Бот платформа Битрикс24

Разработчику - I

Ольга Пичужкина

документация

Бот платформа Битрикс24

Разработчику - I

Ольга Пичужкина

документация

Bot platform Bitrix24

Attention! Please note that the online version of the course is more complete. It changes faster and contains information that is not included in the offline file: videos and pop-up windows with additional information. We recommend using the online version. File version dated 05/06/2022.

Where to begin

The chapter gives an idea of what chatbots are, what they are for, how they work, what they can do, as well as how to create your own chatbot application and, in fact, an example of a ready-made chatbot for the Bitrix24 platform .

Attention! There is a limit on creating chatbots within rest applications: no more than 5 chatbots per application.

What is a chatbot?

What are chatbots and what can they do?

A chatbot is a virtual interlocutor, a program that was created to simulate human behavior when communicating with one or more interlocutors.

What are chatbots, why are they needed and why develop them at all?

To a greater extent, this trend is being formed now abroad - there are a huge number of bots for ***Slack*** or ***Telegram that*** solve a variety of tasks - from searching for airline tickets to managing small teams of developers. And to get all this wealth, users do not even need to leave their preferred messenger.

What can chatbots do?

- **Routine replacement** - allows you to perform certain functions without involving people, and the work will be done instantly and flawlessly;
- **Search and aggregation** of news, analytics, data (Data-Driven Collaboration), data is available in the decision-making place - instant messengers and all participants who need it;

- **E-commerce** - for spontaneous purchases without a long search, mobile ecommerce + visual search + chatbots, for communication with customers;
- **First line** of work with clients, assistants, consultants, standard questions, telephony;
- **Just for Fun** - just for fun.

Bot platform Bitrix24

And in ***Bitrix24*** chats (both individual and group) are part of a much more complex ecosystem, one of the main channels of user communication, fully integrated with other business tools. And in this context, the use of chatbots opens up much more interesting prospects for business users, since ***Bitrix24*** (in the browser, desktop and mobile applications) is now the main workplace for a large number of companies.

It is very easy to write a chatbot that, for example, will report in chat to the right users with urgent information about the indicators of the internal accounting system integrated with ***Bitrix24.*** You can write a chatbot that will help couriers conveniently process orders based on ***Bitrix24 CRM*** deals on a mobile device right in the messenger - and you don't have to write a separate mobile application for them.

Chatbot development in ***Bitrix24*** is a very promising option fast and convenient automation of specific business processes. Convenient, because, as we have already found out, receiving information and managing through the messenger is what the mass user now prefers. And fast - because the development of a chat bot for ***Bitrix24*** is a fairly simple process.

Note: For a clearer understanding of what chat can do bots on the Bitrix24 platform, you can watch video examples of the capabilities of ready-made chat bots here.

Chatbot features

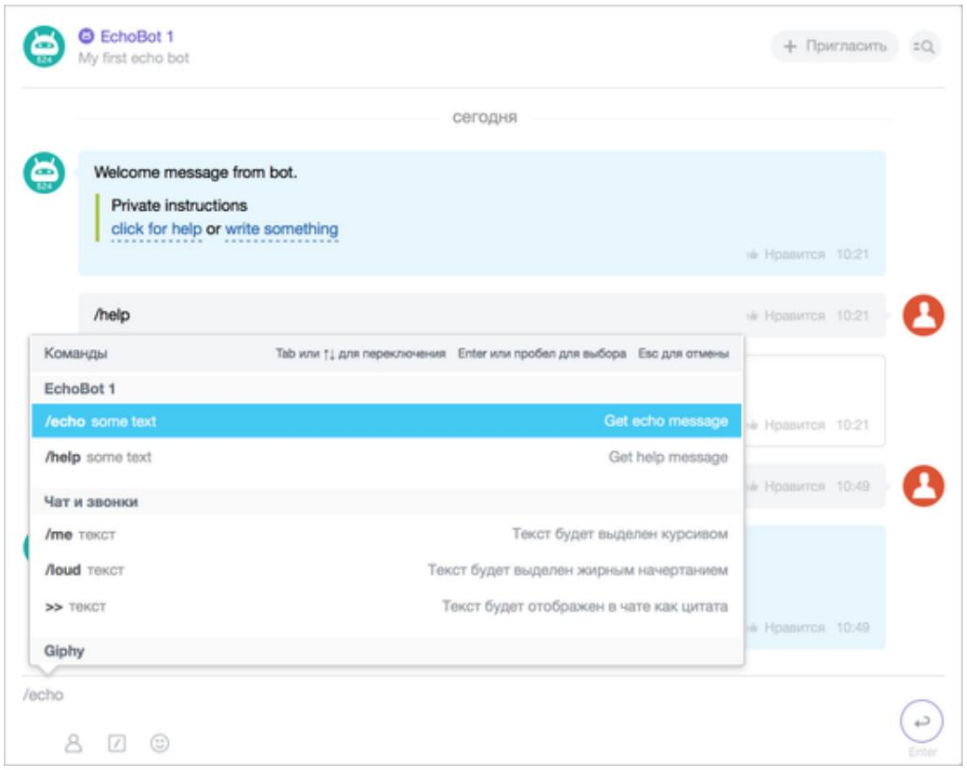
Chatbot:

- this is a special user in the system with whom you can communicate in a chat, but no one can log in under him;

- supports processing slash commands;
- allows you to use your keyboards for answers, turning a simple chat into a terminal.

Slash commands

Slash commands allow you to quickly create requests for output or receiving any information, format messages.



Note: You can read more about working with commands here .

Keyboards

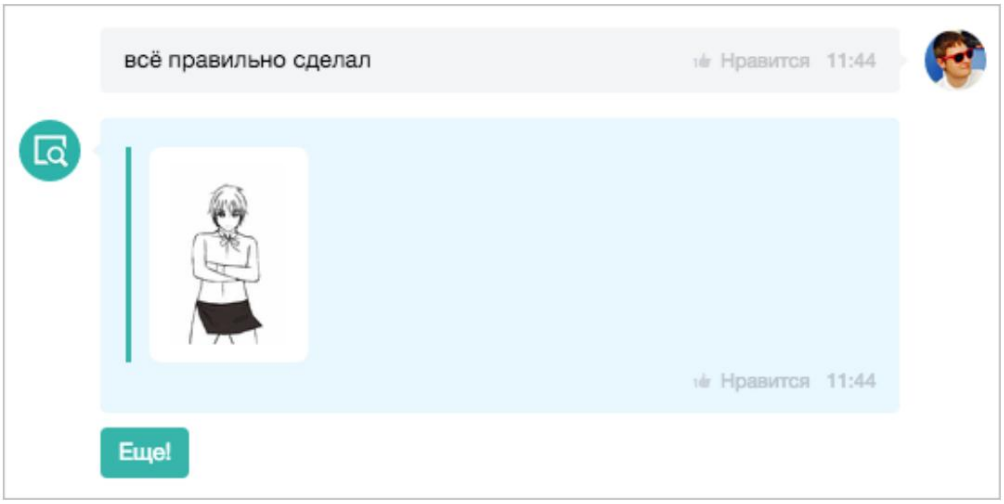
The possibilities of the keyboards are quite wide.

1. **EchoBot**

   Pagination, buttons when calling the "Help" command

2. **March**
   Just write March Play with me!. The keyboard is used as a playing field:

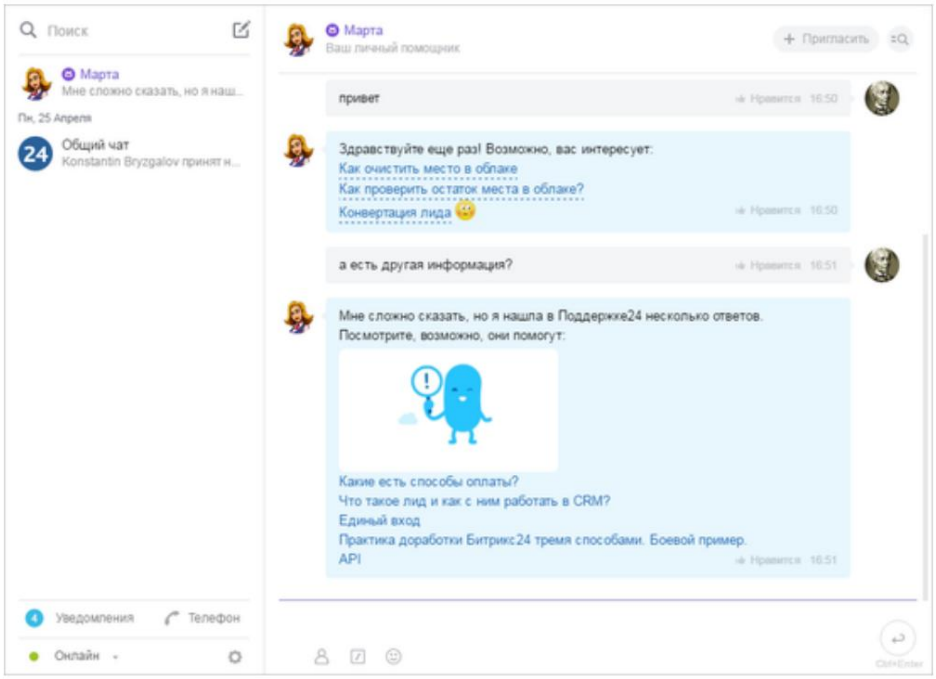
3. **Giphy**
   The **More** button allows you to view other pictures on the same topic without re-entering the search word:



Note: You can read more about working with keyboards here .

Chats

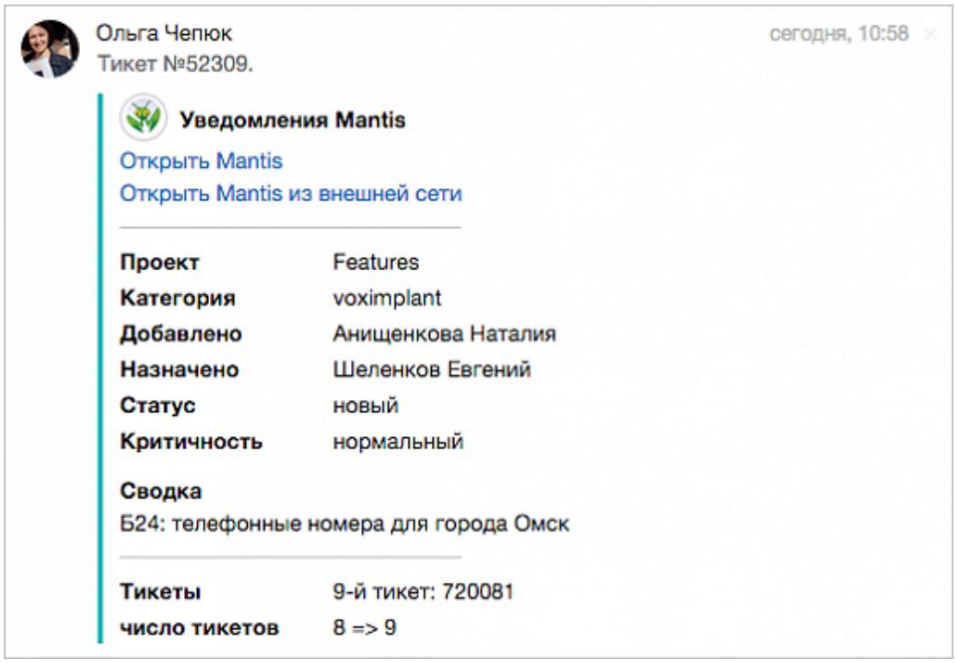
Chatbots can chat almost like real people. They can also remind you of various events (about current tasks, meetings) or provide background information. In addition to the fact that chatbots can write to chats, they can also create such chats and automatically invite people there, for example, to discuss a task.



Note: You can read more about working with chats here .

Notifications

Chatbot notifications can be useful and informative. They can consist of several blocks of various information from external system.

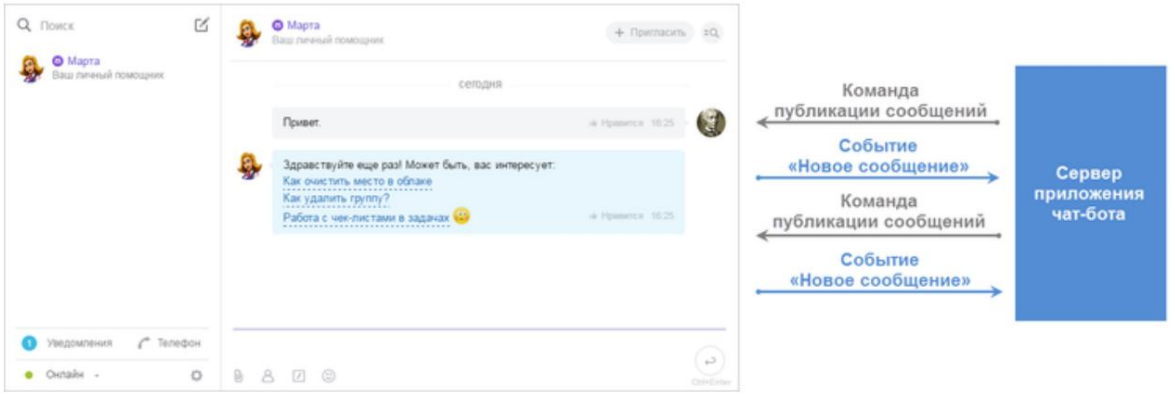


Note: You can read more about working with notifications here .

Chatbot lifecycle

The chatbot publishes its messages to the chat via Rest API, receives responses and user commands via Rest API Events (POST request).

The diagram of the life of a chatbot looks like this:



Creating your application

The main thing we need to understand about chatbots is that their logic usually based on response to some user actions and systems.

And we have 6 events that fully cover the required spectrum of reactions:

- ONAPPINSTALL - an event for installing an application with a chat bot.
- ONAPPUPDATE - application update event.
- ONIMJOINCHAT - event after the invitation of the chatbot "to the conversation", i.e. either when a user calls a chatbot in an individual chat, or when a chatbot connects to a group chat.
- ONIMBOTMESSAGEADD - event after sending a message from the user to the chatbot (in a group chat, with an explicit

bot mention).

- ONIMCOMMANDADD - event after sending a command from the user to the chatbot (in personal correspondence with the chatbot, or in a group chat (if the command is global, then the chatbot may not participate in the chat)).
- ONIMBOTDELETE - event after application deletion. The event is called in parallel with OnAppUninstall.

In other words, we must write handlers for the specified events, to implement simple logic:

1. Register a chatbot on the user portal when installation.
2. Display a greeting-help on behalf of the chatbot at the moment the chatbot is invited to the chat.
3. Learn to analyze the text of a message from the user and send something in response, and analysis means simple “command line parsing”, and not lexical parsing
   natural language.

And for this we have a set of simple methods added to REST An API specifically for chatbots, we only need two to get started:

- imbot.register - chatbot registration.
- imbot.message.add - send a message from the chatbot.

Obviously, in the ONAPPINSTALL event handler we will call the imbot.register method in order to add a chatbot to the current portal, then in the ONIMJOINCHAT event use the imbot.message.add method to display help about the functionality of the chatbot, and in the ONIMBOTMESSAGEADD handler we will respond to the user using the same imbot.message.add. Nothing complicated, agree?

Also, you don't have to implement full-fledged OAuth 2.0 in your application, since the parameters required for authorization come to the handlers in the **$_REQUEST array.**

Note: A complete list of Bot API methods and events can be found here.

An example of creating a chat bot

Chatbot code

As an example, let's create a chatbot that will inform the user about his overdue tasks. Our chatbot will know and process only one command **What is on.** By analogy, you can always expand its functionality to receive the necessary reports directly in the chat based on any data in ***Bitrix24.*** An example of a simple

chatbot:

```
<?php
/**
 * Useful chatbot with reports for bitrix24
 */

$appsConfig = array();
$configFileName = '/config_' . trim(str_replace('.', '_', $_REQUEST['auth']['domain'])) . '.php'; if
(file_exists(__DIR__ . $configFileName))
{ include_once __DIR__ . $configFileName;
}
// receive event "new message for bot"
if ($_REQUEST['event'] == 'ONIMBOTMESSAGEADD') {
// check the event - register this application or not
if (!isset($appsConfig[$_REQUEST['auth']['application_token']])) {
return false;
}
// response time
$arReport = getAnswer($_REQUEST['data']['PARAMS']['MESSAGE'], $_REQUEST['data']['PARAMS']
['FROM_USER_ID']);
$arReport['attach'][] = array("MESSAGE" => 'As you deal with these tasks, just
ask again [send=what's on]What's on?[/send] and I'll give you a new summary!');

// send answer message
$result = restCommand('imbot.message.add',
array(
"DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
"MESSAGE" => $arReport['title'] . "\n" . $arReport['report'] . "\n",
"ATTACH" => array_merge(
$arReport['attach']
```

```
),
),
$_REQUEST["auth"]);

} // receive event "open private dialog with bot" or "join bot to group chat"
else {
if ($_REQUEST['event'] == 'ONIMBOTJOINCHAT') {
// check the event - register this application or not
if (!isset($appsConfig[$_REQUEST['auth']['application_token']])) {
return false;
}
// send help message how to use chat-bot. For private chat and for group chat need send different instructions.

$result = restCommand('imbot.message.add', array(
 'DIALOG_ID' => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
      'MESSAGE' => 'Hello! I am Reporter, I report everything as it is.',
      "ATTACH" => array(
      array('MESSAGE' => '[send=what's on]What's on?[/send]'),
),
), $_REQUEST["auth"]);

} // receive event "delete chat-bot"
else {
if ($_REQUEST['event'] == 'ONIMBOTDELETE') {
// check the event - register this application or not
if (!isset($appsConfig[$_REQUEST['auth']['application_token']])) {
return false;
}
// unset application variables
unset($appsConfig[$_REQUEST['auth']['application_token']]);
// save params
saveParams($appsConfig);

} // receive event "Application install"
else {
if ($_REQUEST['event'] == 'ONAPPINSTALL') {
```

```
// handler for events
$handlerBackUrl = ($_SERVER['SERVER_PORT'] == 443 ? 'https' : 'http') . '://' .
$_SERVER['SERVER_NAME'] . (in_array($_SERVER['SERVER_PORT'],
array(80, 443)) ?  : ':' . $_SERVER['SERVER_PORT']) . $_SERVER['SCRIPT_NAME'];
// If your application supports different localizations
// use $_REQUEST['data']['LANGUAGE_ID'] to load correct localization
// register new bot
$result = restCommand('imbot.register', array(
    'CODE' => 'ReportBot',
// bot string identifier unique within your application (required)
 'TYPE' => 'B',
// Bot type, B - bot, answers come immediately, H - human, answers come with a delay of 2 to 10 seconds

 'EVENT_MESSAGE_ADD' => $handlerBackUrl,
// Link to the event handler for sending a message to the bot (required)
 'EVENT_WELCOME_MESSAGE' => $handlerBackUrl,
// Link to the event handler for opening a dialog with a bot or inviting him to
group chat (required)
 'EVENT_BOT_DELETE' => $handlerBackUrl,
// Reference to the event handler for the removal of the bot from the client side (required)
 'PROPERTIES' => array( // Personal data of the chatbot (required)
 'NAME' => 'Reporter',
// Bot name (mandatory one of the NAME or LAST_NAME fields)
 'LAST_NAME' => '',
// Last name of the bot (obligatory one of the NAME or LAST_NAME fields)
    'COLOR' => 'AQUA',
   // Bot color for mobile app
 RED, GREEN, MINT, LIGHT_BLUE, DARK_BLUE, PURPLE, AQUA, PINK, LIME, BROWN, AZURE, KHAKI,
SAND, MARENGO, GRAY, GRAPHITE
    'EMAIL' => 'no@mail.com',
// Contact email
 'PERSONAL_BIRTHDAY' => '2016-03-23',
// Birthday in YYYY-mm-dd format
 'WORK_POSITION' => 'Reporting on business',
// Position held, used as a description of the bot
 'PERSONAL_WWW' => '',
// Link
```

```
        'PERSONAL_GENDER' => 'M',
        // Gender of the bot, valid values M - male, F - female, empty if not required
indicate
            'PERSONAL_PHOTO' =>
        'iVBORw0KGgoAAAANSUhEUgAAADoAAAA6CAYAAADhu0ooAAAACXBIWXMAAAsTAAALEw
        EAmpwYAAAKT2lDQ1BQaG90b3Nob3AgSUNDIHByb2ZpbGUAAHjanVNnVFPpFj333vRCS4iAlE
        tvUhUIIFJCi4AUkSYqIQkQSoghodkVUcERRUUEG8igiAOOjoCMFVEsDIoK2AfkIaKOg6OIisr7',
        // Bot avatar - base64
    ),
    ), $_REQUEST["auth"]);
    // save params
    $appsConfig[$_REQUEST['auth']['application_token']] = array(
            'BOT_ID' => $result['result'],
        'LANGUAGE_ID' => $_REQUEST['data']['LANGUAGE_ID'],

    ); saveParams($appsConfig);
    // write debug log //
    writeToLog($result, 'ReportBot register');
    }
    }
    }
    }
    /**
     * 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['auth']['domain'])) . '.php';
    file_put_contents(__DIR__ . $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(), array $auth = array()) {
$queryUrl = 'https://' . $auth['domain'] . '/rest/' . $method; $queryData =
http_build_query(array_merge($params, array('auth' => $auth['access_token'])));
// writeToLog(array('URL' => $queryUrl, 'PARAMS' => array_merge($params, array("auth" => $auth["access_token"]))),
'ReportBot send data');
$curl = curl_init();
curl_setopt_array($curl, array(
CURLOPT_POST => 1,
CURLOPT_HEADER => 0,
CURLOPT_RETURNTRANSFER => 1,
CURLOPT_URL => $queryUrl,
CURLOPT_POSTFIELDS => $queryData,
));
$result = curl_exec($curl);
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(__DIR__ . '/imbot.log', $log, FILE_APPEND);
return true;
}
/**
 * We generate a report on the command
 *
 * @param string $text string sent by the user @param int $user id
 * of the user who wrote to us
 *
 * @return array
 */
function getAnswer($command = '', $user) {

switch (strtolower($command)) {
      case 'what's on':
$arResult = b24BadTasks($user);
break;
default:
$arResult = array(
      'title' => 'Tuplu-s',
      'report' => 'I don't understand what you want to know. Or maybe I can't at all...',
);
}

return $arResult;
}

function b24BadTasks ($user) {
$tasks = restCommand('task.item.list',
array(
```

```
'ORDER' => array('DEADLINE' => 'desc'),
'FILTER' => array('RESPONSIBLE_ID' => $user, '<DEADLINE' => '2016-03-23'),
'PARAMS' => array(),
'SELECT' => array()
),
$_REQUEST["auth"]);
if (count($tasks['result']) > 0) {
$arTasks = array();
foreach ($tasks['result'] as $id => $arTask) {
$arTasks[] = array(
'LINK' => array(
'NAME' => $arTask['TITLE'],
'LINK' => 'https://'.$_REQUEST['auth']
['domain'].'/company/personal/user/'.$arTask['RESPONSIBLE_ID'].'/tasks/task/view/'.$arTask['ID'].'/'
)
);
$arTasks[] = array(
'DELIMITER' => array(
'SIZE' => 400,
'COLOR' => '#c6c6c6'
)
);
}
    $arReport = array(
'title' => 'Yes, some tasks have already passed, for example:',
    'report' => '',
'attach' => $arTasks
);
}
else {
$arReport = array(
    'title' => 'Great work!',
    'report' => 'Nothing to upset - not a single overdue task',
);
}
return $arReport;
```

```
}
```

Now for you, this code looks large and incomprehensible, but after passing throughout the tutorial, you'll see that it's actually quite simple. Creating a

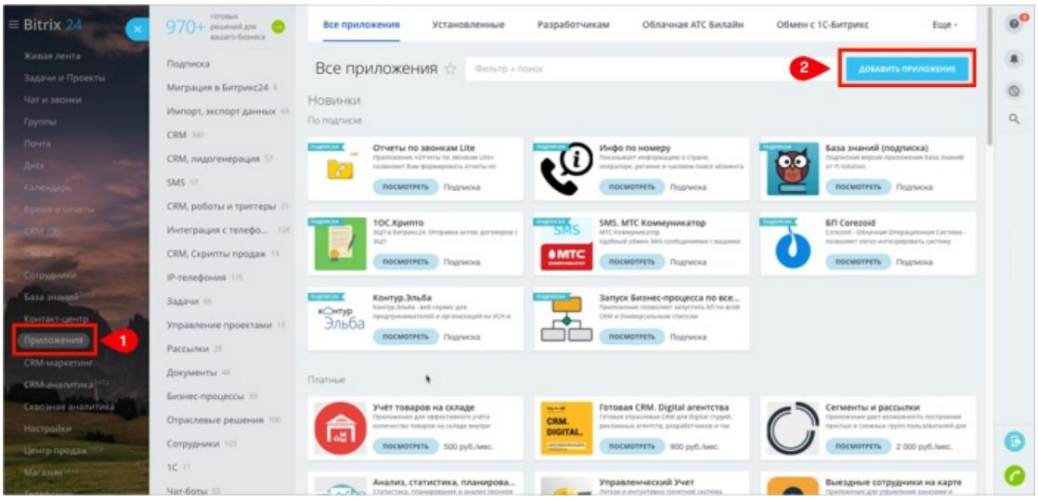
new chatbot application is available as a "personal use", and in the account for publication to all users ***of Bitrix24*** through ***the Market.***

Note: Chatbots are not required to use https certificate, but highly recommended due to the possible transfer of sensitive client data. The application itself must be encoded in UTF-8.

