-2008, «1С-

Частые кейсы

- Пример коннектора Открытых линий для онлайн-чата на сайте

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Методы

Пространство имен, содержащее REST-методы **Коннектора соединений**.

Метод	Описание	С версии
imconnector.register	Метод регистрирует новый тип коннектора.	
imconnector.unregister	Метод удаляет коннектор.	
imconnector.send.messages	Метод отправки сообщений в Открытые Линии (ОЛ).	
imconnector.update.messages	Метод обновления сообщений в ОЛ.	
imconnector.delete.messages	Метод удаления сообщений в ОЛ.	
imconnector.send.status.delivery	Метод подтверждения доставки сообщения от ОЛ во внешнюю систему.	

imconnector.send.status.reading	Метод подтверждения прочтения сообщения от ОЛ во внешнюю систему.	
imconnector.set.error	Метод отключения настроенного коннектора в ОЛ из внешней системы с пометкой статуса как ошибка.	
imopenlines.crm.chat.user.add	Метод добавляет пользователя в чат, полученый по CRM-сущности.	
imconnector.connector.data.set	Метод устанавливает данные для rest-коннектора.	
imopenlines.crm.chat.getLastId	Метод получает ID последнего чата, который привязан к CRM сущности.	

События

Событие	Описание
OnImConnectorLineDelete	Событие удаления открытой линии.
OnImConnectorMessageAdd	Событие нового сообщения из ОЛ.
OnImConnectorMessageUpdate	Событие изменения сообщения из ОЛ.
OnImConnectorMessageDelete	Событие удаления сообщения из ОЛ.

Ссылки по теме:

- Общее описание REST - События

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Настройка канала

Тип коннектора не может работать сам по себе. Он работает в рамках настроенной открытой линии. Со своей стороны необходимо хранить информацию об ID настроенной открытой линии.

Необходимо зарегистрировать PLACEMENT: ‘SETTING_CONNECTOR’

Передаваемые параметры

Параметр	Описание
CONNECTOR	ID коннектора.
LINE	ID линии.
STATUS	Статус подключения.
ACTIVE_STATUS	Статус активации.
CONNECTION_STATUS	Статус подключения.
REGISTER_STATUS	Статус регистрации.
ERROR_STATUS	Статус ошибки.

Методы

Метод	Описание	С версии
imconnector.activate	Метод устанавливает, активировать или деактивировать канал конкретной ОЛ.	
imconnector.status	Метод возвращает текущее состояние коннектора.	

© «Битрикс», 2001-2008, «1С-

Роботизация бизнеса

REST-методы, доступные при работе с роботами. Разрешение **rpa**.

Дополнительно

- Настройки пользовательских полей

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Методы для работы с сущностью Страница

Метод	Описание	С версии
landing.landing.add	Метод для добавления страницы.	
landing.landing.addByTemplate	Метод для добавления Страницы по шаблону.	
landing.landing.copy	Метод копирует указанную страницу.	
landing.landing.delete	Метод для удаления страницы.	
landing.landing.getadditionalfields	Метод для получения дополнительных полей страницы	
landing.landing.getlist	Метод для получения списка страниц.	
landing.landing.getpreview	Метод возвращает	

	путь до превью страницы.	
landing.landing.getpublicurl	Метод возвращает веб-адрес страницы.	
landing.landing.markDelete	Метод помечает страницу как удаленную.	
landing.landing.markUnDelete	Метод помечает страницу как не удаленную.	
landing.landing.move	Метод перемещает страницу в другой сайт и/или папку.	21.800.0
landing.landing.publication	Метод для публикации страницы.	
landing.landing.removeEntities	Метод удаляет связанные сущности лендинга.	
landing.landing.resolveIdByPublicUrl	Метод по переданному относительному URL страницы возвращает идентификатор страницы.	21.800.0
landing.landing.unpublic	Метод для снятия с публикации страницы.	
landing.landing.update	Метод для изменения	

	страницы.	