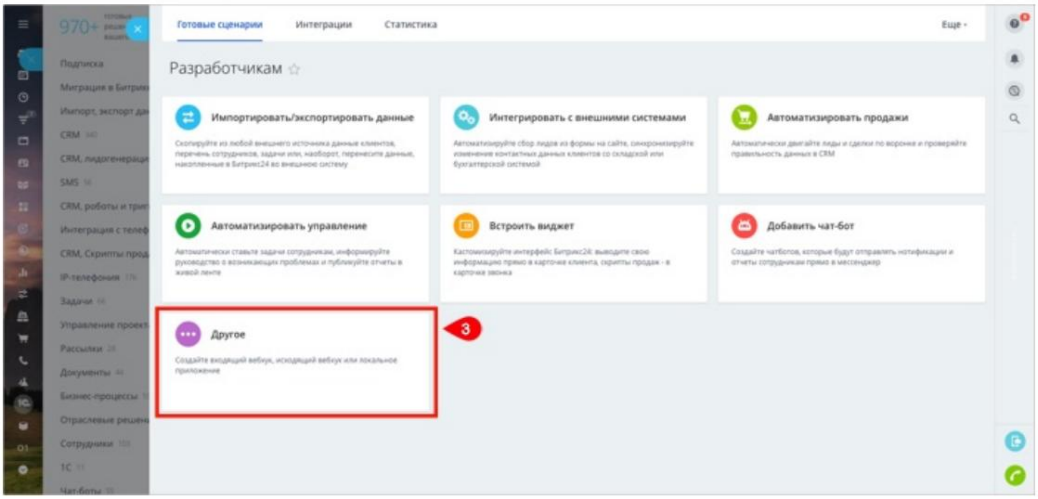
Launching a chatbot on your portal

You can already take the chatbot example code, put it on your server and launch a chatbot on your portal without publishing it through ***the Market.*** To do this, ***Bitrix24*** has the ability to install local applications:

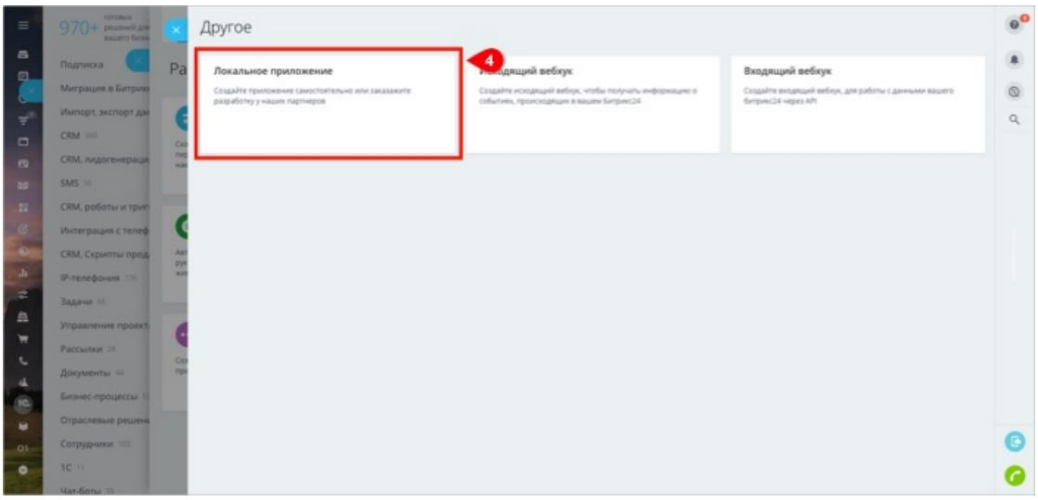
- In the left menu section **of Application** 1, click the **Add Application** 2 button:



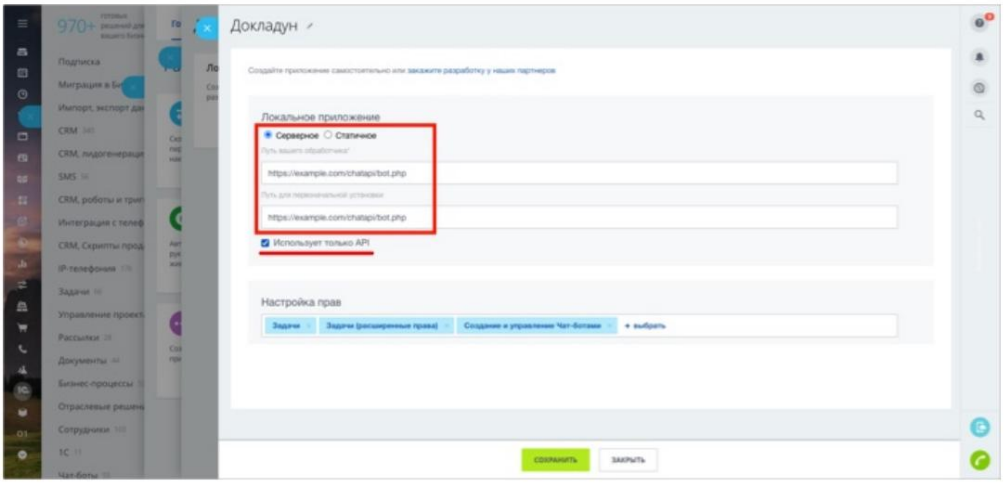
- Next, select **Other** 3:



- And then **the Local Application** 4 script:



- Select **the Server** application type. We indicate the name of the bot, and we will call it “Reporter”. Enable the **Use only API** option and give the application the following access rights: **Create and manage Chatbots** -- without these rights, the application will not be able to register the chatbot, as well as **Tasks** and **Tasks (advanced rights)** -- without these rights, the application will not be able to generate a report on tasks to report them to the user via the chatbot:



- Since our script is written in such a way that it is a handler for all events, in the application form we will specify both links to the same URL.
- If the bot is created for the boxed version of Bitrix24, that is, the **Use only API** option is not used , then you need to use the application type **Static** and the required field **Archive with your application (zip)** to specify the path to the installer file.

Actually, that's all - a message will appear in the general chat stating that a new user has joined the portal: a chatbot named "Reporter". We can open a chat with him, click on the link "What's on fire?" and get your list of overdue tasks.

Note: Another example is EchoBot, with a lot of features and constantly updated, can be downloaded from the GitHub service here.

An example of creating a chat bot for Open Lines

Special Requirements

Creating a chat bot for ***Open Lines*** is similar to creating regular chatbot, except for a few notes:

1. When creating a chat bot for *Open Lines* in imbot.register V the **TYPE** parameter must be passed O.

2. If you need to expand the capabilities of an existing chatbot, then you need to transfer a new key OPENLINE => Y, then the chatbot will work in hybrid mode.

Attention! *In hybrid mode, the chatbot must correctly handle the situation when it is invited and communicated in group, personal and open lines chat. To do this, in all incoming events (ONIMBOTMESSAGEADD, ONIMBOTJOINCHAT) check the* ***CHAT_ENTITY_TYPE parameter,*** *for Open lines it should be CHAT_ENTITY_TYPE => LINES.*

In all other respects, this is a familiar and already familiar

chatbot. For tighter integration with ***Open Lines,*** you must have access rights to scope **imopenlines.**

With these rights, the following commands will be available:

- imopenlines.network.join - connecting an open line of your company to the ***Bitrix24 portal,*** after which employees will be able to write.
- imopenlines.bot.session.operator - switching the conversation to a free operator.
- imopenlines.bot.session.transfer - switching the conversation to a specific operator.
- imopenlines.bot.session.finish - end of the current session.

Note: Chatbots are not required to use https certificate, but highly recommended due to the possible transfer of sensitive client data. The application itself must be encoded in UTF-8.

Download an example of a chat bot for Open Lines

As an example of a chat bot for open lines, we have prepared a chat ITR Bot . You can download it:

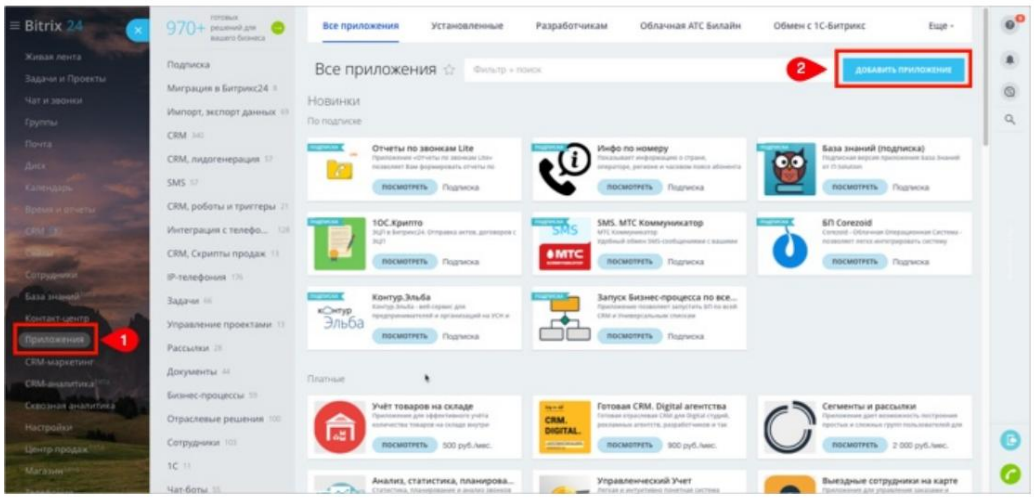
- from the GitHub service **(itr.php).**
- or take it from *Bitrix24 in a box* here: \Bitrix\ImBot\Bot\OpenlinesMenuExample.

This chatbot acts as the first line of support - first, all messages will be sent to it, and only then to employees in a queue after a period of time specified in the open line settings. It also added a class for building a multi-level menu in chats.

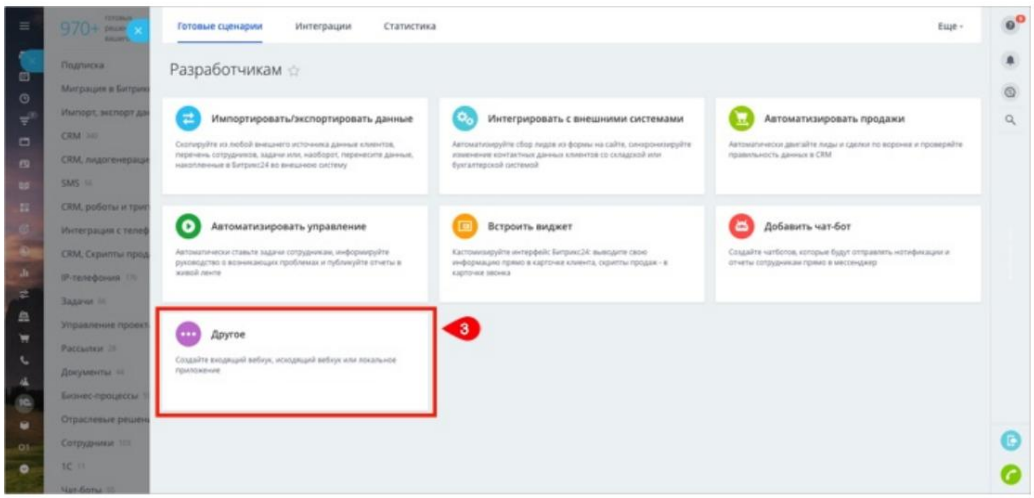
Launch on your portal

So you can take the chatbot example code right now above, put it on your server and run the chatbot on your portal as a local application without publishing it through ***the Market:***

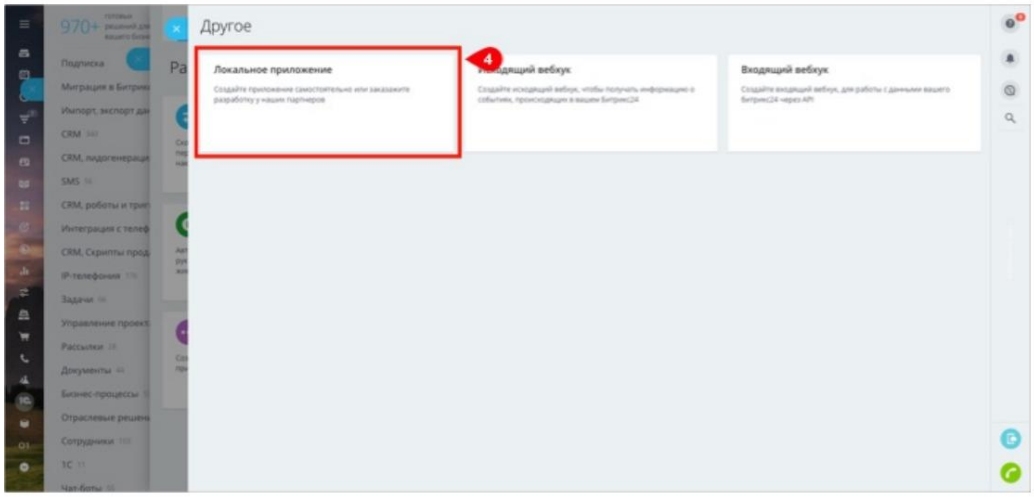
- In the left menu section **of Application** 1, click the **Add Application** 2 button:



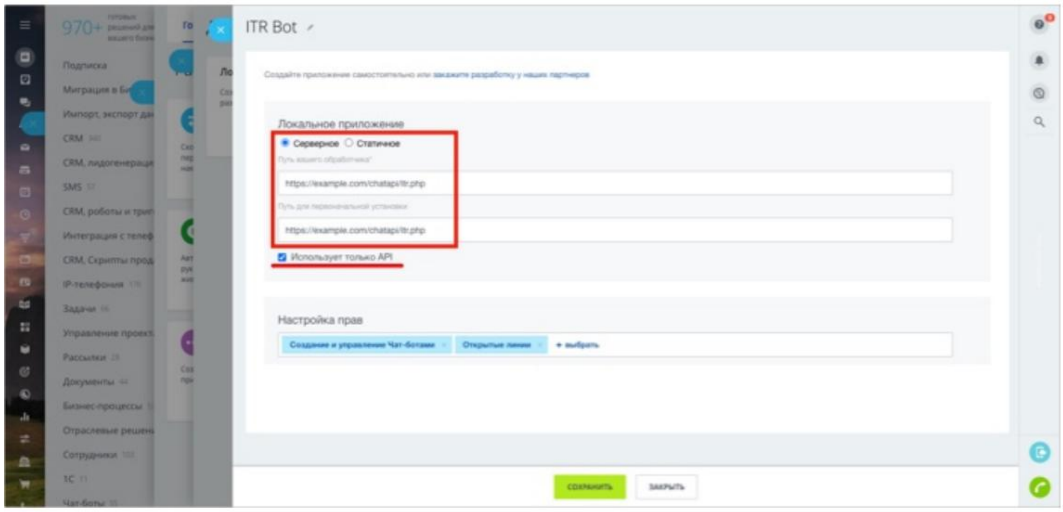
- Next, select **Other** 3:



- And then **the Local Application** 4 script:

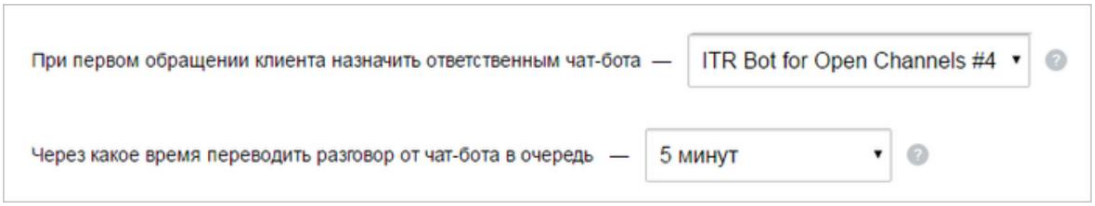


- Select **the Server** application type. Specify the name of the bot, and we will call it "ITR Bot". Enable the **Uses only API** option and give the application permissions, at least to **Create and manage Chatbots** (without these rights, the application will not be able to register a chatbot), as well as to **Open lines** (without these rights, the application will not be able to work with open lines) .



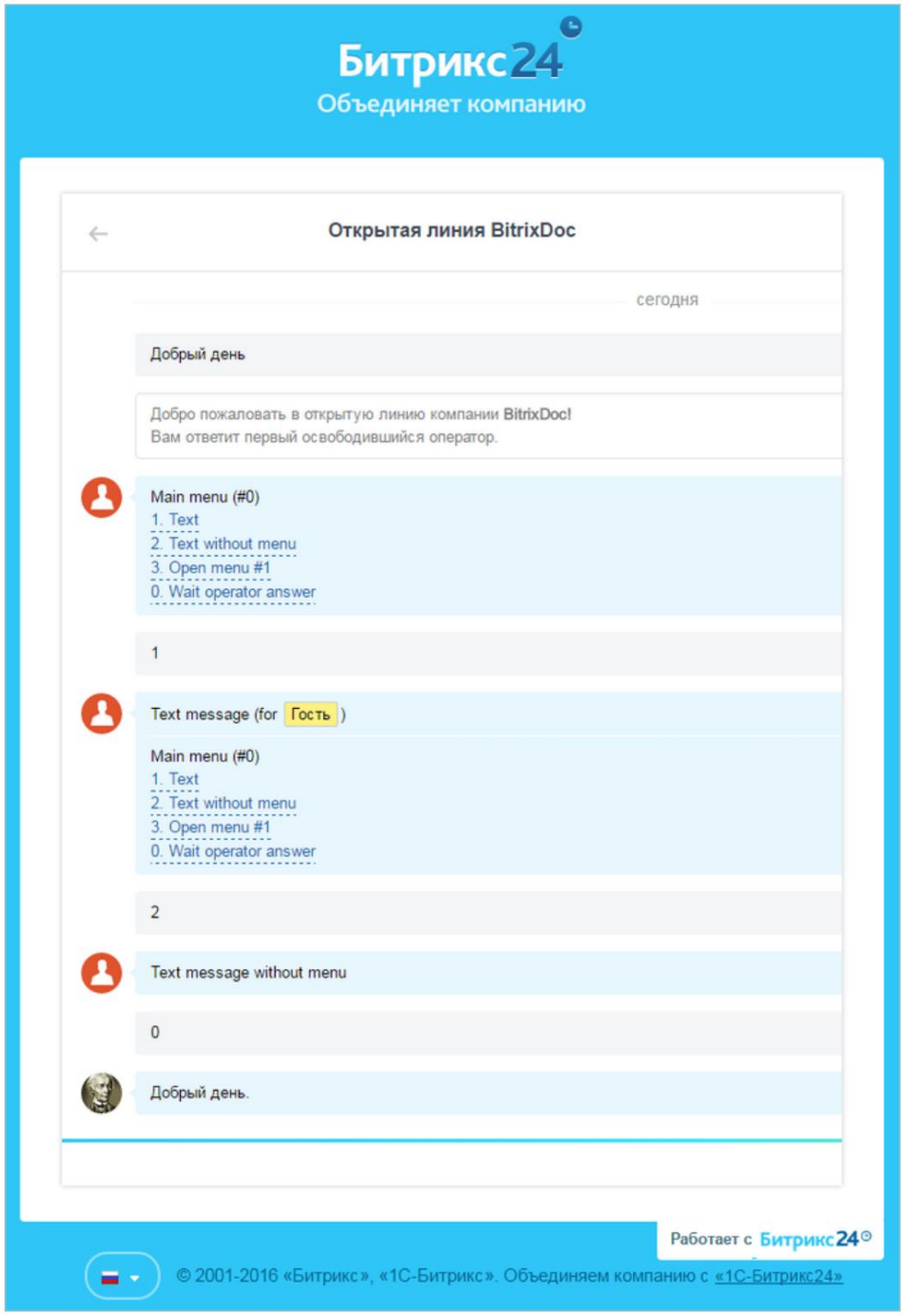
- Since our script is written in such a way that it is a handler for all events, in the application form we will specify both links to the same URL.
- **Please note that** this bot does not publish messages that he was invited to the portal. But after installation, it will be available in the settings of open lines, where you need to select it

responsible and indicate after what time to transfer the conversation from the chatbot to the queue:



Note: *The client can switch to the operator earlier by sending the message 0 (digit "zero") or by selecting the menu item* ***0. Wait operator answer.*** *In general, the user can click on 0 in any chatbot and the user will be redirected to the operator, no additional processing is required.*

- After saving, the bot is ready to go. The example shows a dialog - first ITR Bot answers, the client clicks on the menu items, then the queue goes to the operator (the client selected the menu item **0. Wait operator answer):**



- You can set up your own menu in the ITR Bot in the **itrRun method.**

```
/**
 * Run ITR menu
 *
```

```
     * @param $portalId
     * @param $dialogId
     * @param $userId
     * @param string $message
     * @return bool
     */
    function itrRun($portalId, $dialogId, $userId, $message = '')
    {
        if ($userId <= 0)
            return false;

        $menu0 = new ItrMenu(0);
        $menu0->setText('Main menu (#0)');
        $menu0->addItem(1, 'Text', ItrItem::sendText('Text message (for #USER_NAME#)'));
        $menu0->addItem(2, 'Text without menu', ItrItem::sendText('Text message without menu', true));

        $menu0->addItem(3, 'Open menu #1', ItrItem::openMenu(1));
        $menu0->addItem(0, 'Wait operator answer', ItrItem::sendText('Wait operator answer', true));

        $menu1 = new ItrMenu(1);
        $menu1->setText('Second menu (#1)');
        $menu1->addItem(2, 'Transfer to queue', ItrItem::transferToQueue('Transfer to queue'));
        $menu1->addItem(3, 'Transfer to user', ItrItem::transferToUser(1, false, 'Transfer to user #1'));
        $menu1->addItem(4, 'Transfer to bot', ItrItem::transferToBot('marta', true, 'Transfer to bot
Marta', 'Marta not found :('));
        $menu1->addItem(5, 'Finish session', ItrItem::finishSession('Finish session'));
        $menu1->addItem(6, 'Exec function', ItrItem::execFunction(function($context){
            $result = restCommand('imbot.message.add', Array(
                "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
                "MESSAGE" => 'Function executed (action)',
            ), $_REQUEST["auth"]);
            writeToLog($result, 'Exec function');
        }, 'Function executed (text)'));
        $menu1->addItem(9, 'Back to main menu', ItrItem::openMenu(0));

        $itr = new Itr($portalId, $dialogId, 0, $userId);
        $itr->addMenu($menu0);
```