Методы для работы с Блоками на Странице

Метод	Описание	С версии
landing.landing.addblock	Метод для добавление нового блока на страницу.	
landing.landing.copyblock	Метод для копирования блока со страницы на страницу.	
landing.landing.deleteblock	Метод для удаление блока со страницы.	
landing.landing.downblock	Метод для опускания блока на одну позицию вниз на странице.	
landing.landing.favoriteBlock	Метод сохраняет имеющийся на странице блок в "Мои блоки".	21.800.0
landing.landing.hideblock	Метод скрывает блок со странице.	

landing.landing.markdeletedblock	Метод помечает блок как удаленный, но не удаляет его физически.	
landing.landing.markundeletedblock	Метод восстанавливает блока из помеченных как удаленный	
landing.landing.moveblock	Метод для переноса блока со страницы на страницу.	
landing.landing.showblock	Метод для показа блока со странице.	
landing.landing.unFavoriteBlock	Метод удаляет блок, который был сохранен в "Мои блоки".	21.800.0
landing.landing.upblock	Метод для поднятия блока на одну позицию вверх на странице.	

Методы для работы с сущностью Блоки

Методы работы с Блоками.

Метод	Описание	вер
landing.block.clonecard	Метод для клонирования карточки блока.	
landing.block.removecard	Метод для удаления блока	
landing.block.updatenodes	Метод для изменения контента блока.	
landing.block.changeNodeName	Метод изменяет название тега.	
landing.block.updateattrs	Метод для изменения атрибутов ноды блока.	
landing.block.updateStyles	Метод для изменения стилей блока.	
landing.block.getcontent	Метод для получения контента блока.	
landing.block.getlist	Метод для получение списка блоков страницы.	

landing.block.getbyid	Метод для получения блока по его идентификатору.	
landing.block.getmanifest	Метод для получения манифеста конкретного блока, уже размещенного на странице.	
landing.block.getmanifestfile	Метод для получения манифеста блока из репозитория.	
landing.block.getrepository	Метод возвращает список блоков из репозитория.	
landing.block.uploadfile	Метод загружает картинку и привязывает ее к указанному блоку.	
landing.block.updatecontent	Метод обновляет содержимое уже размещенного на странице блока на любой произвольный.	
landing.block.addcard	Метод полностью повторяет работу landing.block.clonecard но дает возможность вставить карточку сразу с измененным контентом.	
landing.block.updateCards	Метод для массового изменения карточек блока.	
landing.block.changeAnchor	Метод изменяет символьный код якоря.	

landing.block.getContentFromRepository	Метод получает контент блока из репозитория "как есть" до добавления блока на какую-либо страницу.	18.7

Сущность Сайт

Описание основных и дополнительных полей сущности Сайт и методов работы с ними.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Сущность Страница

Описание основных и дополнительных полей сущности Страница и методов работы с ними.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Сущность Блоки

Описание идеологии и методов работы с Блоками.

© «Битрикс», 2001-2008, «1С-

Интерактивные блоки

Возможность подключать описанные ниже расширения включена с версии модуля 18.7.0.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Сущность Шаблон представления

Сущность описывает шаблоны представления страницы (с шапкой, подвалом, сайдбаром).

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Методы для работы с сущностью Шаблон

Метод	Описание	С версии
landing.template.getlist	Метод для получения списка шаблонов.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Партнёрские блоки

Описание репозитория партнерских блоков и методов работы с ним.

Партнерский репозиторий работает по тем же принципам, что и штатные блоки. Партнерский репозиторий позволяет добавлять в локальный каталог (доступный только на текущем портале) ваши собственные блоки практически без ограничений.

Метод	Описание	С версии
landing.repo.getList	Метод для получения списка блоков текущего приложения.	
landing.repo.register	Метод добавления блока в репозиторий.	
landing.repo.unregister	Метод удаления блока.	
landing.repo.checkContent	Метод проверяет контент на опасные подстроки.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Права

Доступны с версии 18.7.0.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Расширенная модель

Метод	Описание	С версии
landing.site.getRights	Метод вернет права текущего пользователя.	
landing.site.setRights	Устанавливает права доступа для сайта.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Ролевая модель

Метод	Описание	С версии
landing.role.getList	Метод позволяет получить список ролей.	
landing.role.getRights	Метод позволяет получить список сайтов, права на которые установлены в рамках роли.	
landing.role.setAccessCodes	Метод устанавливает для роли коды доступа, для которых будет действовать данная роль.	
landing.role.setRights	Метод устанавливает необходимые права в рамках роли для списков сайта.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

telephony

Описанные ниже методы необходимы для построения бесшовных интеграций с внешними системами. В такой интеграции сам звонок не поступает в Битрикс24, но позволяет показывать и скрывать карточку звонка, фиксировать лид и запись разговора.