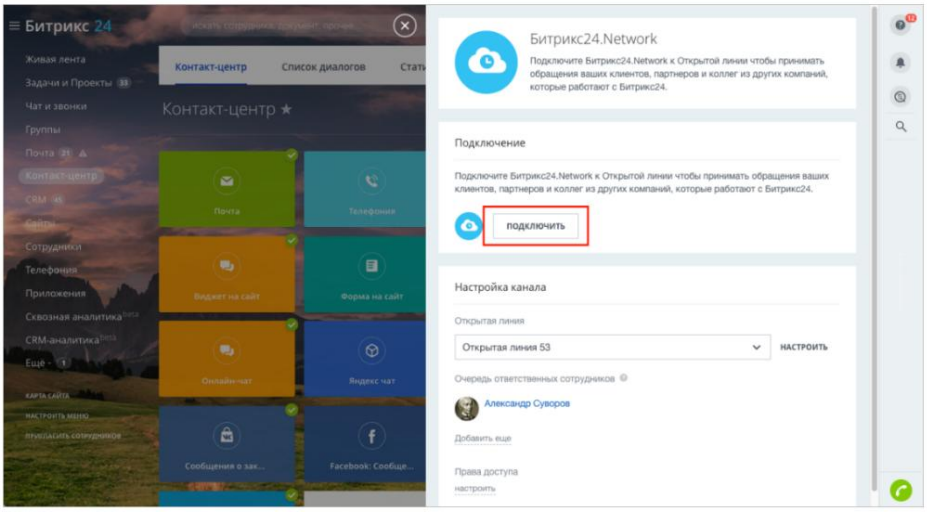
```
        $itr->addMenu($menu1); $itr-
        >run(prepareText($message));

        return true;
    }
```

Example of creating a support channel

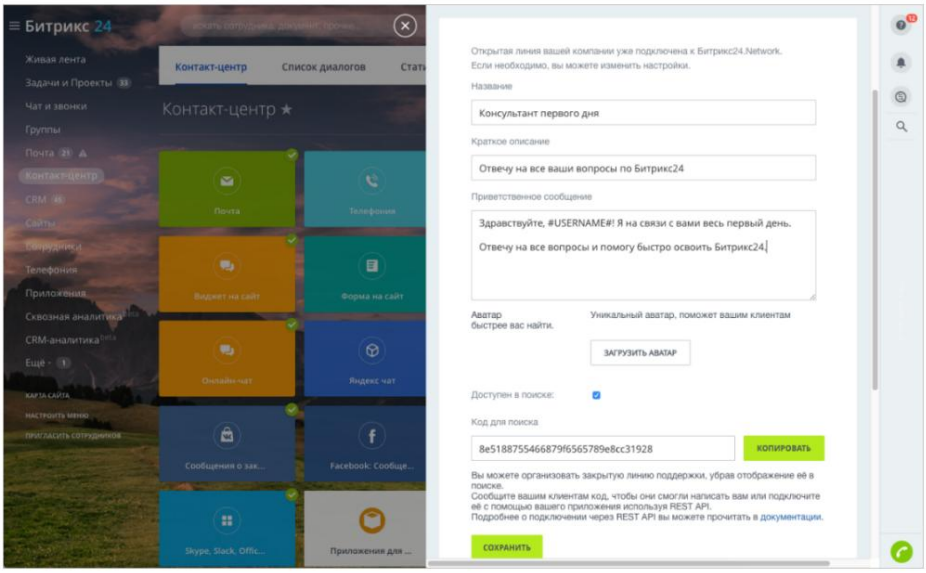
Thanks to the **Open Lines** module, you can easily organize technical support for any ***Bitrix24 application,*** including chat bots.

For this you need:

1. Go to the **Contact Center** section and connect the **Bitrix24.Network communication channel:**

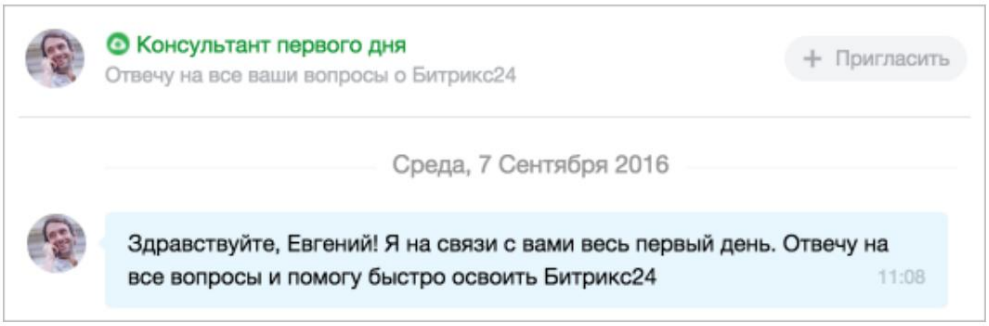


2. After connecting, the Bitrix24.Network connection settings form will be available to you :



- Be sure to fill in **the Name, Short Description** and **Avatar** fields - this will help customers find you more easily.

- **A welcome message** will be automatically sent as soon as the user goes to your ***Bitrix24.Network Open Line:***

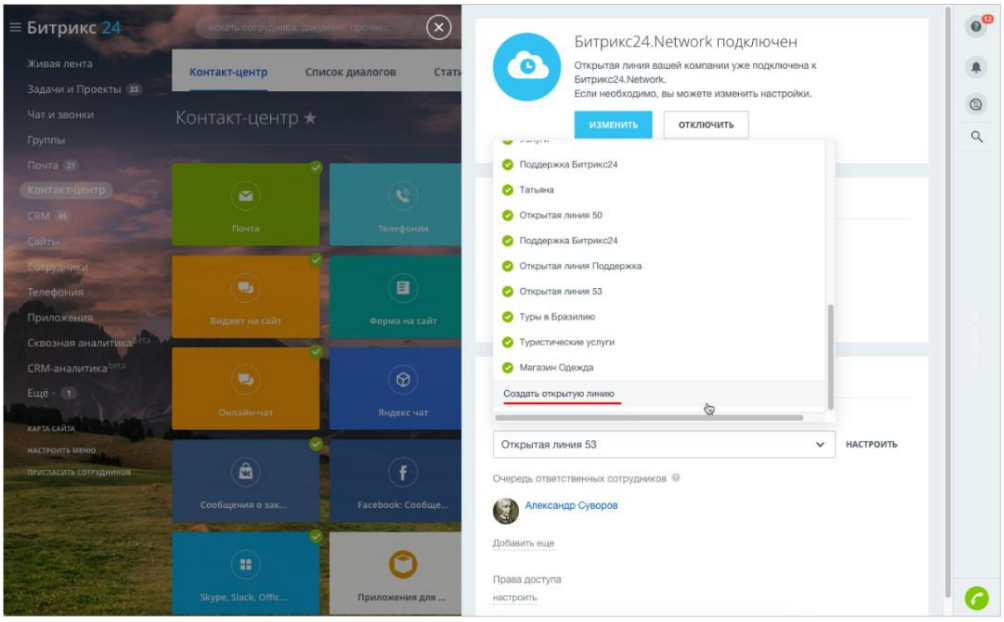


- Pay attention to the checkbox **Available in the search:** if you want to organize a closed support line, you need to turn off this item.

After that, your line will disappear from the **Business Chat** search portal and it will be possible to find it only by entering a unique code in the search.

If the **Available in search** option is enabled, then your open the line can be found by name in the **Business Chat** search of the portal, just like the user of ***Bitrix24.Network.***

3. Create a new or select an existing open technical support line for your product:



To make it convenient for your users, you can use Rest imopenlines.network.join commands automatically connect your open line to them on the portal:

```
$result = restCommand('imopenlines.network.join', Array(

    'CODE' => 'a588e1a88baaf301b9d0b0b33b1eefc2b' // code to search from the page connectors

), $_REQUEST["auth"]);
```

After installing the open line, you can write to the client welcome message using the imopenlines.network.message.add Rest command:

```
Thank you for the installation, we will be happy to help, if you have any questions - write to this chat.
Have a good day! :)
```

Note: The restCommand function is a method for sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript-

method BX24.callMethod or bitrix24-php-sdk. It is also possible to open such a support channel via the BX24.im.openMessenger JavaScript method .

Possible types of Bitrix24 chatbots

For a clearer understanding of what chatbots can do on the platform ***Bitrix24,*** we will give video examples of the capabilities of ready-made chat bots. The types of chatbots are given conditionally, you can create a chatbot that can combine 2 or 3 types.

https://www.youtube.com/watch?v=tcRytBQ-Gzw

Personal assistant

- **Martha** (1C-Bitrix) is your personal assistant. she helps to find answers to questions, reminds about meetings, you can play Tic-tac-toe with her, and she is ready to just talk.

reference system

- **Details of the counterparty** ("1C-Bitrix") - searches the database of the Federal Tax Service and provides the necessary information.
- **Poisk24** (G-Tech studio) is your personal search robot. Searches in Yandex and displays the 5 most accurate answers.

For special tasks

- **Support Bot** ("1C-Bitrix") - an internal chat bot for quick access to the Bitrix24 ticket system.

Working with services

- **OCR Bot** (First Open Systems) - recognizes scans of any documents and saves them in RTF.
- **Translator** (PWEB) - helps to translate texts from one language to another. Supports translation into 63 languages.

Working with open lines

- **Chatbot for Open Lines** ("1C-Bitrix") will be able to act as an assistant at the first contact with the user or help you in the middle of a dialogue.

For fun

- **Andreika** (PWEB) is a master of jokes. Supports dialogue
  jokes and funny quotes.
- **Giphy** ("1C-Bitrix") - search through a large library of animated GIFs, which makes it easy to find the desired image and share it with colleagues.

Note: The full list of chatbots for the Bitrix24 platform can be seen in the section of the same name Bitrix24 app store.

Messages

In this chapter, we discuss how messages appear in ***Bitrix24 chats,*** their creation, formatting and application.

Formatting

You can format the message, add bold,
cross out and insert quotes. The lesson provides user commands,
appearance and REST API for message formatting
chat.

Note! All methods are specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

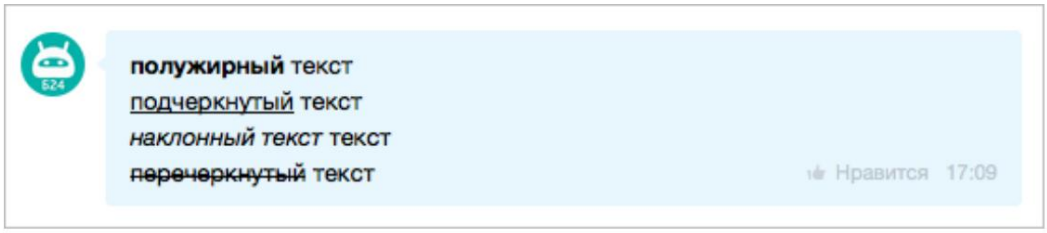
- Format codes
- Line break
- Quoting
- Links
-
- Indentation Active links (commands)
- Icons

Format codes

If you send this message to the chat:

```
[B]bold[/B] text
[U]underlined[/U] text
[I]Italic text[/I] text
[S]strikethrough[/S] text
```

Then it will display like this:



Formatting with rest-api:

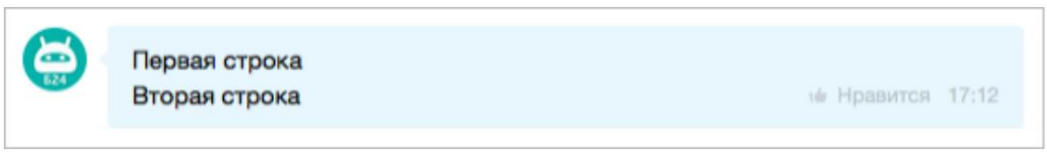
```
restCommand('imbot.message.add', Array(
  "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
  "MESSAGE" => "[B]bold[/B] text
[U]underlined[/U] text
[I]Italic text[/I] text
[S]strikethrough[/S] text",
), $_REQUEST["auth"]);
```

Line break

Line wrapping is carried out by adding characters to the text:

```
[BR]
#BR# (no spaces)
\n
```

Any of these codes will generate a line break:



Line wrapping with rest-api:

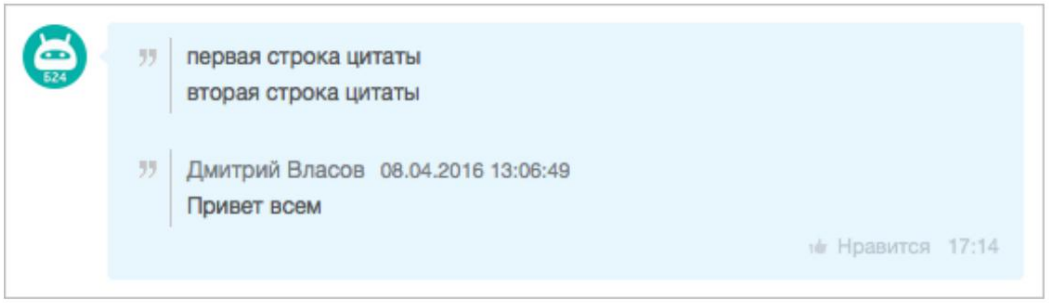
```
restCommand('imbot.message.add', Array(
    "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
    "MESSAGE" => "First line[BR]Second line",
), $_REQUEST["auth"]);
```

Citation

Quoting text can be done in two ways:

```
>>first line of quote
>>second line of quote

------------------------------------------------------

Dmitry Vlasov [04/08/2016 13:06:49]
Hi all
------------------------------------------------------
```

The type of citation will be slightly different - in the second case, the author and time of writing the quotation will be indicated:



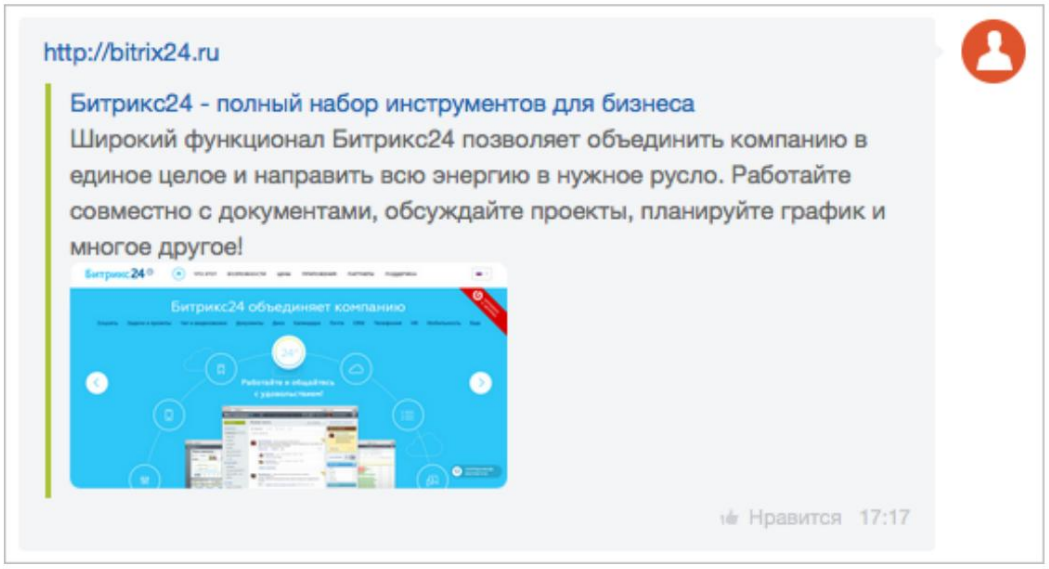
Quoting with rest-api:

```
restCommand('imbot.message.add', Array(
    "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
    "MESSAGE" => ">>first line of quote
>>second line of quote

------------------------------------------------------

Dmitry Vlasov [04/08/2016 13:06:49]
Hi all
------------------------------------------------------",
```

```
), $_REQUEST["auth"]);
```

Links

Any link in the text will automatically become clickable. If the link address has “rich formatting”, then the link will automatically
will pick it up:



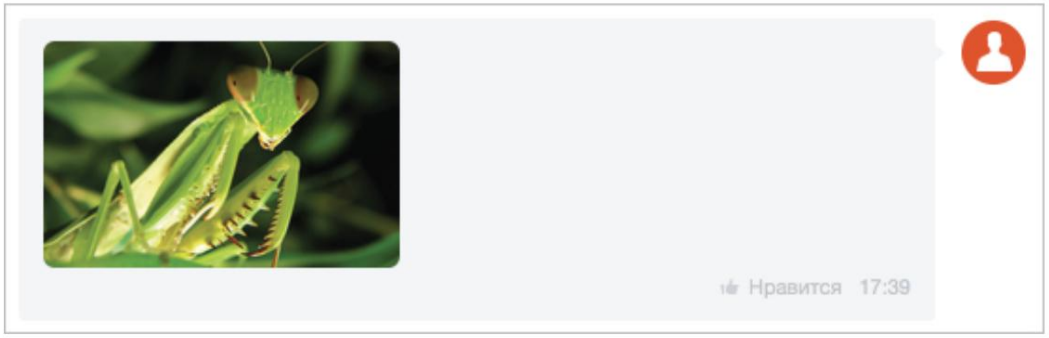
Links using rest-api:

```
restCommand('imbot.message.add', Array(
    "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
    "MESSAGE" => "http:// bitrix24.ru",
), $_REQUEST["auth"]);
```

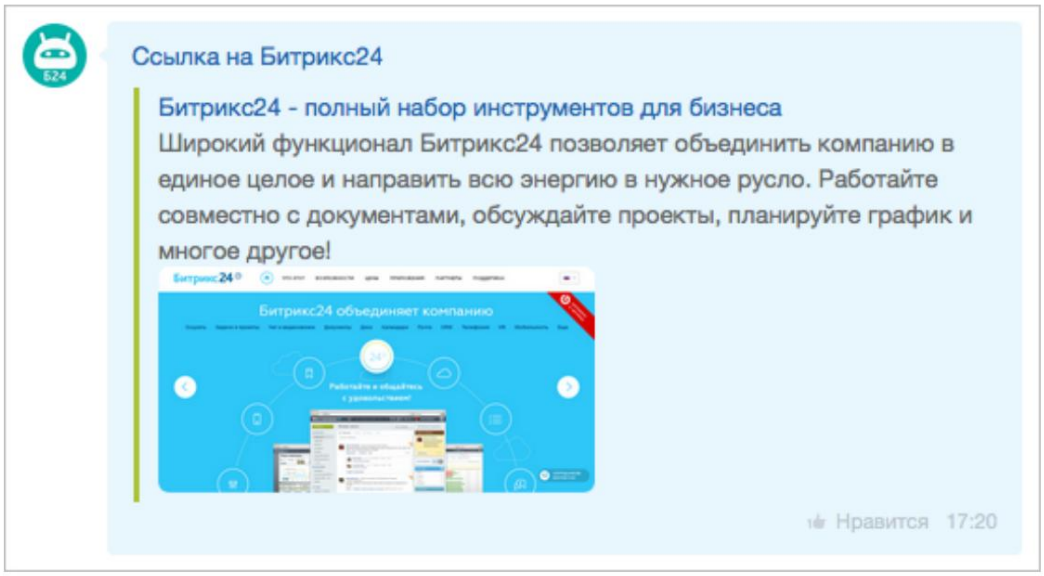
Note: "Rich Formatting" can be disabled by passing the 'URL_PREVIEW' => 'N' key ***when creating a post:***

```
restCommand('imbot.message.add', Array(
 "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
 "MESSAGE" => "http:// bitrix24.ru",
 'URL_PREVIEW' => 'N'
), $_REQUEST["auth"]);
```

If you send a link to an image https://files.shelenkov.com/bitrix/images/mantis.jpg (the end of the link must be .png, .jpg, .gif), then it will be automatically converted to an image:



You can generate a link yourself through the **URL** code - **[URL=http://bitrix24.ru]Link to Bitrix24[/URL]:**



Forming a link using rest-api:

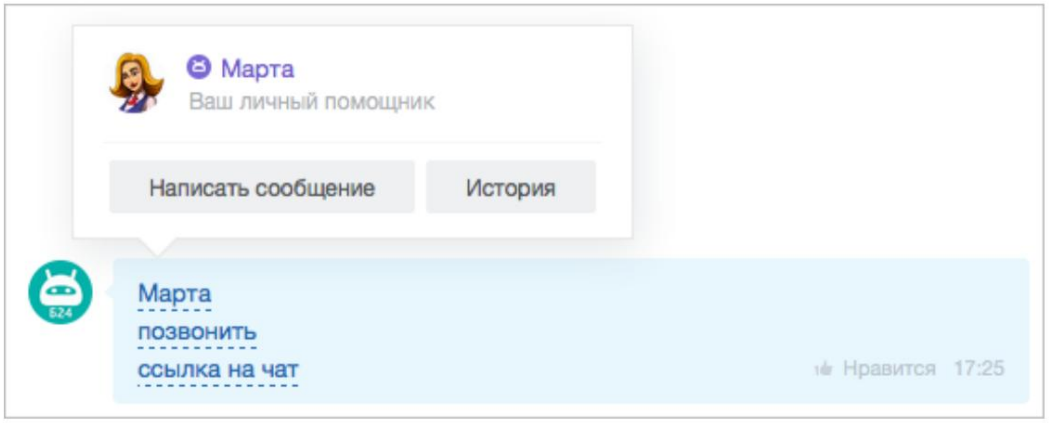
```
restCommand('imbot.message.add', Array(
    "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
    "MESSAGE" => "[URL=http:// bitrix24.ru]Link to Bitrix24[/ URL]",
), $_REQUEST["auth"]);
```

By analogy with the **URL code,** there are special codes for links inside the messenger.

- [USER=5]Marta[/USER] - user link.

- [CALL=84012334455]call[/CALL] - button for making a call via *Bitrix24.*

- [CHAT=12]link to chat[/CHAT] - link to chat.

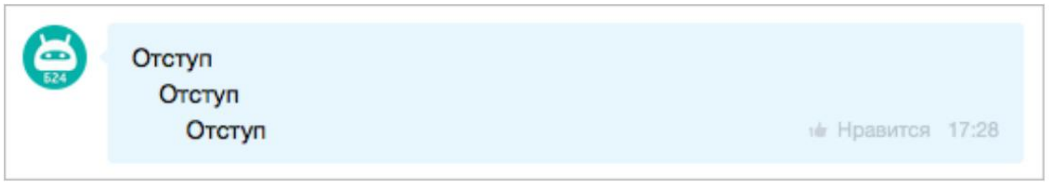


Formatting custom codes with rest-api:

```
restCommand('imbot.message.add', Array(
   "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
   "MESSAGE" => "[USER=5]Marta[/ USER]
[CALL=84012334455]call[/ CALL]
[CHAT=12]chat link[/ CHAT]",
), $_REQUEST["auth"]);
```

Indentation

To indent a message, specify a tab character:



Indentation with rest-api:

```
restCommand('imbot.message.add', Array(
   "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
   "MESSAGE" => "
Indent
  Indent
    Indent",
```

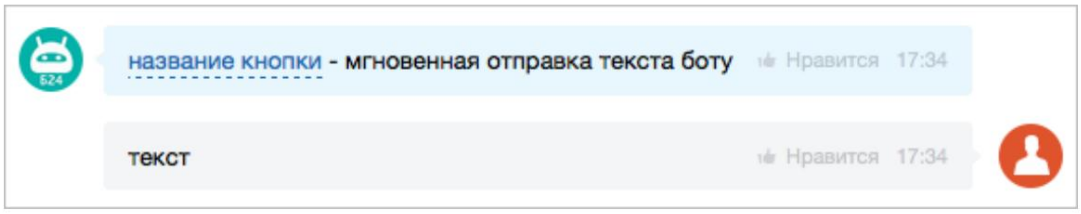
```
), $_REQUEST["auth"]);
```

Active links (commands)

If you want the user to send some text when clicking on the link, use the **SEND tag:**

```
[send=text]button name[/send] - instant text sending to the bot.
```

With this tag, you can force the user to send command of your bot, but there is a more preferred way - Typed keyboards



Active links (commands) using rest-api:

```
restCommand('imbot.message.add', Array(
  "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
  "MESSAGE" => "[send=text]button name[/send] - instantly send text to bot",
), $_REQUEST["auth"]);
```

If it is necessary for the user to add something to the command, use **PUT code:**

```
[put=/search]Enter search string[/put]
```

With this tag, you can allow the user to provide additional data for your command instead of the user typing the command manually. After clicking on such a link, the text specified in the **PUT** tag will be substituted in the input field and the user will only need to add the necessary data and click the submit button (but there is a more preferable method - Typing keyboards):

Sending a bot command using rest-api:

```
restCommand('imbot.message.add', Array(
    "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
    "MESSAGE" => "[put=/search]Enter search string[/put]",
), $_REQUEST["auth"]);
```

Icons

Adding your icon to the message is done by sending the code:

```
[icon=http://files.shelenkov.com/images/unicorn.png size=30 title=ÿÿÿÿÿÿÿÿ]
```

The icon will appear like this:

Also, after that, the icon will be added to the set of Business Chat emoticons. You can remove an icon from a set by right-clicking on an icon in the set and selecting **Delete.**

A mandatory property is to specify the path to the image (without spaces).

Additional attributes available:

- **title** - title;
- **height** - height;
- **width** - width;
- **size** - height and width;

For the best display on all devices, the icon size should be twice as large as specified in the display options.

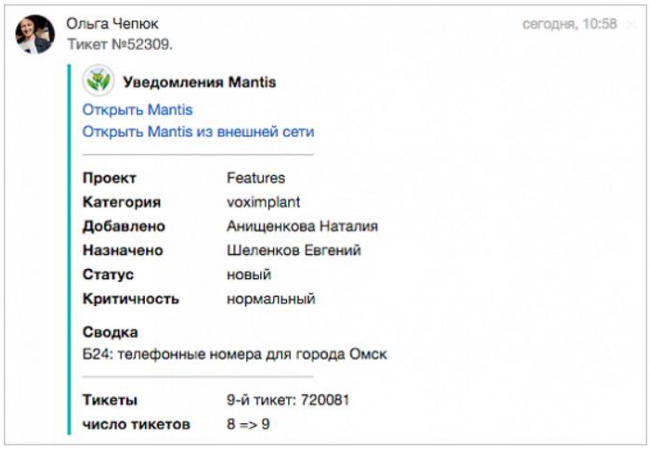
Adding your icon using rest-api:

```
restCommand('imbot.message.add', Array(
    "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
    "MESSAGE" => "[icon=http:// files.shelenkov.com/ images/ unicorn.png size=30 title=ÿÿÿÿÿÿÿÿ]",
), $_REQUEST["auth"]);
```

Note Learn more about how to use advanced attachments. read the format inside the message here.

Attachments

Attachments can be applied to any messages (user or bot) and notifications within the messenger.



How to use attachments

You collect an **Attachment** object and pass it to the send message method in the **ATTACH** key (this can be **a full** or **shortened** form of an attachment).

Full version of the ATTACH object

JavaScript:

```
ATTACH: {
    ID: 1,
    COLOR: "#29619b",
    BLOCKS: [
```

```
        {...},
        {...},
      ]
    }
```

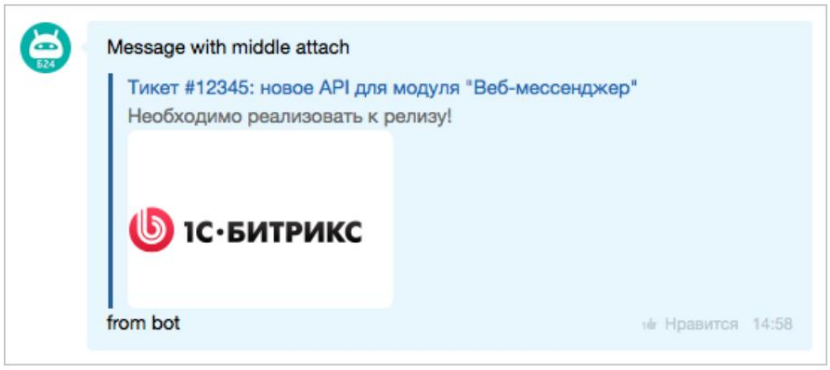
PHP:

```
"ATTACH" => Array(
  "ID" => 1,
  "COLOR" => "#29619b",
  "BLOCKS" => Array(
    array(...),
    array(...),
  )
)
```

Array keys:

- ID - block identifier.
- COLOR - responsible for the color highlighting of the attachment. By default, the color of the attachment is assigned to the color of the recipient's chat (or if it is a notification, to the color of the current user). This key can be omitted if not required.
- BLOCKS must contain the markup blocks that we

  Let's take a look below.



Example:

JavaScript:

```
```

```
BX24.callMethod('imbot.message.add', {
    DIALOG_ID: 'chat20921',
    MESSAGE: 'Message from bot',
    ATTACH: {
        ID: 1,
        COLOR: "#29619b",
        BLOCKS: [
         {LINK: {
            NAME: "Ticket #12345: new API for \"Web Messenger\" module",
            DESC: "Must be implemented by release!",
            LINK: "https:// api.bitrix24.com/"
         }},
         {IMAGE: {
            NAME: "Example implementation",
            LINK: "http:// dev.1c-bitrix.ru/ bitrix/ templates/ 1c-bitrix-new/ images/ logo.png",
         }}
        ]
      }
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP:

```
restCommand('imbot.message.add', Array(
    "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
    "MESSAGE" => "Message with attach",
    "ATTACH" => Array(
       "ID" => 1,
```

```
        "COLOR" => "#29619b",
        "BLOCKS" => Array(
          Array("LINK" => Array(
              "NAME" => "Ticket #12345: New API for \"Web Messenger\" module",
              "DESC" => "Must be implemented by release!",
              "LINK" => "https:// api.bitrix24.com/"
          )),
          Array("IMAGE" => Array(
              "NAME" => "Example of implementation",
              "LINK" => "http:// dev.1c-bitrix.ru/ bitrix/ templates/ 1c-bitrix-new/ images/ logo.png",
          ))
        )
    )
), $_REQUEST["auth"]);
```

Short version of the ATTACH object

If you're happy with the attachment at the bottom of the post and you don't need to specify a color, you can use **the short** version:

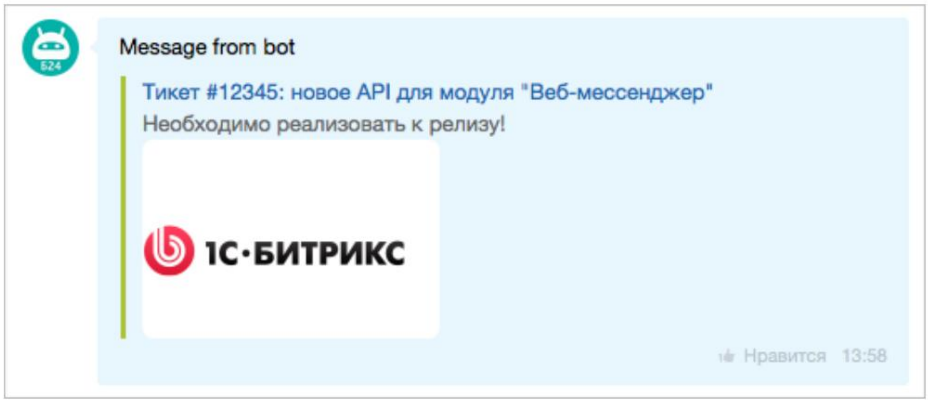
JavaScript:

```
    ATTACH: [
      {...},
      {...},
    ]
```

PHP:

```
  "ATTACH" => Array(
    array(...),
    array(...),
  )
```

Unlike the full version, markup blocks are immediately specified at the first level, without declaring the **BLOCKS** key



Example:

JavaScript:

```
BX24.callMethod('imbot.message.add', {
    DIALOG_ID: 'chat20921',
    MESSAGE: 'Message from bot',
    ATTACH: [
        {LINK: {
            NAME: "Ticket #12345: new API for \"Web Messenger\" module",
            DESC: "Must be implemented by release!",
            LINK: "https:// api.bitrix24.com/"
        }},
        {IMAGE: {
            NAME: "Example implementation",
            LINK: "http:// dev.1c-bitrix.ru/ bitrix/ templates/ 1c-bitrix-new/ images/ logo.png",
        }}
    ]

}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
```

```
});
```

PHP:

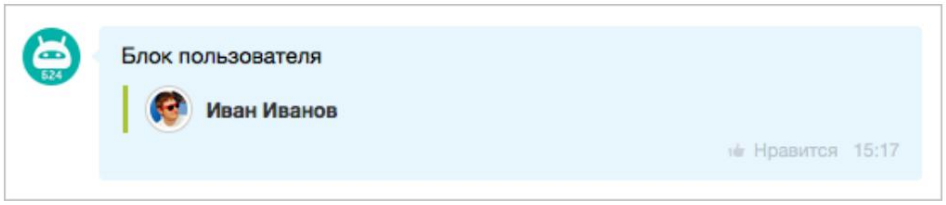
```
restCommand('imbot.message.add', Array(
    "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
    "MESSAGE" => "Message from bot",
    "ATTACH" => Array(
      Array("LINK" => Array(
          "NAME" => "Ticket #12345: New API for \"Web Messenger\" module",
          "DESC" => "Must be implemented by release!",
          "LINK" => "https:// api.bitrix24.com/"
      )),
      Array("IMAGE" => Array(
          "NAME" => "Example of implementation",
          "LINK" => "http:// dev.1c-bitrix.ru/ bitrix/ templates/ 1c-bitrix-new/ images/ logo.png",
      ))
    )
), $_REQUEST["auth"]);
```

Note: the restCommand function is a method for sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

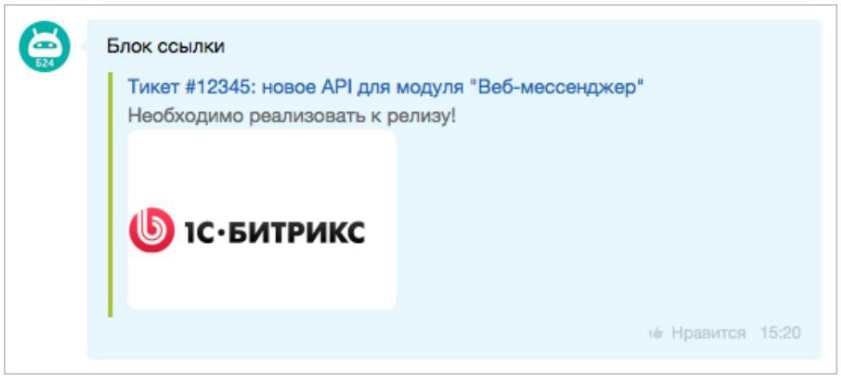
Please note: due to the complexity of the structure, attachments are not automatically added when sent to XMPP, mail, or as a PUSH notification to the phone.

Block Collections

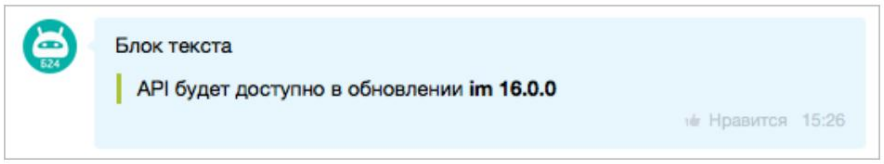
User block (USER) .



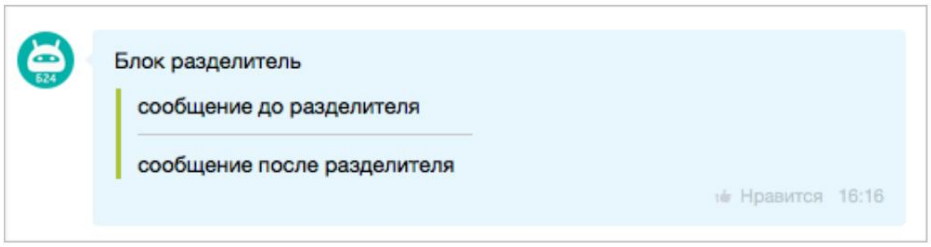
Block with links (LINK) .



Block with text (MESSAGE)

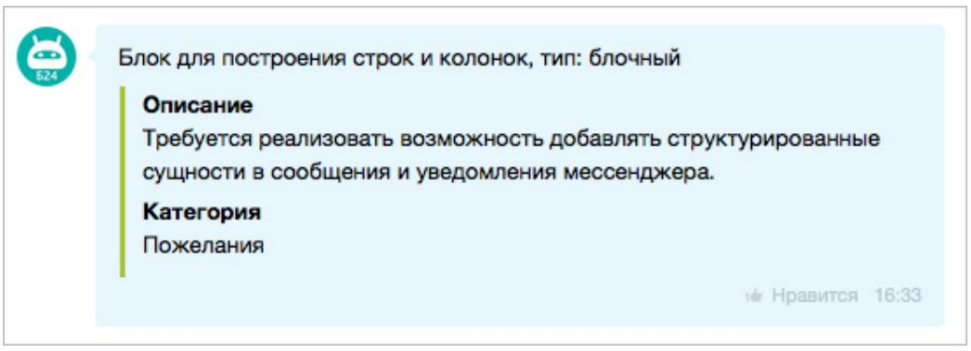


Block with separator (DELIMITER)

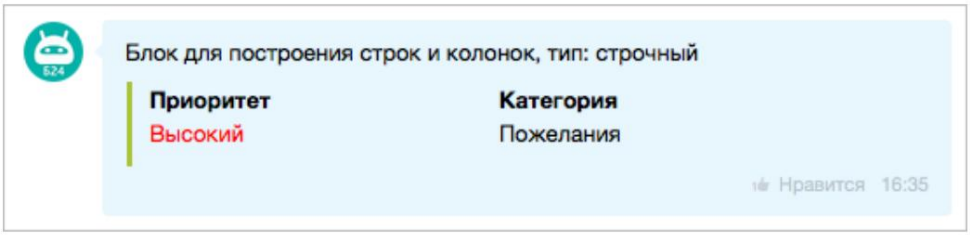


Block for building rows and columns (GRID)

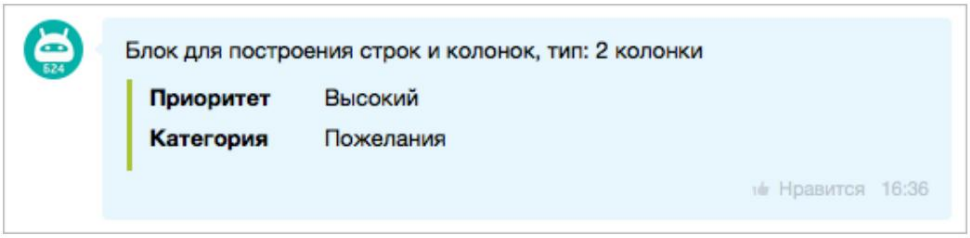
1. ### Block construction (BLOCK)



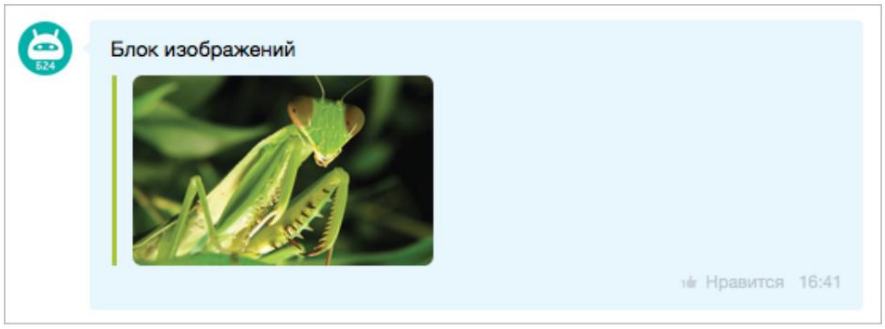
2. ### Line construction (LINE)



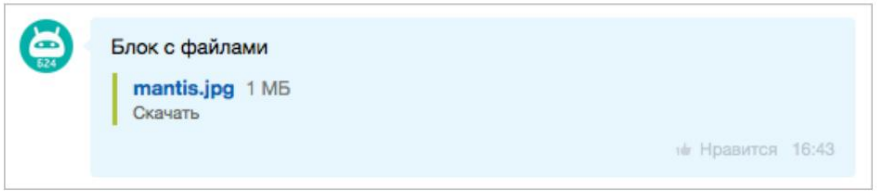
3. **Construction in the form of two columns (COLUMN)**



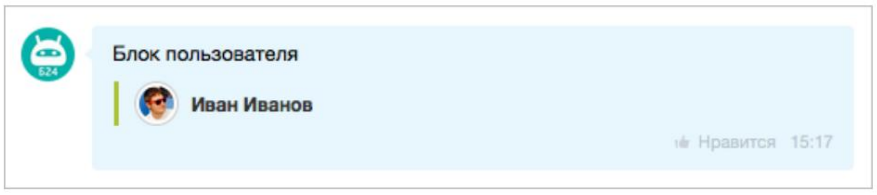
Block with images (IMAGE) .



Block with files (FILE) .



User block



USER - output block with avatar and username.

The AVATAR (avatar) and **LINK** (link) fields are optional.

Example:

JavaScript:

```
{USER: {
    NAME: "Ivan Ivanov",
    AVATAR: "https:// files.shelenkov.com/ bitrix/ images/ avatar.png",
    LINK: "https:// shelenkov.com"
}},
```

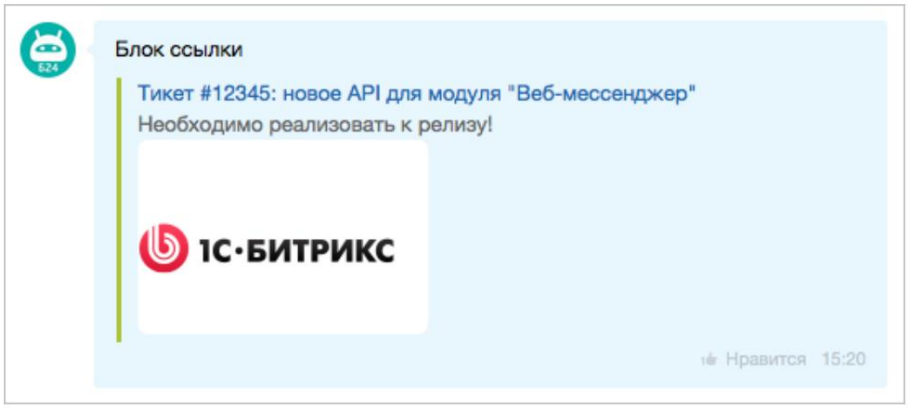
PHP:

```
Array("USER" => Array(
    "NAME" => "Ivan Ivanov",
    "AVATAR" => "https:// files.shelenkov.com/ bitrix/ images/ avatar.png",
    "LINK" => "https:// shelenkov.com/",
)),
```

Instead of the **LINK** key, you can also use links to entities:

- CHAT_ID - to specify a link to the chat;
- BOT_ID - to specify a link to the bot;
- USER_ID - to specify a link to the user.

Block with links



LINK - block output with a link to the resource, description and picture explanation. This block is used in the automatic creation of rich links.

The DESC (description) and **PREVIEW** (picture) fields are not required fields.

The fields **WIDTH** (width) and **HEIGHT** (height) are optional, but it is recommended to specify them now in order to correctly display the image.

Example:

JavaScript:

```
{LINK: {
    PREVIEW: "https:// dev.1c-bitrix.ru/ bitrix/ templates/ 1c-bitrix-new/ images/ logo.png",
    WIDTH: 1000,
    HEIGHT: 638,
    NAME: "Ticket #12345: new API for \"Web Messenger\" module",
    DESC: "Must be implemented by release!",
    LINK: "https:// api.bitrix24.com/",
}},
```

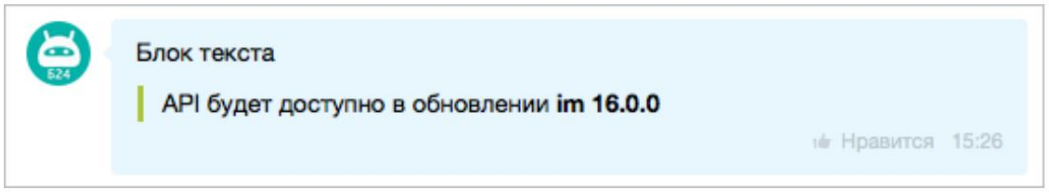
PHP:

```
Array("LINK" => Array(
    "PREVIEW" => "https:// dev.1c-bitrix.ru/ bitrix/ templates/ 1c-bitrix-new/ images/ logo.png",
    "WIDTH" => "1000",
    "HEIGHT" => "638",
    "NAME" => "Ticket #12345: New API for \"Web Messenger\" module",
    "DESC" => "Must be implemented by release!",
    "LINK" => "https:// api.bitrix24.com/" )),
```

Instead of the **LINK** key, you can also use links to entities:

- CHAT_ID => 1 - to specify a link to the chat;
- USER_ID => 1 - to specify a link to the user.

Block with text



MESSAGE - Plain text output without formatting.

Example:

JavaScript:

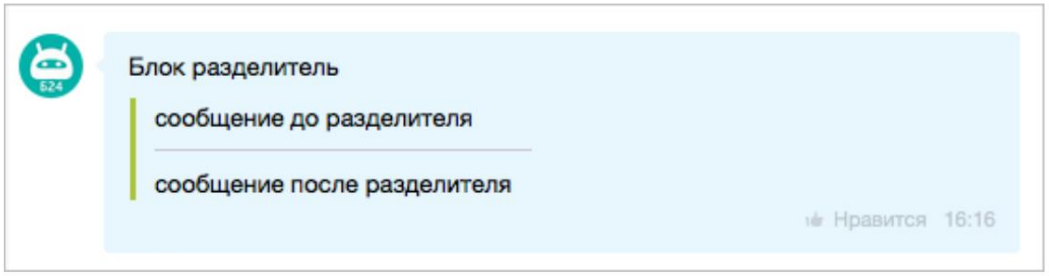
```
{MESSAGE: "API will be available in update [B]im 16.0.0[/B]"},
```

PHP:

```
Array("MESSAGE" => "API will be available in update [B]im 16.0.0[/B]"),
```

bb-codes are available in the text: USER, CHAT, SEND, PUT, CALL, BR, B, U, I, S, URL.

Block with separator



DELIMITER - delimiter output, color and size can be set.

The fields **COLOR** (color) and **SIZE** (size) are optional fields

Example:

JavaScript:

```
{DELIMITER: {SIZE: 200, COLOR: "#c6c6c6"}},
```

PHP:

```
Array("DELIMITER" => Array(
    'SIZE' => 200,
    'COLOR' => "#c6c6c6"
)),
```

Block for building rows and columns

GRID - there are three representations of the block: **block, inline** and **in the form of 2 columns.**

Attention In all 3 representations it is impossible to mix elements within one record. If you need to use blocks of different types at the same time, you need to create a new block: **JavaScript**

example