Разрешение приложения: **Телефония** (telephony)

Методы

Метод	Описание
telephony.externalcall.register	Регистрирует звонок в Битрикс24
telephony.externalcall.finish	Завершает звонок
telephony.externalcall.show	Показывает карточку звонка пользователю
telephony.externalcall.hide	Скрывает карточку звонка у пользователя
telephony.externalCall.attachRecord	Метод прикрепляет запись к звонку и к делу звонка.
telephony.call.attachTranscription	Метод добавляет расшифровку записи к звонку.
telephony.externalLine.add	Метод добавляет

	внешнюю линию.
telephony.externalLine.update	Метод позволяет изменить название внешней линии.
telephony.externalLine.delete.	Метод для удаления внешней линии.
telephony.externalLine.get	Метод позволяет получить список внешних линий приложения.
telephony.externalCall.searchCrmEntities	Метод позволяет получить одним запросом по номеру телефона информацию о клиенте из CRM.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

voximplant

Разрешение приложения: **Телефония** (telephony)

Внимание! При создании приложений учтите, что методы доступны пользователям в соответствии с заданными для пользователей правами. Уровень требуемых прав указан в описании метода.

Методы

Метод	Описание
voximplant.url.get	Возвращает набор ссылок для навигации по страницам телефонии.
voximplant.sip.get	Возвращает список всех sip-линии (созданных приложением).
voximplant.sip.add	Создает новую sip-линию с привязкой к приложению. После создания данная линия становится исходящей линией по-умолчанию.
voximplant.sip.update	Обновляет существующую sip-линию (созданную приложением).
voximplant.sip.delete	Удаляет существующую sip-

	линию (созданную приложением).
voximplant.sip.status	Возвращает текущий статус sip-регистрации (только для облачных АТС).
voximplant.sip.connector.status	Возвращает текущий статус SIP-Коннектора.
voximplant.line.outgoing.sip.set	Установка выбранной sip-линии в качестве исходящей линии по-умолчанию.
voximplant.line.get	Возвращает список всех доступных исходящих линий.
voximplant.line.outgoing.set	Установка выбранной линии в качестве исходящей линии по-умолчанию.
voximplant.line.outgoing.get	Возвращает текущую выбранную линии в качестве исходящей линии по-умолчанию.
voximplant.callback.start	Метод запускает обратный звонок.
voximplant.infocall.startwithsound	Осуществляет звонок на указанный номер с проигрыванием файла формата mp3 по URL
voximplant.infocall.startwithtext	Осуществляет звонок на указанный номер с автоматическим произнесением заданного текста.
voximplant.tts.voices.get	Возвращает массив доступных голосов для

	синтеза речи.
voximplant.user.get	Возвращает настройки пользователей.
voximplant.user.deactivatePhone	Отключает сотруднику признак наличия sip-аппарата.
voximplant.user.activatePhone	Устанавливает сотруднику признак наличия sip-аппарата.

Статистика звонков

В статистику звонков попадают как звонки, сделанные через встроенную телефонию VoxImplant, так и через звонки, которые были зарегистрированы внешней телефонией при помощи метода telephony.externalcall.register.

Метод	Описание	С версии
voximplant.statistic.get	Возвращает список истории звонков.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

События

Событие	Вызывается
OnVoximplantCallInit	при инициализации звонка (о поступлении или начале исходящего звонка).
OnVoximplantCallStart	при начале разговора (ответе оператора при входящем и ответе абонента при исходящем).
OnVoximplantCallEnd	при окончании разговора (запись в историю).

Ссылки по теме:

- Общее описание REST - События

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Методы

Метод	Описание
CallCardSetMute	Позволяет со стороны приложения выключить микрофон оператора.
CallCardSetUiState	Позволяет со стороны приложения изменить состояние интерфейса карточки звонка.
CallCardClose	Позволяет со стороны приложения закрыть карточку звонка.
CallCardSetStatusText	Позволяет со стороны приложения изменить текст в центре карточки звонка.
CallCardSetCardTitle	Позволяет изменить со стороны приложения титульник карточки звонка.
CallCardGetListUiStates	Позволяет получить список доступных состояний интерфейса карточки звонка.
CallCardSetHold	Позволяет со стороны приложения поставить звонок на удержание.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Перечисления

Методы перечисления.