```
{GRID: [
    {
        NAME: "Priority",
        VALUE: "high",
        DISPLAY: "COLUMN",
    },
    {
        NAME: "Category",
        VALUE: "Wishes",
        DISPLAY: "COLUMN",
    },
]},

{GRID: [
    {
        NAME: "Description",
        VALUE: "It is required to implement the ability to add structured entities to messenger messages and
notifications.",
        DISPLAY: "BLOCK",
        WIDTH: 250
```

```
        },
        {
            NAME: "Category",
            VALUE: "Wishes",
            DISPLAY: "BLOCK",
            WIDTH: 100
        },
    ]},
```

Example of PHP

```
    Array("GRID" => Array(
        Array(
            "NAME" => "Priority",
            "VALUE" => "High",
            "DISPLAY" => "COLUMN",
        ),
        Array(
            "NAME" => "Category",
            "VALUE" => "Wish",
            "DISPLAY" => "COLUMN"
        ),
    )),

    Array("GRID" => Array(
        Array(
            "NAME" => "Description",
            "VALUE" => "It is required to implement the ability to add structured
entities into messenger messages and notifications.",
            "DISPLAY" => "BLOCK",
            "WIDTH" => "250"
        ),
        Array(
```

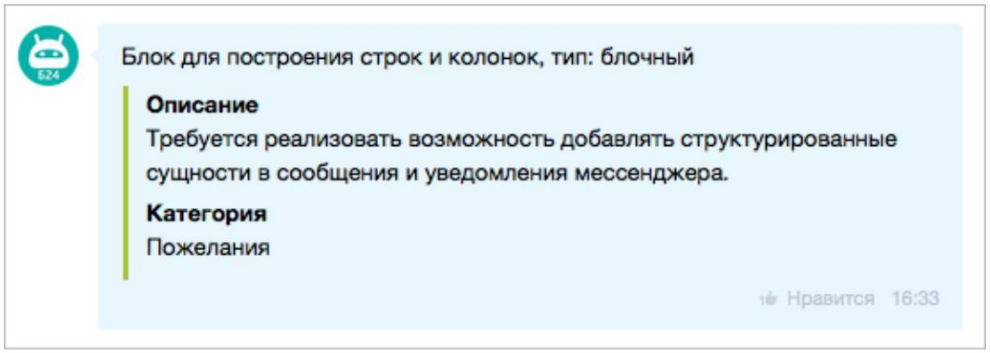
```
        "NAME" => "Category",
        "VALUE" => "Wish",
        "DISPLAY" => "BLOCK"
    ),
)),
```

For all types of blocks for building rows and columns are available additional keys:

- **COLOR** - value color;
- **CHAT_ID** - the value will become a link to the chat in the web messenger;
- **USER_ID** - the value will become a link to the user in the web messenger;
- **LINK** - the value will become a link.

- bb-codes are available for the **VALUE** key: USER, CHAT, SEND, PUT, CALL, BR, B , U, I, S, URL.

Block representation



"DISPLAY" => "BLOCK" - the data goes under each other.

A **WIDTH** key is available to specify the block width (in pixels).

Example:

JavaScript:

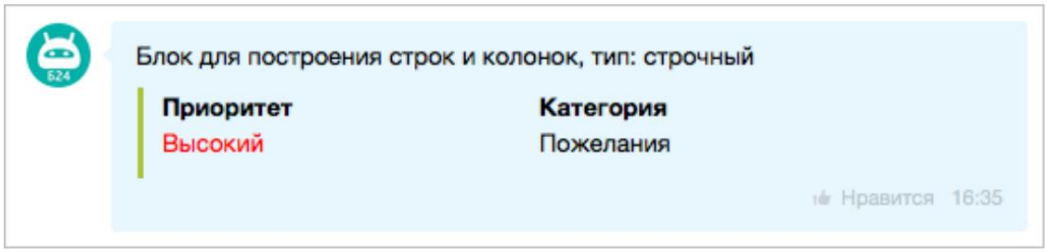
```
{GRID: [
    {
```

```
        NAME: "Description",
        VALUE: "It is required to implement the ability to add structured entities to messenger messages and
notifications.",
        DISPLAY: "BLOCK",
        WIDTH: 250
    },
    {
        NAME: "Category",
        VALUE: "Wishes",
        DISPLAY: "BLOCK",
        WIDTH: 100
    },
]},
```

PHP:

```
Array("GRID" => Array(
    Array(
        "NAME" => "Description",
        "VALUE" => "It is required to implement the ability to add structured
entities into messenger messages and notifications.",
        "DISPLAY" => "BLOCK",
        "WIDTH" => "250"
    ),
    Array(
        "NAME" => "Category",
        "VALUE" => "Wish",
        "DISPLAY" => "BLOCK"
    ),
)),
```

Line representation



"DISPLAY" => "LINE" - each block goes one after another until it ends
available space, after which it wraps to a new line.

A **WIDTH** key is available to specify the block width (in pixels).

Example:

JavaScript:

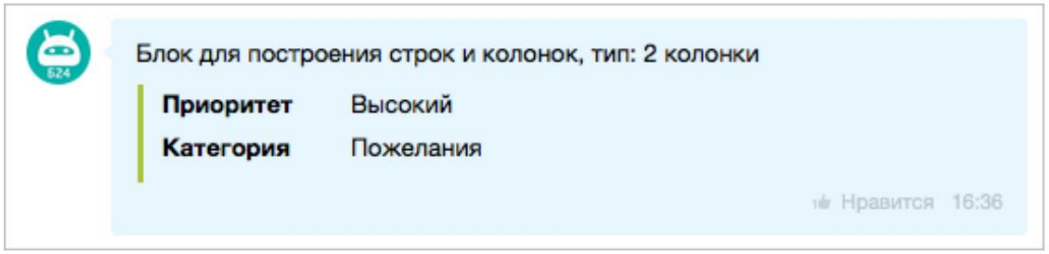
```
{GRID: [
  {
    NAME: "Priority",
    VALUE: "high",
    COLOR: "#ff0000",
    DISPLAY: "LINE",
    WIDTH: 250
  },
  {
    NAME: "Category",
    VALUE: "Wishes",
    DISPLAY: "LINE",
  },
]},
```

PHP:

```
Array("GRID" => Array(
  Array(
    "NAME" => "Priority",
    "VALUE" => "High",
    "COLOR" => "#ff0000",
    "DISPLAY" => "LINE",
     "WIDTH" => "250"
```

```
    ),
    Array(
        "NAME" => "Category",
        "VALUE" => "Wish",
        "DISPLAY" => "LINE"
    ),
)),
```

Two column view



"DISPLAY" => "COLUMN" - construction in the form of two columns.

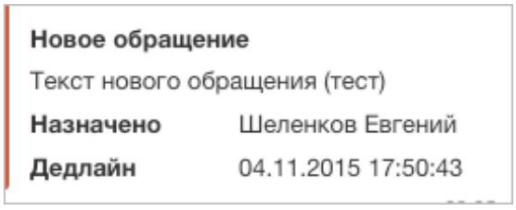
A **WIDTH** key is available to specify the width of the first column (in pixels).

Starting with version 22 of REST, in a two-column view, you can do not pass one of the required **NAME** or **VALUE parameters:**

- if you do not pass **VALUE,** then **NAME** will be the entire width of the table;
- if you do not pass **NAME,** then **VALUE** will be full width columns.

Result example:

The first line is no **VALUE,** the second is no **NAME,** the third and fourth lines are both **NAME** and **VALUE:**



Example:

JavaScript:

```
{GRID: [
    {
        NAME: "Priority",
        VALUE: "high",
        DISPLAY: "COLUMN",
    },
    {
        NAME: "Category",
        VALUE: "Wishes",
        DISPLAY: "COLUMN",
    },
]},
```

PHP:

```
Array("GRID" => Array(
    Array(
        "NAME" => "Priority",
        "VALUE" => "High",
        "DISPLAY" => "COLUMN"
        "WIDTH" => "250"
    ),
    Array(
        "NAME" => "Category",
        "VALUE" => "Wish",
        "DISPLAY" => "COLUMN"
    ),
)),
```

Block with images

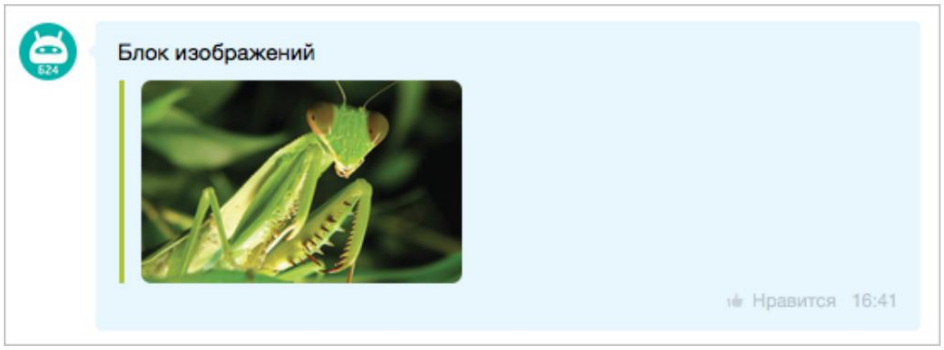


IMAGE - block with images.

It is recommended to fill in the **PREVIEW** field indicating the reduced copies of the image, if the field is empty, **LINK is used.**

The NAME (title) and **PREVIEW** (preview image) fields are optional.

The fields **WIDTH** (width) and **HEIGHT** (height) are optional, but it is recommended to specify them now in order to correctly display the image.

Example:

JavaScript:

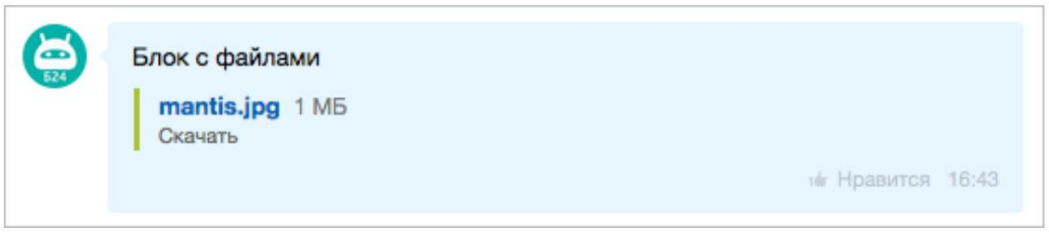
```
{IMAGE: {
    NAME: "This is Mantis"
    LINK: "https:// files.shelenkov.com/ bitrix/ images/ mantis.jpg",
    PREVIEW: "https:// files.shelenkov.com/ bitrix/ images/ mantis.jpg",
    WIDTH: 1000,
    HEIGHT: 638,
}},
```

PHP:

```
Array("IMAGE" => Array(
    Array(
        "NAME" => "This is Mantis",
        "LINK" => "https:// files.shelenkov.com/ bitrix/ images/ mantis.jpg",
        "PREVIEW" => "https:// files.shelenkov.com/ bitrix/ images/ mantis.jpg",
        "WIDTH" => "1000",
        "HEIGHT" => "638"
```

```
    )
)),
```

Block with files



FILE - Displays a formatted link to download a file.

The file size must be specified in bytes. The fields

NAME (file name), **SIZE** (file size) are optional.

Example:

JavaScript:

```
{FILE: {
    NAME: "mantis.jpg",
    LINK: "https:// files.shelenkov.com/ bitrix/ images/ mantis.jpg",
    SIZE: 1500000,
}},
```

PHP:

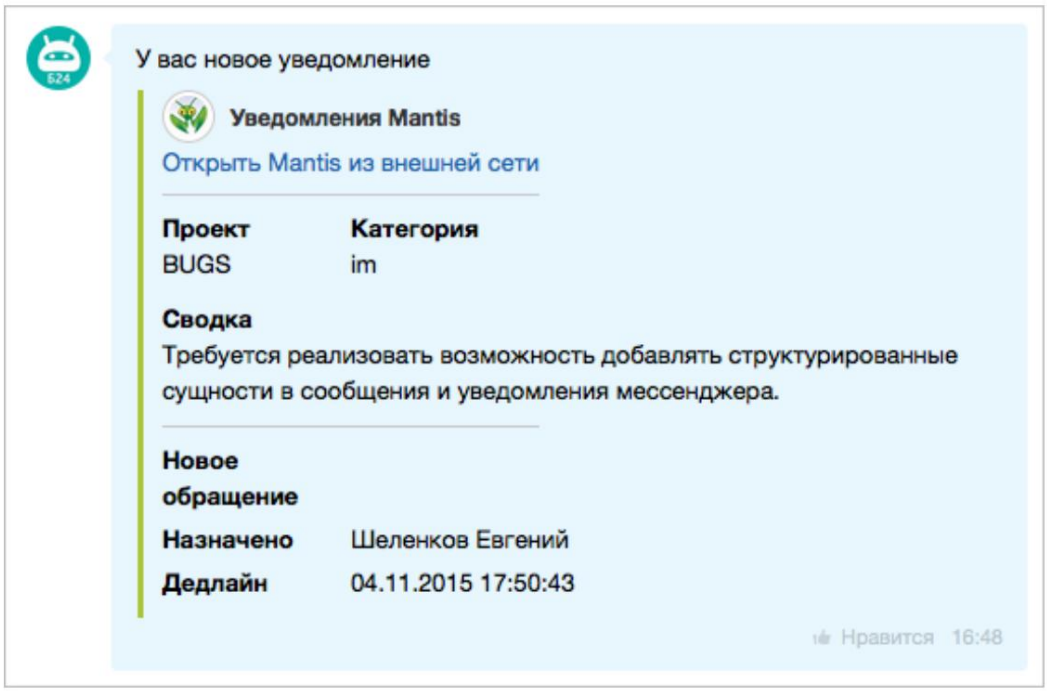
```
Array("FILE" => Array(
    Array(
        "NAME" => "mantis.jpg",
        "LINK" => "https:// files.shelenkov.com/ bitrix/ images/ mantis.jpg",
        "SIZE" => "1500000"
    )
)),
```

Constructor, examples

The **Attachment** object is a constructor, you can "build" it however you want using the available blocks. The order in which blocks are added matters.

- Bug tracker example
- Example "Information sheet"

"bug tracker"



JavaScript:

```
BX24.callMethod('imbot.message.add', {
    DIALOG_ID: 'chat20921',
    MESSAGE: 'Message from bot',
    ATTACH: [
      {USER: {
          NAME: "Mantis notifications",
          AVATAR: "http://files.shelenkov.com/bitrix/images/mantis2.jpg",
          LINK: "http://shelenkov.com/",
      }},
      {LINK: {
          NAME: "Open Mantis from external network",
```

```
    LINK: "http:// shelenkov.com/",
}},
{DELIMITER: {
```

```
        SIZE: 200,
        COLOR: "#c6c6c6"
    }},
    {GRID: [
        {
            NAME: "Project",
            VALUE: "BUGS",
            DISPLAY: "LINE",
            WIDTH: 100
        },
        {
            NAME: "Category",
            VALUE: "im",
            DISPLAY: "LINE",
            WIDTH: 100
        },
        {
            NAME: "Summary",
            VALUE: "It is required to implement the ability to add structured
entities into messenger messages and notifications.",
            DISPLAY: "BLOCK"
        },
    ]},
    {DELIMITER: {
        SIZE: 200,
        COLOR: "#c6c6c6"
    }},
    {GRID: [
        {
            NAME: "New Appeal",
            VALUE: "",
            DISPLAY: "ROW",
            WIDTH: 100
        },
        {
```

```
                NAME: "Appointed",
                VALUE: "Shelenkov Evgeniy",
                DISPLAY: "ROW",
                WIDTH: 100
            },
            {
                NAME: "Deadline",
                VALUE: "04.11.2015 17:50:43",
                DISPLAY: "ROW",
                WIDTH: 100
            },
        ]},
    ]
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

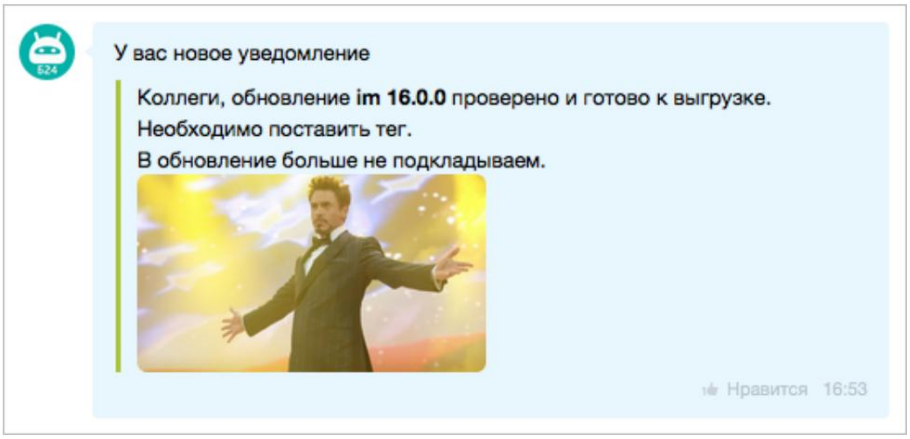
PHP:

```
restCommand('imbot.message.add', Array(
    "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
    "MESSAGE" => "You have a new notification",
    "ATTACH" => Array(
      Array("USER" => Array(
          "NAME" => "Mantis notifications",
          "AVATAR" => "http://files.shelenkov.com/bitrix/images/mantis2.jpg",
          "LINK" => "http://shelenkov.com/",
      )),
      Array("LINK" => Array(
```

```
            "NAME" => "Open Mantis from external network",
            "LINK" => "http://shelenkov.com/",
        )),
        Array("DELIMITER" => Array(
            'SIZE' => 200,
            'COLOR' => "#c6c6c6"
        )),
        Array("GRID" => Array(
            Array(
                "NAME" => "Project",
                "VALUE" => "BUGS",
                "DISPLAY" => "LINE",
                "WIDTH" => 100
            ),
            Array(
                "NAME" => "Category",
                "VALUE" => "im",
                "DISPLAY" => "LINE",
                "WIDTH" => 100
            ),
            Array(
                "NAME" => "Summary",
                "VALUE" => "It is required to implement the ability to add structured
entities into messenger messages and notifications.",
                "DISPLAY" => "BLOCK"
            ),
        )),
        Array("DELIMITER" => Array(
            'SIZE' => 200,
            'COLOR' => "#c6c6c6"
        )),
        Array("GRID" => Array(
            Array(
                "NAME" => "New hit",
                "VALUE" => "",
                "DISPLAY" => "ROW",
```

```
            "WIDTH" => 100
        ),
        Array(
            "NAME" => "Appointed",
            "VALUE" => "Shelenkov Evgeny",
            "DISPLAY" => "ROW",
            "WIDTH" => 100
        ),
        Array(
            "NAME" => "Deadline",
            "VALUE" => "04.11.2015 17:50:43",
            "DISPLAY" => "ROW",
            "WIDTH" => 100
        ),
    )),
)
), $_REQUEST["auth"]);
```

"Information leaflet"



JavaScript:

```
BX24.callMethod('imbot.message.add', {
        DIALOG_ID: 'chat20921',
    MESSAGE: 'You have a new notification',
    ATTACH: {
        ID: 1,
```

```
            COLOR: "#29619b",
            BLOCKS: [
                {MESSAGE: "Colleagues, update [b]im 16.0.0[/b] is checked and ready to upload.[BR]
You need to put a tag.[BR] We don't put it in the update anymore."},
                {IMAGE: {LINK: "http://files.shelenkov.com/bitrix/images/win.jpg"}}
            ]
        }
    }, function(result){
        if(result.error())
        {
            console.error(result.error().ex);
        }
        else
        {
            console.log(result.data());
        }
    });
```

PHP:

```
    restCommand('imbot.message.add', Array(
        "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
        "MESSAGE" => "You have a new notification",
        "ATTACH" => Array(
            Array("MESSAGE" => "Colleagues, update [b]im 16.0.0[/b] has been checked and is ready to be uploaded.
[BR] It is necessary to put a tag.[BR] We do not put it in the update anymore."),
            Array("IMAGE" => Array(
                "LINK" => "http://files.shelenkov.com/bitrix/images/win.jpg",
            )),
        )
    ), $_REQUEST["auth"]);
```

Working with keyboards

Typing keyboards will allow the user to interact with the chatbot by simply pressing buttons. How to add a keyboard to a chatbot

- _____ - - _____ -

- Processing commands by the chatbot
- Keyboard usage examples

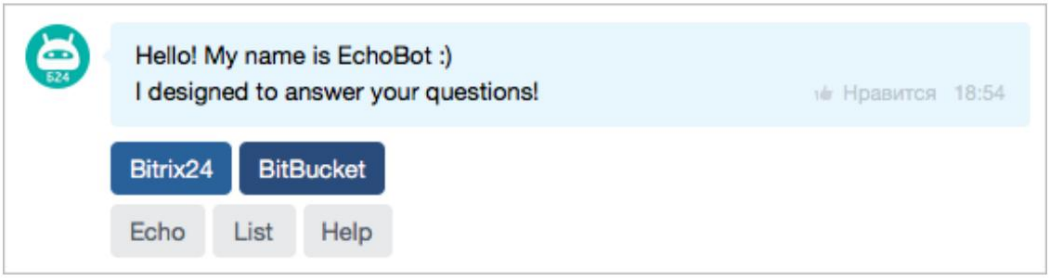
How to add a keyboard to a chatbot

The keyboard is part of the message, when creating a message, you need to add the **KEYBOARD** key and pass the parameters.

Methods that support keyboard:

- imbot.message.add - send a message from the chatbot.
- imbot.message.update - Sending a change message to the chatbot.
- imbot.command.answer - publication of the response to the command.
- im.message.add - send a message to the chat.
- im.message.update - Sending a change message to the chatbot.

Let's take this message as an example:




```
restCommand('imbot.command.answer', Array(
  "COMMAND_ID" => $command['COMMAND_ID'],
  "MESSAGE_ID" => $command['MESSAGE_ID'],
  "MESSAGE" => "Hello! My name is EchoBot :)[br] I designed to answer your questions!",
  "KEYBOARD" => Array(
    Array(
      "TEXT" => "Bitrix24",
      "LINK" => "http:// bitrix24.com",
      "BG_COLOR" => "#29619b",
      "TEXT_COLOR" => "#fff",
      "DISPLAY" => "LINE",
    ),
    Array(
```

```
            "TEXT" => "BitBucket",
            "LINK" => "https:// bitbucket.org/ Bitrix24com/ rest-bot-echotest",
            "BG_COLOR" => "#2a4c7c",
            "TEXT_COLOR" => "#fff",
            "DISPLAY" => "LINE",
        ),
        Array("TYPE" => "NEWLINE"), // line break
        Array("TEXT" => "Echo", "COMMAND" => "echo", "COMMAND_PARAMS" => "test from keyboard",
"DISPLAY" => "LINE"),
        Array("TEXT" => "List", "COMMAND" => "echoList", "DISPLAY" => "LINE"),
        Array("TEXT" => "Help", "COMMAND" => "help", "DISPLAY" => "LINE"),
    )
), $_REQUEST["auth"]);
```

Note: the restCommand function is a method for sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

The keyboard is a set of buttons, each button can consist of the following keys:

- **TEXT** - button text;
- **LINK** - link;
- **COMMAND** - the command that will be sent to the bot;
- **COMMAND_PARAMS** - parameters for the command;
- **BG_COLOR** - button color;
- **BLOCK** - if you specify Y, then after clicking on one button, the entire keyboard will be blocked;
- **DISABLED** - if you specify Y, then this button will not be clickable;
- **TEXT_COLOR** - text color;
- **DISPLAY** - button type. If the **BLOCK type is specified,** then only this button can be on one line. If the **LINE type is specified,** then the buttons will be lined up one after the other;

- **WIDTH** - button width. ***Please note that for maximum convenience, a set of buttons in one line is not recommended to be more than 225 pixels, this is the maximum width on a mobile device.***
- **APP_ID** - ID of the installed chat application.
- **APP_PARAMS** - parameters for launching the chat application.
- **ACTION** - action, can be one of the following types (REST revisions 28):
  - **PUT** - insert into the input field.
  - **SEND** - send text.
  - **COPY** - copy text to clipboard.
  - **CALL** - call.
  - **DIALOG** - open the specified dialog.
- **ACTION_VALUE** - value, for each type means its own (REST revisions 28):
  - **PUT** - text to be inserted into the input field.
  - **SEND** - text to be sent.
  - **COPY** - text to be copied to the clipboard.
  - **CALL** - phone number in international format.
  - **DIALOG** - dialog identifier, it can be a user ID, or a chat ID in the chatXXX format.

The required fields are **TEXT** and either the **LINK** field or the **COMMAND field.**

If the **LINK key is specified,** then the button becomes an external link. If the **COMMAND** and **COMMAND_PARAMS fields are specified,** then the button is an action and sends a command to the chatbot without publishing it to the chat.

If **the APP_ID** and **APP_PARAMS fields are specified,** the button will open a window with chat application.

If you need to make two rows with buttons in a row, then to separate them you need to add a button with the following content: "TYPE" => "NEWLINE".

Processing commands by the chatbot

Commands are used to process keystrokes on the keyboard .

1. In order for the command to work on the keyboard (and not only), you must first register it via the imbot.command.register method (in order for the command to be available only for the keyboard, you must create it with the key "HIDDEN"

   => "AND").

   The button contains the following keys:

```
"COMMAND" => "page", // the command that will be sent to the chatbot
"COMMAND_PARAMS" => "1", // parameters for the command
```

2. Pressing the button will generate an event ONIMCOMMANDADD.
3. Inside this event, you need to either create a new message or edit the old one (thus creating the pagination effect).
4. Inside the event, in the array **[data][COMMAND]** will be passed triggered event data. It has the **COMMAND_CONTEXT** value - this is a special key that describes the context in which the command was called:
   - if the command was written by the user himself, there will be **TEXTAREA;**
   - if the command came from the keyboard, then it will be **KEYBOARD;**
   - if the command came from the context menu, then it will be **MENU.**

Ready example, you can see in the updated version of EchoBot **(bot.php).**

Handling the opening of the chat application

The chat application launched from the context menu works according to the principles of the Context application.

Keyboard usage examples

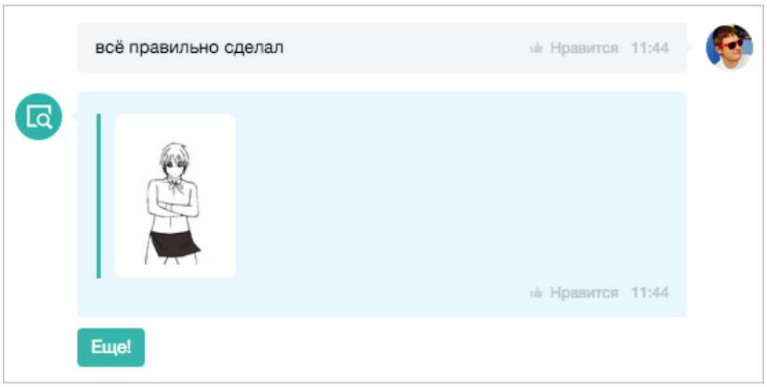
1. **EchoBot**
   Page navigation, buttons when calling the "Help" command

   https://youtu.be/2v5MUeVSBX4

2. **March**
   Just write March Play with me!. The keyboard is used as a playing field:

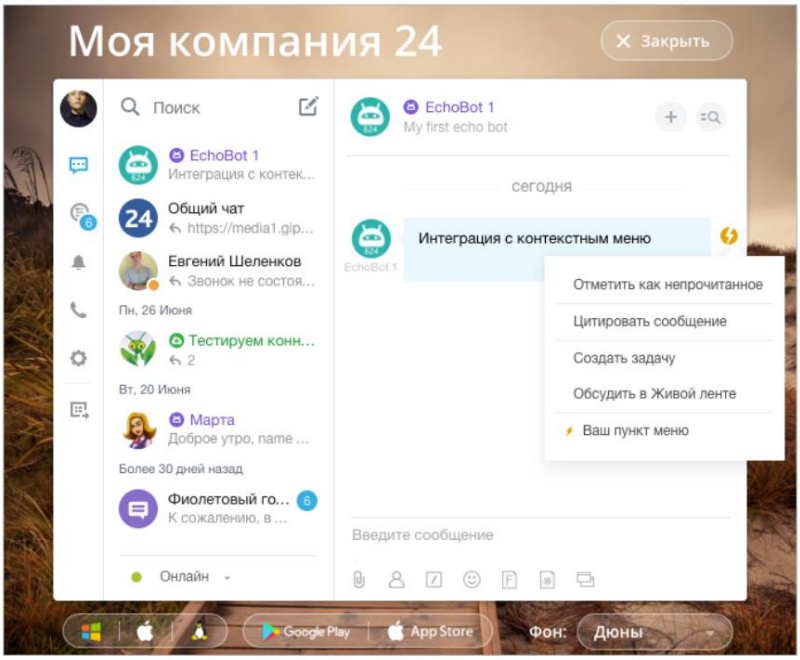
   https://youtu.be/qSDKsDwJsBI

3. **Giphy**
   The **More** button allows you to view other pictures on the same topic without re-entering the search word:



Working with the context menu

The context menu will allow the user to interact with the chat bot or chat application from the context menu of the message.



How to add custom items to the context menu

The context menu is part of the message, when creating the message, you need to add the MENU key and pass the parameters.

Methods that support context menu:

- imbot.message.add - send a message from the chatbot.
- imbot.message.update - Sending a change message to the chatbot.
- imbot.command.answer - publication of the response to the command.
- im.message.add - send a message to the chat.
- im.message.update - Sending a change message to the chatbot.

Let's take this message as an example:

```
restCommand('im.message.add', Array(
  "DIALOG_ID" => 12,
  "MESSAGE" => "Hello! Message with context menu!",
  "MENU" => Array(
    Array(
      "TEXT" => "Bitrix24",
      "LINK" => "http:// bitrix24.com",
    ),
```

```
        Array(
            "TEXT" => "Echo",
            "COMMAND" => "echo",
            "COMMAND_PARAMS" => "test from keyboard"
        ),
        Array(
            "TEXT" => "Open app",
            "APP_ID" => "12",
            "APP_PARAMS" => "TEST"
        ),
    )
), $_REQUEST["auth"]);
```

Note: the restCommand function is a method for sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

The context menu is a set of buttons, each button can consist of from the following keys:

- **TEXT** - button text;
- **LINK** - link;
- **COMMAND** - the command that will be sent to the bot;
- **COMMAND_PARAMS** - parameters for the command;
- **APP_ID** - ID of the installed chat application.
- **APP_PARAMS** - parameters for launching the chat application.
- **DISABLED** - if you specify Y, then this button will not be clickable.
- **ACTION** - action, can be one of the following types (REST revisions 28):
  - **PUT** - insert into the input field.
  - **SEND** - send text.
  - **COPY** - copy text to clipboard.

- **CALL** - call.
- **DIALOG** - open the specified dialog.

- **ACTION_VALUE** - value, for each type means its own (REST revisions 28):
  - **PUT** - text to be inserted into the input field.
  - **SEND** - text to be sent.
  - **COPY** - text to be copied to the clipboard.
  - **CALL** - phone number in international format.
  - **DIALOG** - dialog identifier, it can be a user ID, or a chat ID in the chatXXX format.

The required fields are **TEXT** and either the **LINK** field or the **COMMAND field.**

If the **LINK key is specified,** then the button becomes an external link. If the **COMMAND** and **COMMAND_PARAMS fields are specified,** then the button is an action and sends a command to the chatbot without publishing it to the chat. Available **from API revision 26** (platform version).

If **the APP_ID** and **APP_PARAMS fields are specified,** the button will open a window with chat application.

If you need to make two rows with buttons in a row, then to separate them you need to add a button with the following content: "TYPE" => "NEWLINE".

Processing commands by the chatbot

Commands are used to handle pressing of keyboard buttons and menu items .

1. In order for the command to work on the keyboard (and not only), you must first register it via the imbot.command.register method (in order for the command to be available only for the keyboard, you must create it with the key "HIDDEN" => "AND").

The button contains the following keys:

```
"COMMAND" => "page", // the command that will be sent to the chatbot
```

```
"COMMAND_PARAMS" => "1", // parameters for the command
```

2. Pressing the button will generate an ONIMCOMMANDADD event.

3. Inside this event, you need to either create a new message or edit the old one (thus creating the pagination effect).

4. Inside the event, in the array **[data][COMMAND]** will be passed triggered event data. It has the **COMMAND_CONTEXT** value - this is a special key that describes the context in which the command was called:
   - if the command was written by the user himself, there will be **TEXTAREA;**
   - if the command came from the keyboard, then it will be **KEYBOARD;**
   - if the command came from the context menu, then it will be **MENU.**

Handling the opening of the chat application

The chat application launched from the context menu works according to the principles of the Context application.

Bot API

To work with **the Bot API,** when creating an application (or uploading a new version), you need to specify scope: **imbot** (Creating and managing chat bots).

Working with chatbots

Important Notes

Note: the restCommand function is a method for sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

- imbot.register - chatbot registration.
- imbot.unregister - removal of the chatbot from the system.
- imbot.update - updating chatbot data.

- imbot.bot.list - receiving available chatbots.

Attention! For the bots to work, access to marta.bitrix.info from the portal or website must be provided.

Attention! *To work with chat bots through Webhooks you need to pass the CLIENT_ID parameter to all methods , this code must be unique for each chatbot.*

Chatbot registration

Rest-method: imbot.register

Method call:

```
$result = restCommand('imbot.register', Array(

    'CODE' => 'newbot', // String identifier for the bot, unique within your application (required)

    'TYPE' => 'H', // Bot type, B - chat bot, answers come immediately, H - human, answers come with a delay of 2 to 10 seconds, O - chat bot for Open lines, S - Chatbot with elevated privileges (supervisor)

    'EVENT_HANDLER' => 'http://www.hazz/chatApi/event.php', // Handler link events received from the server, see Event Handlers below (required).

    'OPENLINE' => 'Y', // Enabling the Open Line support mode, can be omitted if TYPE = 'O'

    'CLIENT_ID' => '', // chatbot string identifier, used only in Webhook mode

    'PROPERTIES' => Array( // Personal data of the chatbot (required)

        'NAME' => 'NewBot', // Name of the chatbot (required one of the NAME or LAST_NAME)

        'LAST_NAME' => '', // Last name of the chatbot (mandatory one of the NAME fields or LAST_NAME)

        'COLOR' => 'GREEN', // Chatbot color for mobile app RED, GREEN, MINT, LIGHT_BLUE, DARK_BLUE, PURPLE, AQUA, PINK, LIME, BROWN, AZURE, KHAKI, SAND, MARENGO, GRAY, GRAPHITE

        'EMAIL' => 'test@test.ru', // E-mail for communication. DO NOT use an e-mail that duplicates the e-mail of real users

        'PERSONAL_BIRTHDAY' => '2016-03-11', // Birthday in YYYY-mm-dd format

        'WORK_POSITION' => 'My first bot', // Position held, used as description of the chatbot
```

```
        'PERSONAL_WWW' => 'http://test.ru', // Link to site
        'PERSONAL_GENDER' => 'F', // Chatbot gender, valid values M - male, F -
female, empty if not required
        'PERSONAL_PHOTO' => '/* base64 image */', // Chatbot avatar - base64
    )

), $_REQUEST["auth"]);
```

Required fields:

CODE - string identifier of the bot, unique within your applications.

PROPERTIES - array with personal data of the chatbot.

EVENT_HANDLER - link to the event handler for sending a message to the chat bot.

or

EVENT_MESSAGE_ADD - link to the event handler for sending a message to the chat bot.

EVENT_WELCOME_MESSAGE - a link to the event handler for opening a dialog with a chatbot or inviting it to a group chat.

EVENT_BOT_DELETE - a link to the event handler for deleting a chatbot from client side.

Execution result: numeric identifier of the created bot BOT_ID or an error.

Event handlers: If you need to handle events using different handlers, then instead of EVENT_HANDLER you can specify each handler individually:

```
    'EVENT_MESSAGE_ADD' => 'http://www.hazz/chatApi/event.php', // Link to handler
events of sending a message to the chatbot
    'EVENT_WELCOME_MESSAGE' => 'http://www.hazz/chatApi/event.php', // Link to handler
events of opening a dialogue with a chatbot or inviting it to a group chat
    'EVENT_BOT_DELETE' => 'http://www.hazz/chatApi/event.php', // Link to client-side
chatbot delete event handler
    'EVENT_MESSAGE_UPDATE' => 'http://www.hazz/chatApi/event.php', // Link to handler
change event subscription events
```

```
'EVENT_MESSAGE_DELETE' => 'http:// www.hazz/ chatApi/ event.php', // Link to the event handler for subscribing to message deletion events
```

Related links:

Rest API - Install and Update Events

Possible mistakes:

Error code	Error Description
EVENT_MESSAGE_ADD_ERROR	The event handler reference is invalid or not indicated.
EVENT_WELCOME_MESSAGE_ERROR	Link Handler event invalid or not indicated.
EVENT_BOT_DELETE_ERROR	The event handler reference is invalid or not indicated.
CODE_ERROR	The string identifier of the chatbot was not specified.
NAME_ERROR	One of the required fields **NAME** or **LAST_NAME** of the chatbot.
WRONG_REQUEST	Something went wrong.
MAX_COUNT_ERROR	Achieved maximum quantity

registered bots for one

applications.

A chatbot with elevated privileges (supervisor) is a bot that gets access to all messages in the chats where it is a member (all if it was invited with access to history, and new ones if access to history is not available to it).

To create a chatbot with elevated privileges in the method **imbot.register** in the TYPE field specify type S.

Please note *if this is not required by the logic of your application, it is recommended that the bot respond to user messages only when this chat bot is mentioned. You can check this by the TO_USER_ID field, which will be passed to the event.*

Attention! If you plan to install more than one chatbot in within one application: Bitrix24 Rest imposes a restriction on working with event handlers - there can be only one handler per application. Therefore, when registering the second chatbot, the links to the EVENT_MESSAGE_ADD, EVENT_WELCOME_MESSAGE and EVENT_BOT_DELETE handlers must be the same as those of the first bot.

If you need to process several bots within one application, then you need to provide this inside the event handler. To do this, when an event arrives, an array of bots is transmitted so that they can be processed correctly.

Maximum number of bots for one application: 5.

Removing a chatbot from the system

Attention! All one-on-one chats of this chatbot with users will be lost.

Rest-method: imbot.unregister

Method call:

```
$result = restCommand('imbot.unregister', Array(

    'BOT_ID' => 39 // numeric id of the bot
    'CLIENT_ID' => '', // chatbot string identifier, used only in Webhook mode


), $_REQUEST["auth"]);
```

Execution result: true or error.

Related links:

Rest API - Install and Update Events

Possible mistakes:

Error code	Error Description
BOT_ID_ERROR	Chatbot not found. The chatbot does not belong to this application, **APP_ID_ERROR** can only work with chatbots installed within the application.

Update bot data

Remainder: imbot.update

Method call:

```
$result = restCommand('imbot.update', Array(

    'BOT_ID' => 39, // ID of the chatbot to change (required)
```

```
        'CLIENT_ID' => '', // chatbot string identifier, used only in Webhook mode

        'FIELDS' => Array( // Data to update (required)
            'CODE' => 'newbot', // String identifier of the chatbot, unique within the framework
your application
            'EVENT_HANDLER' => 'http://www.hazz/chatApi/event.php', // Handler link
events received from the server, see Event Handlers below (required).
            'PROPERTIES' => Array( // Mandatory when updating bot data
                'NAME' => 'UpdatedBot', // Chatbot name
                'LAST_NAME' => '', // Last name of the chatbot
                'COLOR' => 'MINT', // Chatbot color for mobile app RED, GREEN, MINT,
LIGHT_BLUE, DARK_BLUE, PURPLE, AQUA, PINK, LIME, BROWN, AZURE, KHAKI, SAND, MARENGO,
GRAY, GRAPHITE
                'EMAIL' => 'test2@test.ru', // Contact email
                'PERSONAL_BIRTHDAY' => '2016-03-12', // Birthday in YYYY-mm-dd format
                'WORK_POSITION' => 'My second bot', // Position held, used as chatbot description

                'PERSONAL_WWW' => 'http://test2.ru', // Link to website
                'PERSONAL_GENDER' => 'M', // Bot gender, valid values M - male, F -
 feminine, empty if not required
                'PERSONAL_PHOTO' => '/* base64 image */', // Chatbot avatar - base64
            )
        )

    ), $_REQUEST["auth"]);
```

Required fields:

BOT_ID - ID of the chatbot to be changed.

FIELDS - an array with data to update.

PROPERTIES - array with personal data of the chatbot.

EVENT_HANDLER - link to the event handler for sending a message to the chat bot.

or

EVENT_MESSAGE_ADD - link to the event handler for sending a message to the chat bot.

EVENT_WELCOME_MESSAGE - a link to the event handler for opening a dialog with a chatbot or inviting it to a group chat.

EVENT_BOT_DELETE - a link to the event handler for deleting a chatbot from

client side.

Execution result: true or error.

Event handlers: If you
need to handle events using different handlers, then instead of
EVENT_HANDLER you can specify each handler individually:

```
'EVENT_MESSAGE_ADD' => 'http://www.hazz/chatApi/event.php', // Link to handler
events of sending a message to the chatbot
'EVENT_WELCOME_MESSAGE' => 'http://www.hazz/chatApi/event.php', // Link to handler
events of opening a dialogue with a chatbot or inviting it to a group chat
'EVENT_BOT_DELETE' => 'http://www.hazz/chatApi/event.php', // Link to client-side chatbot delete event
handler
'EVENT_MESSAGE_UPDATE' => 'http://www.hazz/chatApi/event.php', // Link to handler
change event subscription events
'EVENT_MESSAGE_DELETE' => 'http://www.hazz/chatApi/event.php', // Link to the event handler for subscribing to
message deletion events
```

Related links:

Rest API - Install and Update Events

Possible mistakes:

Error code	Error Description
BOT_ID_ERROR	Chatbot not found.
APP_ID_ERROR	Chatbot is not belongs to this application, work only possible with chat bots installed within the application.

EVENT_MESSAGE_ADD_ERROR	The event handler reference is invalid or not indicated.
EVENT_WELCOME_MESSAGE_ERROR	Link Handler event is invalid or not specified.
EVENT_BOT_DELETE_ERROR	The event handler reference is invalid or not indicated.
NAME_ERROR	One of the required fields **NAME** or **LAST_NAME** of the chatbot.
WRONG_REQUEST	Something went wrong.

The ***imbot.update*** *method no longer supports changing the TYPE and OPENLINE* fields *(im 17.5.10).*

Getting available chatbots

Remainder: imbot.bot.list

Method call:

```
$result = restCommand('imbot.bot.list', Array( ),
$_REQUEST["auth"]);
```

Execution result: an array of arrays containing data on chat bots:

- **ID** - bot identifier.

- **NAME** - the name of the chatbot.
- **CODE** - internal code.
- **OPENLINE** - whether or not open lines are supported.

Working with chats

Note: the restCommand function is a method for sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

- imbot.chat.user.list - the list of participants of the receipt.
- imbot.chat.user.add - invitation to participants.
- imbot.chat.user.delete - exception participants.
- imbot.chat.leave - exit the chatbot from the specified chat.
- imbot.chat.updateTitle - update the title of the chat.
- imbot.chat.updateColor - update the color of the chat.
- imbot.chat.updateAvatar - updated chat avatar.
- imbot.chat.add - creating a chat on behalf of a chat bot.
- imbot.chat.get - Get the chat ID.
- imbot.chat.setOwner - change the owner of the chat on behalf of the chat bot.
- imbot.dialog.get - getting information about the dialog

Getting a list of participants

Rest-Method: imbot.chat.user.list

Method call:

```
$result = restCommand('imbot.chat.user.list', Array(

    'CHAT_ID' => 13 // Chat ID

    'BOT_ID' => 39, // ID of the chatbot from which the request is coming. Can not
indicate if there is only one chatbot
```

```
), $_REQUEST["auth"]);
```

Execution result: an array of participant IDs or an error.

Possible mistakes:

Error code	Error Description
CHAT_ID_EMPTY	No chat ID passed.

Invitation of participants

Remainder: imbot.chat.user.add

Method call:

```
$result = restCommand('imbot.chat.user.add', Array(

    'CHAT_ID' => 13 // Chat ID 'USERS' =>
    Array(3,4) // IDs of new users 'BOT_ID' => 39, // ID of the chatbot from
    which the request comes. Can not
indicate if there is only one chatbot

), $_REQUEST["auth"]);
```

Execution result: true or error.

Possible mistakes:

Error code	Error Description

CHAT_ID_EMPTY	Chat ID not passed.
WRONG_REQUEST	You do not have sufficient rights to add member to the chat or the member is already a member chat.

Exclusion of members

Remainder: imbot.chat.user.delete

Method call:

```
$result = restCommand('imbot.chat.user.delete', Array(

    'CHAT_ID' => 13 // Chat ID
    'USER_ID' => 4 // ID of the excluded user
    'BOT_ID' => 39, // ID of the chatbot from which the request is coming. Can not indicate if there is only one chatbot

), $_REQUEST["auth"]);
```

Execution result: true or error.

Possible mistakes:

Error code	Error Description
CHAT_ID_EMPTY	Chat ID not passed.
USER_ID_EMPTY	Identifier not passed user.
WRONG_REQUEST	You don't have enough rights to add a participant to a chat or a participant

is already in the chat.

Exit the chatbot from the specified chat

Remainder: imbot.chat.leave

Method call:

```
$result = restCommand('imbot.chat.leave', Array(

    'CHAT_ID' => 13 // ID of the chat to exit

    'BOT_ID' => 39, // ID of the chatbot from which the request is coming. Can not indicate if there is only one chatbot

), $_REQUEST["auth"]);
```

Execution result: true or error.

Possible mistakes:

Error code	Error Description
CHAT_ID_EMPTY	No chat ID passed.
ACCESS_ERROR	You do not have sufficient rights to submit the status to this chat.
BOT_ID_ERROR	Chatbot not found.

Chat title update

Rest-method: imbot.chat.updateTitle

Method call:

```
$result = restCommand('imbot.chat.updateTitle', Array(

    'CHAT_ID' => 13 // Chat ID
    'TITLE' => 'New name for the chat' // New title
    'BOT_ID' => 39, // ID of the chatbot from which the request is coming. Can not indicate if there is only one chatbot

), $_REQUEST["auth"]);
```

Execution result: true or error.

Possible mistakes:

Error code	Error Description
CHAT_ID_EMPTY	Chat ID not passed.
TITLE_EMPTY	No new chat title passed.
WRONG_REQUEST	Title already set or specified chat does not exist.

Chat color update

Remainder: imbot.chat.updateColor

Method call:

```
$result = restCommand('imbot.chat.updateColor', Array(
```

```
    'CHAT_ID' => 13 // Chat ID
    'COLOR' => 'MINT' // Mobile app chat color - RED, GREEN, MINT,
LIGHT_BLUE, DARK_BLUE, PURPLE, AQUA, PINK, LIME, BROWN, AZURE, KHAKI, SAND, MARENGO,
GRAY, GRAPHITE
    'BOT_ID' => 39, // ID of the chatbot from which the request is coming. Can not
indicate if there is only one chatbot

), $_REQUEST["auth"]);
```

Execution result: true or error.

Possible mistakes:

Error code	Error Description
CHAT_ID_EMPTY	Chat ID not passed.
WRONG_COLOR	The color is not included in the list of available colors.
WRONG_REQUEST	The color is already set or the specified chat does not exist.