Метод	Описание
catalog.enum.getRoundTypes	Метод перечисления типов округления.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Наценка

Методы работы с наценками:

Метод	Описание
catalog.extra.get	Метод для получения значений полей наценки по ID.
catalog.extra.getFields	Метод возвращает поля наценки.
catalog.extra.list	Метод получает список наценок по фильтру.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Единица измерения

Методы работы с единицами измерений:

Метод	Описание
catalog.measure.get	Метод для получения значений полей единиц измерения по ID.
catalog.measure.getFields	Метод возвращает поля единиц измерения.
catalog.measure.list	Метод получает список единиц измерения по фильтру.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Цена

Методы работы с ценами:

Метод	Описание
catalog.price.delete	Метод для удаления цены товара.
catalog.price.get	Метод для получения значений полей цены товара по ID.
catalog.price.getFields	Метод возвращает поля цен товаров.
catalog.price.list	Метод для получения списка цен товаров по фильтру.
catalog.price.modify	Метод для изменения элементов коллекции цен товара.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Тип цены

Методы работы с типами цен:

Метод	Описание
catalog.priceType.get	Метод для получения значений полей типа цены по ID.
catalog.priceType.getFields	Метод возвращает поля типа цен.
catalog.priceType.list	Метод получает список типов цен по фильтру.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Товар

Методы работы с товаром торгового каталога:

Метод	Описание
catalog.product.add	Метод добавляет товар торгового каталога.
catalog.product.delete	Метод удаляет товар торгового каталога.
catalog.product.download	Метод скачивания файлов товара торгового каталога по переданным параметрам.
catalog.product.get	Метод для получения значений полей товара торгового каталога по ID.
catalog.product.getFieldsByFilter	Метод возвращает поля товара.
catalog.product.list	Метод получает список товаров торгового каталога по фильтру.
catalog.product.update	Метод для обновления полей товара торгового каталога.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Коэффициент единицы измерения

Методы работы с коэффициентами единиц измерения товаров:

Метод	Описание
catalog.ratio.get	Метод для получения значений полей коэффициента единицы измерения по ID.
catalog.ratio.getFields	Метод возвращает поля коэффициентов единиц измерения товаров.
catalog.ratio.list	Метод получает список коэффициентов единиц измерения по фильтру.

© «Битрикс», 2001-2008, «1С-Битрикс»: 2008-2022

Правила округления цен

Методы работы с правилами округления:

Метод	Описание
catalog.roundingRule.get	Метод для получения значений полей правил округления по ID.
catalog.roundingRule.getFields	Метод возвращает поля правил округления цен.
catalog.roundingRule.list	Метод получает список правил округления цен по фильтру.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Секция каталога

Методы работы с разделами каталога:

Метод	Описание
catalog.section.add	Добавляет секцию торгового каталога.
catalog.section.delete	Удаляет секцию торгового каталога.
catalog.section.get	Метод для получения значений полей секции торгового каталога по ID.
catalog.section.getFields	Метод возвращает поля секции торгового каталога.
catalog.section.list	Метод получает список секций торгового каталога по фильтру.
catalog.section.update	Метод для обновления полей секции торгового каталога

© «Битрикс», 2001-2008, «1С-

Налоги

Методы работы со ставками НДС:

Метод	Описание
catalog.vat.get	Метод для доступа к значениям полей налога по ID.
catalog.vat.getFields	Метод возвращает поля налога.
catalog.vat.list	Метод возвращает список налогов по фильру.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Универсальные списки

Rest-методы для работы с универсальными списками. Требуется разрешение **lists**.

Внимание! Если список заполняется через API, то нужно учитывать свойство **Привязка к сущности CRM**, а также разрешенные для нее привязки. Если в настройках поля разрешена привязка только к одному типу сущностей, то надо чтобы значение поля заполнялось идентификатором этой сущности без префикса.

Если указывать с префиксом, то визуально отображаться будет, а вот выгружаться в Excel нет.

© «Битрикс», 2001-2008, «1С-

Работа с элементами списка

Rest-методы для работы с элементами списка

Метод	Описание	С версии
lists.element.add	Метод создаёт элемент списка.	
lists.element.delete	Метод удаляет элемент списка.	
lists.element.get	Метод возвращает список элементов или элемент.	
lists.element.update	Метод обновляет элемент списка.	
lists.element.get.file.url	Метод возвращает путь к файлу.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Работа с полями списка

Rest-методы для работы с полями списка

Метод	Описание	С версии
lists.field.add	Метод создает поле списка.	
lists.field.delete	Метод удаляет поле списка.	
lists.field.get	Метод возвращает данные поля.	
lists.field.type.get	Метод возвращает доступные типа полей для указанного списка.	
lists.field.update	Метод обновляет поле списка.	