Chat avatar update

Rest-method: imbot.chat.updateAvatar

Method call:

```
$result = restCommand('imbot.chat.updateAvatar', Array(

    'CHAT_ID' => 13 // chat ID
    'AVATAR' => '/* base64 image */' // Base64 image
    'BOT_ID' => 39, // Identifier of the chatbot from which the request comes, you can not
indicate if there is only one chatbot
```

```
), $_REQUEST["auth"]);
```

Execution result: true or error.

Possible mistakes:

Error code	Error Description
CHAT_ID_EMPTY	Chat ID not passed.
AVATAR_ERROR	Incorrect image format transferred.
WRONG_REQUEST	Chat does not exist.

Creating a chat on behalf of a chatbot

Remainder: imbot.chat.add

Method call:

```
$result = restCommand('imbot.chat.add', Array(

    'TYPE' => 'CHAT' // OPEN - chat open for joining, CHAT - regular chat by prompts, default is CHAT
    'TITLE' => 'My new private chat', // Chat title
    'DESCRIPTION' => 'Very important chat', // Chat description
    'COLOR' => 'PINK', // Mobile app chat color - RED, GREEN, MINT, LIGHT_BLUE, DARK_BLUE, PURPLE, AQUA, PINK, LIME, BROWN, AZURE, KHAKI, SAND, MARENGO, GRAY, GRAPHITE
    'MESSAGE' => 'Welcome to the chat', // The first welcome message in the chat
    'USERS' => Array(1,2), // Chat members (required)
    'AVATAR' => '/* base64 image */', // Avatar chat in base64 format
    'ENTITY_TYPE' => 'CHAT', // Identifier of an arbitrary entity (for example, CHAT, CRM, OPENLINES, CALL, etc.) can be used to search for a chat and to easily identify
```

```
context in the event handlers ONIMBOTMESSAGEADD, ONIMBOTMESSAGEUPDATE,
ONIMBOTMESSAGEDELETE

        'ENTITY_ID' => 13, // Numeric entity ID, can be used to
chat search and for easy definition of context in event handlers
ONIMBOTMESSAGEADD, ONIMBOTMESSAGEUPDATE, ONIMBOTMESSAGEDELETE

        'OWNER_ID' => 39, // Chat owner ID. You may not indicate if you
create a chat under the desired user

        'BOT_ID' => 39, // ID of the chatbot from which the request is coming. Can not
indicate if there is only one chatbot


    ), $_REQUEST["auth"]);
```

Required fields: USERS (chat participants).

Execution result: numeric chat ID CHAT_ID or error.

Possible mistakes:

Error code	Error Description
USERS_EMPTY	Chat members not transferred.
WRONG_REQUEST	Something went wrong.

Getting an ID chat

Rest-Method: imbot.chat.get

Method call:

```
    $result = restCommand('imbot.chat.get', Array(


        'ENTITY_TYPE' => 'OPEN', // Identifier of an arbitrary entity (for example, CHAT, CRM,
OPENLINES, CALL, etc.) can be used to search for a chat and to easily identify
```

```
context in ONIMBOTMESSAGEADD, ONIMBOTMESSAGEUPDATE,
ONIMBOTMESSAGEDELETE event handlers (required)
    'ENTITY_ID' => 13, // Numeric entity ID, can be used to
chat search and for easy context detection in ONIMBOTMESSAGEADD,
ONIMBOTMESSAGEUPDATE, ONIMBOTMESSAGEDELETE event handlers (required)
    'BOT_ID' => 39, // ID of the chatbot from which the request is coming. Can not
indicate if there is only one chatbot

), $_REQUEST["auth"]);
```

Required fields:

ENTITY_TYPE - identifier of an arbitrary entity (for example, CHAT, CRM, OPENLINES, CALL, etc.), can be used to search for a chat and to easily determine the context in the ONIMBOTMESSAGEADD, ONIMBOTMESSAGEUPDATE, ONIMBOTMESSAGEDELETE event handlers.

ENTITY_ID - numeric identifier of the entity, can be used to search for a chat and to easily determine the context in the event handlers ONIMBOTMESSAGEADD, ONIMBOTMESSAGEUPDATE, ONIMBOTMESSAGEDELETE.

Execution result: returns chat ID CHAT_ID or null.

Changing the owner of a chat on behalf of a chatbot

Rest-method: imbot.chat.setOwner

Method call:

```
$result = restCommand('imbot.chat.setOwner', Array(

    'CHAT_ID' => 13 // Chat ID
    'USER_ID' => '2' // ID of the new chat owner
    'BOT_ID' => 39, // ID of the chatbot from which the request is coming. Can not
indicate if there is only one chatbot

), $_REQUEST["auth"]);
```

Execution result: true or error.

Possible mistakes:

Error code	Error Description
CHAT_ID_EMPTY	Chat ID not passed.
USER_ID_EMPTY	New identifier not passed chat owner.
WRONG_REQUEST	The method was not called by the chat owner or the specified user is not chat member.

imbot.dialog.get

Getting information about a dialog

Revision: 24

Options

Parameter	Example	Required	Description	Revision
DIALOG_ID	chat29 or 256	**Yes**	Identifier dialogue. Format: **chatXXX** -- chat recipient, if chat message or **XXX** -- identifier recipient, if	24

			message for private conversation	

Method call

JavaScript

```
BX24.callMethod('imbot.dialog.get', {
    DIALOG_ID: 'chat29'
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('imbot.dialog.get', Array(
    'DIALOG_ID': 'chat29'
), $_REQUEST["auth"]);
```

Response Example


```
{
    "result":
  {
    "id": "21191",
    "title": "Mint Chat #3",
    "owner": "2",
    "extranet": false,
    "avatar": "",
    "color": "#4ba984",
```

```
    "type": "chat",
    "entity_type": "",
    "entity_data_1": "",
    "entity_data_2": "",
    "entity_data_3": "",
    "date_create": "2017-10-14T12:15:32+02:00",
    "message_type": "C"
  }
}
```


Description of keys:

- id -- chat identifier
- title -- chat name
- owner -- user ID of the chat owner
- extranet -- sign that an external extranet user is participating in the chat (true/false)
- color -- chat color in hex format
- avatar -- link to the avatar (if empty, no avatar has been set)
- type -- chat type (group chat, call chat, open line chat, etc.)
- entity_type -- external code for chat -- type
- entity_id -- external code for chat -- identifier
- entity_data_1 -- external data for chat
- entity_data_2 -- external data for chat
- entity_data_3 -- external data for chat
- date_create -- chat creation date in ATOM format
- message_type -- chat message type

An example of a response when an error occurs


```
{
    "error": "DIALOG_ID_EMPTY",
```

```
    "error_description": "Dialog ID can't be empty"
}
```


Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
DIALOG_ID_EMPTY	Dialog ID not passed
ACCESS_ERROR	The current user does not have access rights to the dialog

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Working with messages

Important Notes

Note! All methods are specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

- imbot.message.add - send a message from the chatbot.
- imbot.message.update - Sending a change message to the chatbot.

- imbot.message.delete - delete message chat-bot.
- imbot.message.like - "I like" setting.
- imbot.chat.sendtyping - sending a message "Chatbot is writing a message..."

Note! These messaging methods are usually are applied when the user writes something, so the ONIMBOTMESSAGEADD event must be handled.

Sending a message from a chatbot

Remainder: imbot.message.add

Method call:

```
$result = restCommand('imbot.message.add', Array(

    'BOT_ID' => 39, // Identifier of the chatbot from which the request comes, you can not indicate if there is only one chatbot
    'DIALOG_ID' => 1, // Dialog ID, this is either the user's USER_ID or chatXX - where XX is the chat ID, passed in the ONIMBOTMESSAGEADD event and ONIMJOINCHAT
    'MESSAGE' => 'answer text', // Test message
    'ATTACH' => '', // Attachment, optional field
    'KEYBOARD' => '', // Keyboard, optional field
    'MENU' => '', // Context menu, optional field
    'SYSTEM' => 'N', // Display messages as a system message, optional field, default 'N'
    'URL_PREVIEW' => 'Y' // Convert links to rich links, optional field, default 'Y'

), $_REQUEST["auth"]);
```

You can post a message on behalf of the bot to private dialogs (will be shown as a system message). For this, instead of

DIALOG_ID ÿÿÿÿÿÿÿÿ USER_FROM_ID ÿ USER_TO_ID.

Execution result: message identifier MESSAGE_ID or error.

Related links:

How to work with typing keyboards How to work with attachments Formatting a message Event for the chatbot to receive a message ONIMBOTMESSAGEADD

Possible mistakes:

Error code	Error Description
BOT_ID_ERROR	Chatbot not found.
APP_ID_ERROR	The chatbot does not belong to this application, you can only work with chatbots installed within the application.
DIALOG_ID_EMPTY	Dialog ID not passed.
MESSAGE_EMPTY	Message text not sent.
ATTACH_ERROR	The entire attachment object passed is not has been validated.
ATTACH_OVERSIZE	The maximum allowed attachment size (30 KB) has been exceeded.
KEYBOARD_ERROR	The entire passed keyboard object did not pass validation.

KEYBOARD_OVERSIZE Maximum OVERSIZE exceeded.

	keyboard size (30 Kb).
MENU_ERROR	The entire passed menu object is not has been validated.
MENU_OVERSIZE	The maximum allowable menu size (30 KB) has been exceeded.
WRONG_REQUEST	Something went wrong.

Sending a chatbot message change

Remainder: imbot.message.update

Method call:

```
$result = restCommand('imbot.message.update', Array(

    'BOT_ID' => 39, // Identifier of the chatbot from which the request comes, you can not indicate if there is only one bot
    'MESSAGE_ID' => 1, // Message ID
    'MESSAGE' => 'answer text', // Message text, optional field if passed empty value - the message will be deleted
    'ATTACH' => '', // Attachment, optional field
    'KEYBOARD' => '', // Keyboard, optional field
    'MENU' => '', // Context menu, optional field
    'URL_PREVIEW' => 'Y' // Convert links to rich links, optional field

), $_REQUEST["auth"]);
```

Execution result: true or error.

Related links:

How to work with typed keyboards
How to work with attachments
Formatting a message

Event for the chatbot to receive the message ONIMBOTMESSAGEADD

Possible mistakes:

Error code	Error Description
BOT_ID_ERROR	Chatbot not found.
APP_ID_ERROR	The chatbot does not belong to this application, you can only work with chatbots installed within the application.
MESSAGE_EMPTY	Message text not sent.
CANT_EDIT_MESSAGE	You do not have access to this post or the time for its modification has expired (more than 3 days have passed since the publication).
ATTACH_ERROR	The entire attachment object passed is not has been validated.
ATTACH_OVERSIZE	The maximum allowed attachment size (30 KB) has been exceeded.
KEYBOARD_ERROR	The entire passed keyboard object failed validation.
KEYBOARD_OVERSIZE	The maximum allowed keyboard size (30 KB) has been exceeded.
MENU_ERROR	The entire passed menu object is not has been validated.
MENU_OVERSIZE	Exceeded the maximum allowable

menu size (30 Kb).

Deleting a chatbot message

Remainder: imbot.message.delete

Method call:

```
$result = restCommand('imbot.message.delete', Array(

    'BOT_ID' => 39, // Identifier of the chatbot from which the request comes, you can not indicate if there is only one bot
    'MESSAGE_ID' => 1, // Message ID
    'COMPLETE' => 'N', // If the message needs to be deleted completely, without traces, then you must specify Y (optional parameter)

), $_REQUEST["auth"]);
```

Execution result: true or error.

Possible mistakes:

Error code	Error Description
BOT_ID_ERROR	Chatbot not found.
APP_ID_ERROR	The chatbot does not belong to this application, you can only work with chatbots installed within the application.

MESSAGE_ID_ERROR No message ID passed.

CANT_EDIT_MESSAGE You do not have access to this message.

"Like" setting

Rest method: imbot.message.like

Method call:

```
$result = restCommand('imbot.message.like', Array(

    'BOT_ID' => 39, // Identifier of the chatbot from which the request comes, you can not indicate if there is only one chatbot
    'MESSAGE_ID' => 1, // Message ID (any message sent in private conversations or group chats where the chatbot is present)
    'ACTION' => 'auto' // Action associated with the method: plus - will put a "Like" label; minus - will remove the "I like" label; auto - automatically calculates whether to mark or unmark. If not specified, it will work in auto mode

), $_REQUEST["auth"]);
```

Execution result: true or error.

Possible mistakes:

Error code	Error Description
BOT_ID_ERROR	Chatbot not found.
APP_ID_ERROR	The chatbot does not belong to this application, you can only work with chatbots installed within the application.

MESSAGE_ID_ERROR No message ID passed.

WITHOUT_CHANGES After calling the status state "I

like" has not changed.

Sending a message "Chatbot is writing a message..."

Rest-method: imbot.chat.sendTyping

Method call:

```
$result = restCommand('imbot.chat.sendTyping', Array(

    'BOT_ID' => 39, // Identifier of the chatbot from which the request comes, you can not indicate if there is only one bot
    'DIALOG_ID' => 1, // Dialog ID, this is either the user's USER_ID or chatXX - where XX is the chat ID, passed in the ONIMBOTMESSAGEADD event and ONIMJOINCHAT

), $_REQUEST["auth"]);
```

Execution result: true or error.

Related links:

Event for the chatbot to receive the message ONIMBOTMESSAGEADD
Event for the chatbot to receive information about its inclusion in the chat (or personal correspondence) ONIMJOINCHAT

Possible mistakes:

Error code	Error Description
BOT_ID_ERROR	Chatbot not found.
DIALOG_ID_EMPTY	Not transferred Dialog ID.

Teams

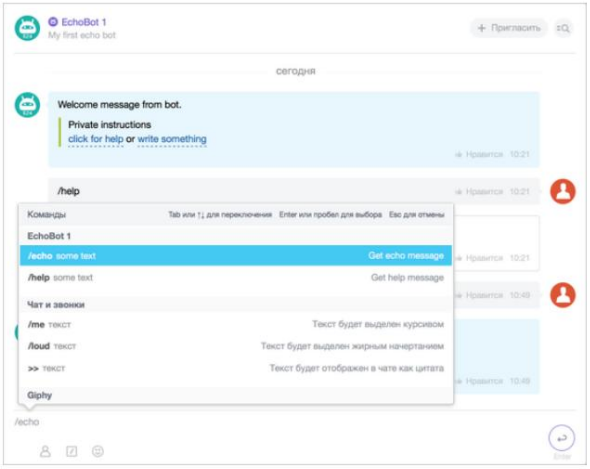
Commands are **general** and **local:**

General Commands

- **General commands** work in any dialog or any chat. An example of such a command (COMMON = Y):

```
/giphy image
```

Calling such a command in any chat will generate a response from the chatbot, even in a chat where the chatbot is not a member.

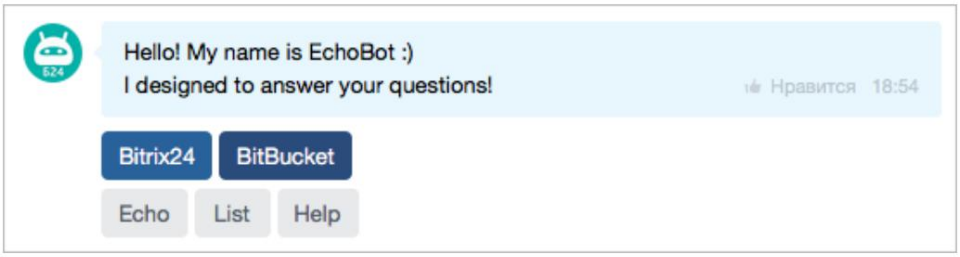


Local commands

- **Local commands** work only in personal correspondence with the chat bot and in group chats where it is in the participants. An example of such a command (COMMON = N):

```
/help
```

Calling such a command in EchoBot **(bot.php)** will list available commands.



Working with teams

Important Notes

Note! All methods are specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

- imbot.command.register - registering a command for processing by the chatbot.
- imbot.command.unregister - deletion of command processing.
- imbot.command.update - updating data in the team.
- imbot.command.answer - publication of the response to the command.

Note! To process a command, you need to application was handling the event of adding the ONIMCOMMANDADD command.

Registering a team for processing by a chatbot

Rest-Method: imbot.command.register

Method call:

```
$result = restCommand('imbot.command.register', Array(

  'BOT_ID' => 62, // Chatbot ID of the team owner
  'COMMAND' => 'echo', // The text of the command that the user will enter in chats
```

```
        'COMMON' => 'Y', // If Y is specified, then the command is available in all chats, if N, then it is
available only in those where the chatbot is present
        'HIDDEN' => 'N', // Hidden command or not - default is N
        'EXTRANET_SUPPORT' => 'N', // Is the command available to Extranet users, by
default N
        'CLIENT_ID' => '', // chatbot string identifier, used only in Webhook mode

        'LANG' => Array( // Array of translations, be sure to specify at least for RU and EN
          Array('LANGUAGE_ID' => 'in', 'TITLE' => 'Get echo message', 'PARAMS' => 'some text'), //
a description of the command, what data after the command you need to enter.
        ),
        'EVENT_COMMAND_ADD' => 'http://www.hazz/chatApi/bot.php', // Link to command handler

    ), $_REQUEST["auth"]);
```

Important! Be sure to specify an array of *LANG translations , at least for RU and EN. If there is no phrase for BY, UA, KZ, then the phrase RU is shown by default, if there is no phrase in RU, the command is hidden. For other languages, the same is true - if there are no phrases, then EN phrases are shown by default, if there is no phrase in EN, then the command is hidden in the public part.*

Execution result: command ID COMMAND_ID or error.

Related links:

Event for the chatbot to receive the ONIMCOMMANDADD command

Possible mistakes:

Error code	Error Description
EVENT_COMMAND_ADD	Link event handler invalid or not specified.

COMMAND_ERROR	The text of the command to which the chatbot should respond is not specified.
BOT_ID_ERROR	Chatbot not found.
APP_ID_ERROR	The chatbot does not belong to this application, you can only work with chatbots installed within the application.
NO ERROR	No language phrases were passed for the visible command.
WRONG_REQUEST	Something went wrong.

Attention! If you plan to install more than one command for chatbot: Bitrix24 Rest imposes a restriction on working with event handlers - there can be only one handler per application. Therefore, when registering the second command, the references to the EVENT_COMMAND_ADD handlers must be the same as for the first command.

If it is necessary to process several commands within one application, then it is necessary to provide this inside the event handler. To do this, when an event arrives, an array of commands is transmitted so that they can be processed correctly.

Deleting command processing

Rest-method: imbot.command.unregister

Method call:

```
$result = restCommand('imbot.command.unregister', Array(
```

```
    'COMMAND_ID' => 13, // Command ID to delete
    'CLIENT_ID' => '', // chatbot string identifier, used only in Webhook mode

), $_REQUEST["auth"]);
```

Execution result: true or error.

Possible mistakes:

Error code	Error Description
COMMAND_ID_ERROR	Command not found.
APP_ID_ERROR	The chatbot does not belong to this application, you can only work with chatbots installed within the application.
WRONG_REQUEST	Something went wrong.

Updating data in a team

Rest-Method: imbot.command.update

Method call:

```
$result = restCommand('imbot.command.update', Array(

  'COMMAND_ID' => 13, // Chat ID
  'FIELDS' => Array(
    'EVENT_COMMAND_ADD' => 'http://www.hazz/chatApi/bot.php', // Link to command handler

    'HIDDEN' => 'N', // Hidden command or not
    'EXTRANET_SUPPORT' => 'N', // Is the command available to Extranet users
```

```
        'CLIENT_ID' => '', // chatbot string identifier, used only in
webhook mode
        'LANG' => Array( // New translation phrases, all previous ones will be deleted
          Array('LANGUAGE_ID' => 'en', 'TITLE' => 'Get echo message', 'PARAMS' => 'some text'),
        ),
      )

  ), $_REQUEST["auth"]);
```

Required fields: team ID and one of the required fields for editing.

Important! Be sure to specify an array of *LANG translations , at least for RU and EN. If there is no phrase for BY, UA, KZ, then the phrase RU is shown by default, if there is no phrase in RU, the command is hidden. For other languages, the same is true - if there are no phrases, then EN phrases are shown by default, if there is no phrase in EN, then the command is hidden in the*

public part. **Execution result:** true or error.

Possible mistakes:

Error code	Error Description
COMMAND_ID_ERROR	Command not found.
APP_ID_ERROR	The chatbot does not belong to this application, you can only work with chatbots installed within the application.
EVENT_COMMAND_ADD	Link event handler invalid or not specified.
WRONG_REQUEST	Something went wrong.

Posting a response to a command

Rest-Method: imbot.command.answer

Method call:

```
$result = restCommand('imbot.command.answer', Array(

    'COMMAND_ID' => 13, // ID of the command that prepared the response
(required to specify or COMMAND)
    'COMMAND' => 'echo', // Name of the command that prepared the response (required
specify or COMMAND_ID)
    'MESSAGE_ID' => 1122, // ID of the message to send to
answer
    'MESSAGE' => 'answer text' // Response text
    'ATTACH' =>  '' // Attachment, optional field
    'KEYBOARD' => '' // Keyboard, optional field
    'MENU' => '' // Context menu, optional field
    'SYSTEM' => 'N' // Display messages as a system message, optional field, default 'N'
    'URL_PREVIEW' => 'Y' // Convert links to rich links, optional field, default 'Y'
    'CLIENT_ID' => '', // chatbot string identifier, used only in Webhook mode

), $_REQUEST["auth"]);
```

Execution result: command message ID MESSAGE_ID or error.

Related links:

How to work with typed keyboards How to work with attachments
Formatting a message

Possible mistakes:

Error code	Error Description
COMMAND_ID_ERROR	Command not found.
APP_ID_ERROR	The chatbot does not belong to this application, you can only work with chatbots installed within the application.
MESSAGE_EMPTY	Message text not sent.
ATTACH_ERROR	The entire attachment object passed did not pass validation.
ATTACH_OVERSIZE	The maximum allowed attachment size (30 KB) has been exceeded.
KEYBOARD_ERROR	The entire passed keyboard object did not pass validation.
KEYBOARD_OVERSIZE	The maximum allowed keyboard size (30 KB) has been exceeded.
MENU_ERROR	The entire passed menu object is not has been validated.
MENU_OVERSIZE	The maximum allowable menu size (30 KB) has been exceeded.
WRONG_REQUEST	Something went wrong.

Working with events

When an event is generated (adding a new message, opening a dialog with a chatbot, inviting a chatbot to a chat, deleting a chatbot from the client side), a request is generated for the event handler specified by you when registering the bot, you can read more about the event mechanism here .

- ONAPPINSTALL - application installation event.
- ONAPPUPDATE - application update event.
- ONIMBOTMESSAGEADD - event for receiving a message by the chatbot.
- ONIMBOTMESSAGEUPDATE - event for updating the message by the chatbot.
- ONIMBOTMESSAGEDELETE - event for deleting a message by the chatbot.
- ONIMCOMMANDADD - event to be received by the chatbot commands.
- ONIMBOTJOINCHAT - event for receiving information by the chat bot about its inclusion in the chat (or personal correspondence).
- ONIMBOTDELETE - event to delete the chatbot. An
- example of a handler code for working with events

Application installation event ONAPPINSTALL

```
[data] => Array(
  [LANGUAGE_ID] = ru // Base language for the portal
  [VERSION] = 1 // Applications version
)
[auth] => Array(
  [access_token] => lh8ze36o8ulgrljbyscr36c7ay5sinva // Key for sending REST requests service
  [scope] => imbot // Allowed access levels
  [domain] => b24.hazz // Bitrix24 portal domain where the application
```

[application_token] => c917d38f6bdb84e9d9e0bfe9d585be73 // The application token will help you "beat off" unnecessary requests to the event handler, this field is in all events

[expires_in] => 3600 // Token expiration time after which a new one will need to be requested

[member_id] => d41d8cd98f00b204e9800998ecf8427e // Unique portal ID, required to renew authorization

[refresh_token] => 5f1ih5tsnsb11sc5heg3kp4ywqnjhd09 // Key to renew authorization

)

Note! In the basic version of working with a chatbot, the expires_in, member_id, refresh_token fields are not required. But, if it is necessary for your application, then you can read how to work with them here. The example bot contains the option to renew.