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Базовые методы

Метод	Описание
timeman.settings	Получение настроек рабочего времени пользователя
timeman.status	Получение информации о текущем рабочем дне пользователя
timeman.open	Начать новый рабочий день либо возобновить закрытый или приостановленный
timeman.close	Закрыть рабочий день
timeman.pause	Приостановить рабочий день

По умолчанию методы работают с рабочим днем владельца авторизационного токена или вебхука. Если владелец обладает правами на запись чужих рабочих дней, то он работает с рабочими днями любого пользователя.

Аналогично остальным методам REST API все параметры времени принимаются в формате ISO-8601 (ATOM). Временная зона, указываемая в передаваемых данных, учитывается и считается временной зоной пользователя. Дата открытия рабочего дня должна соответствовать текущему дню во временной зоне пользователя, а дата закрытия должна совпадать с датой открытия.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Рабочий график

Метод	Описание
timeman.schedule.get	Метод позволяет получить рабочий график по его идентификатору.

© «Битрикс», 2001-2008, «1С-Битрикс», 2008-2022

Задачи скрама

Под задачей Скрама понимается сущность Скрама, которая связана с сущностью задачи.

Методы для работы с задачами Скрама:

Метод	Описание
tasks.api.scrum.task.get	Метод возвращает значения полей задачи Скрама по её id.
tasks.api.scrum.task.getFields	Метод возвращает доступные поля задачи Скрама.
tasks.api.scrum.task.update	Метод создает или изменяет задачу Скрама.

© «Битрикс», 2001-2008, «1С-Битрикс»

Карточка звонка

Плейсмент **CALL_CARD** предназначен для работы с [dw]карточкой звонка[/dw][di][/di] в CRM. Интерфейс возвращается по вызову **BX24.placement.getInterface**.

Функции

Функция	Описание
`getStatus()`	Возвращает информацию о текущем звонке. Возвращает объект с полями: ▪ **CALL_ID** {string}: id текущего звонка ▪ **PHONE_NUMBER** {string}: номер телефона клиента ▪ **CRM_ENTITY_TYPE** {string}: тип связанной со звонком сущности crm (CONTACT|LEAD|COMPANY) ▪ **CRM_ENTITY_ID** {int}: id связанной со звонком сущности crm ▪ **CRM_ACTIVITY_ID** {int}: id дела crm, связанного со звонком ▪ **CALL_DIRECTION** {string}: направление звонка (incoming | outgoing | incomingTransfer | callback) ▪ **CALL_LIST_MODE** {bool}: признак работы в режиме обзвона

	▪ **CRM_BINDINGS** - массив привязок звонка к сущностям CRM.
`disableAutoClose()`	Отключает автоматическое закрытие карточки по завершению звонка

События

Метод	Описание
CallCard::EntityChanged	Возникает при смене текущего клиента в режиме обзвона. В обработчик события передается объект с полями: ▪ **PHONE_NUMBER** {string}: номер телефона клиента ▪ **CRM_ENTITY_TYPE** {string}: тип связанной со звонком сущности crm (CONTACT|LEAD|COMPANY) ▪ **CRM_ENTITY_ID** {int}: id связанной со звонком сущности crm
CallCard::BeforeClose	Возникает перед закрытием карточки звонка. В обработчик ничего не передается.
CallCard::CallStateChanged	Возникает при смене состояния текущего звонка. В обработчик передаются аргументы: ▪ **callState** {string}: текущее состояние звонка (idle | connecting | connected) ▪ **additionalParams** [object]: опционально, объект с

	полями: ▪ **failedCode** [string]: опционально, код завершения звонка (передается только при неуспешном завершении звонка, при переходе в состояние idle)

Пример

Вызов метода плейсмента (важно то, что результат приходит в колбэке):

```
BX24.placement.call('getStatus', {},
function (result) {
    console.log(result);
});

Подписка на событие плейсмента (тут все
обычно для подписки на события):
BX24.placement.bindEvent("CallCard::CallStat
eChanged", function (callState) {
   console.log(callState);
});
```