Application update event ONAPPUPDATE

[data] => Array(

[LANGUAGE_ID] = ru // Base language for the portal

[VERSION] = 2 // New application version

[PREVIOUS_VERSION] => 1 // Old version apps

)

[auth] => Array(

[access_token] => lh8ze36o8ulgrljbyscr36c7ay5sinva // Key for sending REST requests service

```
    [scope] => imbot // Allowed access levels
    [domain] => b24.hazz // Bitrix24 portal domain where the
application
    [application_token] => c917d38f6bdb84e9d9e0bfe9d585be73 // Application token, will help
you to “beat off” unnecessary requests to the event handler, this field is in all events
    [expires_in] => 3600 // Token expiration time after which a new one will need to be requested
    [member_id] => d41d8cd98f00b204e9800998ecf8427e // Unique portal ID, required to renew
authorization
    [refresh_token] => 5f1ih5tsnsb11sc5heg3kp4ywqnjhd09 // Key to renew authorization
)
```

Attention! *Everything described below is the content of the* ***[data]*** field *in the event. The authorization data in the* ***[auth]*** key *contains the user data of the event initiator, to obtain the bot authorization data, you must use [data][BOT][__BOT_CODE__].*

Event for the chatbot to receive the message ONIMBOTMESSAGEADD

```
[BOT] => Array // Array of codes of chatbots the message is intended for
(
    [39] => Array
    (
   [AUTH] => Array // Parameters for authorization under the bot, to perform actions
        (
          [domain] => b24.hazz
          [member_id] => d41d8cd98f00b204e9800998ecf8427e
          [application_token] => 8006ddd764e69deb28af0c768b10ed65
        )
       [BOT_ID] => 39
       [ BOT_CODE ] = > newbot
    )
)
[PARAMS] => Array // Array of message data
(
    [DIALOG_ID] => 1 // Dialog ID
    [CHAT_TYPE] => P // Type of message and chat, can be P (one-on-one chat), C (limited number of
participants), O (public chat), S (supervisor chatbot)
    [CHAT_ENTITY_TYPE] => 'LINES' // to identify the chat (you can set this field at the time of creation),
for chats of open lines, this field will indicate LINES
    [CHAT_ENTITY_ID] => 13 // to identify the chat (you can set these fields at the time of creation)
    [MESSAGE_ID] => 392 // Message ID
    [MESSAGE] => test3 // Message
    [MESSAGE_ORIGINAL] => [USER=39]NewBot[/ USER] test3 // Original message with the BB code of
the chatbot (parameter only available in group chats)
    [FROM_USER_ID] => 1 // ID of the user who sent the message
    [TO_USER_ID] => 1 // Bot or user ID: in a one-on-one conversation, this will be the bot ID, and in a group
chat, the ID of the user to whom the call was directed. If 0 is specified, it means to all chat participants
    [TO_CHAT_ID] => 6 // Chat ID (only available in group chats)
```

```
    [LANGUAGE] => ru // Identifier language is the default portal
  )
  [USER] => Array // Array of post author data, can be empty if ID = 0
  (
    [ID] => 1 // User ID
    [NAME] => Evgeny Shelenkov // User's first and last name
    [FIRST_NAME] => Eugene // Username
    [LAST_NAME] => Shelenkov // User last name
    [WORK_POSITION] => // Position held
    [GENDER] => M // Gender, can be either M (male) or F (female)
    [IS_BOT] => 'Y' // This user is a bot (Y), otherwise N.
    [IS_CONNECTOR] => 'Y' // This is the user - connector (member of the OL chat, client),
otherwise - N.
    [IS_NETWORK] => 'Y' // This is a network user (can be a member of the OL chat, a client or just an external
user), otherwise - N.
    [IS_EXTRANET] => 'Y' // This is the extranet user (all users that are not intranet), N otherwise.

  )
```

Event for updating the ONIMBOTMESSAGEUPDATE message by the chatbot

```
  [BOT] => Array // Array of codes of chatbots the message is intended for
  (
    [39] => Array
    (
   [AUTH] => Array // Parameters for authorization under the bot, to perform actions
      (
        [domain] => b24.hazz
        [member_id] => d41d8cd98f00b204e9800998ecf8427e
        [application_token] => 8006ddd764e69deb28af0c768b10ed65
      )
     [BOT_ID] => 39
     [ BOT_CODE ] = > newbot
    )
  )
  [PARAMS] => Array // Array of message data
  (
    [DIALOG_ID] => 1 // Dialog ID
    [CHAT_TYPE] => P // Type of message and chat, can be P (one-on-one chat), C (limited number of
participants), O (public chat), S (supervisor chatbot)

    [CHAT_ENTITY_TYPE] => 'LINES' // to identify the chat (you can set this field at the time of creation),
for chats of open lines, this field will indicate LINES
    [CHAT_ENTITY_ID] => 13 // to identify the chat (you can set these fields at the time of creation)

    [MESSAGE_ID] => 392 // Message ID
    [MESSAGE] => test3 // Message
    [MESSAGE_ORIGINAL] => [USER=39]NewBot[/ USER] test3 // Original message with the BB code of
the chatbot (parameter only available in group chats)
    [FROM_USER_ID] => 1 // ID of the user who sent the message
    [TO_USER_ID] => 1 // Bot or user ID: in a one-on-one conversation, this will be the bot ID, and in a group
chat, the ID of the user to whom the call was directed. If 0 is specified, it means to all chat participants
```

```
    [TO_CHAT_ID] => 6 // Chat ID (only available in group chats)

    [LANGUAGE] => ru // Identifier language is the default portal
  )
  [USER] => Array // Array of post author data, can be empty if ID = 0
  (
    [ID] => 1 // User ID
    [NAME] => Evgeny Shelenkov // User's first and last name
    [FIRST_NAME] => Eugene // Username
    [LAST_NAME] => Shelenkov // User last name
    [WORK_POSITION] => // Position held
    [GENDER] => M // Gender, can be either M (male) or F (female)
    [IS_BOT] => 'Y' // This user is a bot (Y), otherwise N.
    [IS_CONNECTOR] => 'Y' // This is the user - connector (member of the OL chat, client),
otherwise - N.
    [IS_NETWORK] => 'Y' // This is a network user (can be a member of the OL chat, a client or just an external
user), otherwise - N.
    [IS_EXTRANET] => 'Y' // This is the extranet user (all users that are not intranet), N otherwise.

  )
```

Event for the chatbot to delete the ONIMBOTMESSAGEDELETE message

```
  [BOT] => Array // Array of codes of chatbots the message is intended for
  (
    [39] => Array
    (
   [AUTH] => Array // Parameters for authorization under the bot, to perform actions
      (
        [domain] => b24.hazz
        [member_id] => d41d8cd98f00b204e9800998ecf8427e
        [application_token] => 8006ddd764e69deb28af0c768b10ed65
      )
     [BOT_ID] => 39
     [ BOT_CODE ] = > newbot
    )
  )
  [PARAMS] => Array // Array of message data
  (
    [DIALOG_ID] => 1 // Dialog ID
    [CHAT_TYPE] => P // Type of message and chat, can be P (one-on-one chat), C (limited number of
participants), O (public chat), S (supervisor chatbot)

    [CHAT_ENTITY_TYPE] => 'LINES' // to identify the chat (you can set this field at the time of creation),
for chats of open lines, this field will indicate LINES
    [CHAT_ENTITY_ID] => 13 // to identify the chat (you can set these fields at the time of creation)

    [MESSAGE_ID] => 392 // Message ID
    [MESSAGE] => test3 // Message
    [FROM_USER_ID] => 1 // ID of the user who sent the message
    [TO_CHAT_ID] => 6 // Chat ID (only available in group chats)

    [LANGUAGE] => ru // Identifier language is the default portal
  )
```

```
[USER] => Array // Array of post author data, can be empty if ID = 0
(
    [ID] => 1 // User ID
    [NAME] => Evgeny Shelenkov // User's first and last name
    [FIRST_NAME] => Eugene // Username
    [LAST_NAME] => Shelenkov // User last name
    [WORK_POSITION] => // Position held
    [GENDER] => M // Gender, can be either M (male) or F (female)
    [IS_BOT] => 'Y' // This user is a bot (Y), otherwise N.
    [IS_CONNECTOR] => 'Y' // This is the user - connector (member of the OL chat, client),
otherwise - N.
    [IS_NETWORK] => 'Y' // This is a network user (can be a member of the OL chat, a client or just an
external user), otherwise - N.
    [IS_EXTRANET] => 'Y' // This is the extranet user (all users that are not intranet), N otherwise.

)
```

Event for the chatbot to receive the ONIMCOMMANDADD command

```
[COMMAND] => Array // Array of commands that were called by the user
(
    [14] => Array
    (
     [AUTH] => Array // Parameters for authorization under the chatbot to perform actions

     (
       [domain] => b24.hazz
       [member_id] => d41d8cd98f00b204e9800998ecf8427e
       [application_token] => 8006ddd764e69deb28af0c768b10ed65
     )
     [BOT_ID] => 62 // Chatbot ID
     [BOT_CODE] => echobot // Chatbot code
     [COMMAND] => echo // Called command
     [COMMAND_ID] => 14 // Command ID
     [COMMAND_PARAMS] => test // Parameters with which the command was called
     [COMMAND_CONTEXT] => TEXTAREA // Command execution context. TEXTAREA if
the command is entered manually, or KEYBOARD if you pressed a button in the keyboard
     [MESSAGE_ID] => 1221 // ID of the message to be answered
    )
)
[PARAMS] => Array // Array of message data
(
    [DIALOG_ID] => 1 // Dialog ID
    [CHAT_TYPE] => P // Message and chat type, can be P (one-to-one chat), C (limited number of
participants), O (public chat)
    [MESSAGE_ID] => 1221 // Message ID
    [MESSAGE] => /echo test // Message
    [MESSAGE_ORIGINAL] => /echo test // Original message with bot's BB code (option available only in
group chats)
    [FROM_USER_ID] => 1 // ID of the user who sent the message
    [TO_USER_ID] => 2 // ID of another user (parameter only available in one-on-one chats)

    [TO_CHAT_ID] => 6 // Chat ID (only available in group chats)
```

```
    [LANGUAGE] => ru // Identifier language is the default portal
  )
  [USER] => Array // Array of post author data, can be empty if ID = 0
  (
    [ID] => 1 // User ID
    [NAME] => Evgeny Shelenkov // User's first and last name
    [FIRST_NAME] => Eugene // Username
    [LAST_NAME] => Shelenkov // User last name
    [WORK_POSITION] => // Position held
    [GENDER] => M // Gender, can be either M (male) or F (female)
  )
```

Event for receiving information by the chatbot about its inclusion to chat (or personal correspondence) ONIMBOTJOINCHAT

```
  [BOT] => Array // Array of codes of chatbots that the event is intended for
  (
    [39] => Array // Parameters for authorization under the chatbot to perform actions
    (
   [AUTH] => Array // Parameters for authorization under the chatbot to perform actions
      (
        [domain] => b24.hazz
        [member_id] => d41d8cd98f00b204e9800998ecf8427e
        [application_token] => 8006ddd764e69deb28af0c768b10ed65
      )
      [BOT_ID] => 39
      [ BOT_CODE ] = > newbot
    )
  )
  [PARAMS] => Array
  (
    [DIALOG_ID] => 1 // Dialog ID
    [BOT_ID] => 39 // Identifier bot
    [CHAT_TYPE] => P // Type of message and chat, can be P (one-on-one chat), C (limited number of
participants), O (public chat), S (supervisor chatbot)

    [CHAT_ENTITY_TYPE] => 'LINES' // Chat subtype, can be set to LINES
(Open lines) or empty
    [USER_ID] => 1 // User ID (for one-on-one chat - who opened
chat bot, for group chats - the inviter of the chat bot)
    [LANGUAGE] => ru // Identifier language is the default portal
  )
  [USER] => Array // User data array, may be empty if ID = 0
  (
    [ID] => 1 // User ID
    [NAME] => Evgeny Shelenkov // User's first and last name
    [FIRST_NAME] => Eugene // Username
    [LAST_NAME] => Shelenkov // User last name
    [WORK_POSITION] => // Position held
    [GENDER] => M // Gender, can be either M (male) or F (female)
  )
```

Chatbot deletion event ONIMBOTDELETE

[BOT_ID] => 39 // Chatbot ID
[BOT_CODE] => giphy // Chatbot code

An example of a handler code for working with events

The example is fully working, saves the configuration to a file, it will help you understand how to chatbots are working.

A file with comments and an avatar is available in the GitHub repository

Full list of Bot API methods and events

Working with chatbots

Method	Description of the method
imbot.register	Chatbot registration.
imbot.unregister	Removing a chatbot from the system.
imbot.update	Bot data update.
imbot.bot.list	Getting available chatbots.

Working with chats

Method	Description of the method
imbot.chat.user.list	Getting a list of participants.
imbot.chat.user.add	Participant invitation.
imbot.chat.user.delete	Exclusion of participants.
imbot.chat.leave	Exit the chatbot from the specified chat.
imbot.chat.updateTitle	Chat title update.
imbot.chat.updateColor	Chat color update.
imbot.chat.updateAvatar	Chat avatar update.
imbot.chat.add	Create chat from persons chat-bot.
imbot.chat.get	Getting an ID chat.

imbot.chat.setOwner	Change chat owner on behalf of chat bot.
imbot.dialog.get	Getting information about dialogue.

Working with messages

Method	Description of the method
imbot.message.add	Sending a message from a chatbot.
imbot.message.update	Sending a change message chat bot.
imbot.message.delete	Delete a chatbot message.
imbot.message.like	"I like" setting.
imbot.chat.sendTyping	Sending a Chatbot message writes a message...

Working with teams

Method	Description of the method
imbot.command.register	Team registration for chatbot processing.
imbot.command.unregister	Removing processing commands.
imbot.command.update	Updating data in team.
lust.command.answer	Posting a response to command.

Working with events

Event	Description of the event
ONAPPINSTALL	Installation event

	applications.
ONAPPUPDATE	Update event applications.
ONIMBOTMESSAGEADD	Event for the chatbot to receive a message.
update event chatbot messages.	ONIMBOTMESSAGEUPDATE
ONIMBOTMESSAGEDELETE	Delete event chatbot messages.
ONIMCOMMANDADD	Event for receiving a command by the chatbot.
ONIMJOINCHAT	Event for receiving information by the chatbot about its inclusion in the chat (or personal correspondence).
ONIMBOTDELETE	Event to delete the chatbot.

Chat API

To work with **the Chat API,** when creating an application (or uploading a new version), you need to specify scope: **im** (Chat and notifications).

Platform version

im.revision.get -- getting information about API revisions

Revision: 18

Options

No.

Call example

JavaScript

```
BX24.callMethod('im.revision.get', {}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
```

```
    }
});
```

PHP

```
$result = restCommand('im.revision.get', Array(
), $_REQUEST["auth"]);
```

Response Example

```
{
    "result": {
        "rest": 18,
        "web": 117,
        "mobile": 9,
    }
}
```

Description of keys:

- rest -- api revision for rest clients
- web -- API revision for web/desktop client
- mobile -- API revision for mobile client

Note! The method specified using the restCommand function is the send method data in Bitrix24, this method is in the EchoBot example, and is presented here as an example. You you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Working with chats

Note! All methods are specified using the restCommand function - this is the method sending data to Bitrix24, this method is in the EchoBot example, and is presented here as example. You can use your function or javascript method BX24.callMethod or bitrix24-php sdk.

Method	Description of the method
im.chat.add	Create a chat.
im.chat.user.list	Getting a list of participants.
im.chat.user.add	Participant invitation.
im.chat.user.delete	Exclusion of participants.
im.chat.user.add	Leave chat.
im.chat.updateTitle	Chat title update.
im.chat.updateColor	Chat color update.

im.chat.updateAvatar	Chat avatar update.
im.chat.setOwner	Change the owner of the chat.
im.chat.get	Get the chat ID.
im.chat.mute	Disable chat notifications.

im.chat.add

Create a chat

Revision: 18

Note! The method specified using the restCommand function is the send method data in Bitrix24, this method is in the EchoBot example, and is presented here as an example. You you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
TYPE	CHAT	No	Chat type OPEN | CHAT (OPEN - chat open for entry, CHAT - regular chat by invitations, CHAT by default)	18
TITLE	My new private chat	No	Chat title	18
DESCRIPTION	Very important chat	No	Description chat	18
COLOR	PINK	No	Chat color for mobile applications: RED, GREEN, MINT, LIGHT_BLUE, DARK_BLUE, PURPLE, AQUA, PINK, LIME, BROWN, AZURE, KHAKI, SAND, MARENGO, GRAY, GRAPHITE	18
MESSAGE	Good welcome to chat	No	First welcome message in chat	18
USERS	Array(1,2)	**Yes**	Chat members	18
AVATAR	base64 image	No	Chat avatar in base64 format	18
ENTITY_TYPE CHAT		No	Entity ID, maybe be used to search for this	18

			field and for easy definition of context in event handlers ONIMBOTMESSAGEADD, ONIMBOTMESSAGEUPDATE, ONIMBOTMESSAGEDELETE	
ENTITY_ID	13	No	Numeric identifier of the entity, can be used to search for chat and to easily define context in event handlers ONIMBOTMESSAGEADD, ONIMBOTMESSAGEUPDATE, ONIMBOTMESSAGEDELETE	18
OWNER_ID	39	No	Chat owner ID. You can not specify if you are creating a chat under the desired user	18

Method call and response

PHP

```
$result = restCommand('im.chat.add', Array(

    'TYPE' => 'CHAT',
    'TITLE' => 'My new private chat',
    'DESCRIPTION' => 'Very important chat',
    'COLOR' => 'PINK',
    'MESSAGE' => 'Welcome to chat',
    'USERS' => Array(1,2),
    'AVATAR' => 'base64 image',
    'ENTITY_TYPE' => 'CHAT',
    'ENTITY_ID' => 13,
    'OWNER_ID' => 39,

), $_REQUEST["auth"]);
```

Response Example

```
{
    "result": 123
}
```

Example response when an error occurs

```
{
    "error": "USERS_EMPTY",
    "error_description": "No chat members passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
USERS_EMPTY	Chat members not transferred
WRONG_REQUEST	Something went wrong

im.chat.user.list

Getting a list of participants

Revision: 18

Note! The method specified using the restCommand function is the send method data in Bitrix24, this method is in the EchoBot example, and is presented here as an example. You you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter Example	Required	Description	Revision
CHAT_ID 13	**Yes**	Chat ID	18

Method call and response

PHP

```
$result = restCommand('im.chat.user.list', Array(

   'CHAT_ID' => 13

), $_REQUEST["auth"]);
```

Response Example

```
{
   "result": [1,2]
}
```

Example response when an error occurs

```
{
   "error": "CHAT_ID_EMPTY",
   "error_description": "Chat ID not passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	The chat ID was not passed, or an invalid ID was passed.

im.chat.user.add

Invitation of participants

Revision: 18

Note! The method specified using the restCommand function is the send method data in Bitrix24, this method is in the EchoBot example, and is presented here as an example. You you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
CHAT_ID	13	**Yes**	Chat ID	18
USERS	Array(3,4)	**Yes**	New User IDs	18

Method call and response

PHP

```
$result = restCommand('im.chat.user.add', Array(

   'CHAT_ID' => 13,
   'USERS' => Array(3,4)

), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID not passed"
```

```
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	Chat id not passed
WRONG_REQUEST	You do not have sufficient permissions to add a member to the chat, or the member already is in a chat

im.chat.user.delete

Exclusion of members

Revision: 18

Note! The method specified using the restCommand function is the send method data in Bitrix24, this method is in the EchoBot example, and is presented here as an example. You you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
CHAT_ID	13	**Yes**	Chat ID	18
USER_ID	4	**Yes**	Exclude ID user	18

Method call and response

PHP

```
$result = restCommand('im.chat.user.delete', Array(

    'CHAT_ID' => 13,
    'USER_ID' => 4

), $_REQUEST["auth"]);
```

Response Example

```
{
```

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

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID not passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	Chat id not passed
USER_ID_EMPTY	User ID not passed
WRONG_REQUEST	You do not have sufficient rights to add member to the chat or the member is already in the chat

im.chat.leave

Leave chat

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
CHAT_ID	13	**Yes**	Identifier chat	18

Method call and response

PHP

```
$result = restCommand('im.chat.leave', Array(
    'CHAT_ID' => 13
```

```
), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID not passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	Chat id not passed
WRONG_REQUEST	You are not a member of this chat

im.chat.updateTitle

Chat title update

Revision: 18

Note! The method specified using the restCommand function is the send method data in Bitrix24, this method is in the EchoBot example, and is presented here as an example. You you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
CHAT_ID	13	No	Chat ID	18
TITLE	New name for chat	No	New title	18

Method call and response

PHP

```
$result = restCommand('im.chat.updateTitle', Array(

    'CHAT_ID' => 13,
    'TITLE' => 'New name for the chat'

), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID not passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	Chat id not passed
TITLE_EMPTY	New chat title not passed
WRONG_REQUEST	Title already set or specified chat does not exist

im.chat.updateColor

Chat color update

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision

CHAT_ID 13	**Yes**	Identifier chat	18
COLOR MINT	**Yes**	Chat color for mobile app (RED, GREEN, MINT, LIGHT_BLUE, DARK_BLUE, PURPLE, AQUA, PINK, LIME, BROWN, AZURE, KHAKI, SAND, MARENGO, GRAY, GRAPHITE)	18

Method call and response

PHP

```
$result = restCommand('im.chat.updateColor', Array(

    'CHAT_ID' => 13,
    'COLOR' => 'MINT'

), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID not passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	Chat id not passed
WRONG_COLOR	The color is not in the list of available colors

WRONG_REQUEST The color is already set or the specified chat does not exist

im.chat.updateAvatar

Chat avatar update

Revision: 18

Note! The method specified using the restCommand function is the send method data in Bitrix24, this method is in the EchoBot example, and is presented here as an example. You you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter Example	Required	Description	Revision
CHAT_ID 13	**Yes**	Chat ID	18
AVATAR /* base64 image */	**Yes**	base64 image	18

Method call and response

PHP

```
$result = restCommand('im.chat.updateAvatar', Array(

    'CHAT_ID' => 13,
    'AVATAR' => '/* base64 image */'

), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID not passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	Chat id not passed
AVATAR_ERROR	Incorrect image format sent
WRONG_REQUEST	Chat does not exist

im.chat.setOwner

Change chat owner

Revision: 18

Note! The method specified using the restCommand function is the send method data in Bitrix24, this method is in the EchoBot example, and is presented here as an example. You you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
CHAT_ID	13	**Yes**	Chat ID	18
USER_ID	2	**Yes**	ID of the new chat owner	18

Method call and response

PHP

```
$result = restCommand('im.chat.setOwner', Array(

   'CHAT_ID' => 13,
   'USER_ID' => '2'

), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
   "error": "CHAT_ID_EMPTY",
   "error_description": "Chat ID not passed"
```

```
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	Chat id not passed
USER_ID_EMPTY	New owner id not passed
WRONG_REQUEST	Called by a non-owner of the chat, or the specified user is not a member chat

im.chat.get

Getting a chat ID

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Rev
ENTITY_TYPE	CRM - from CRM, LINES or LIVECHAT - for open lines	**Yes**	Entity identifier. Can be used for search chat and for easy context definitions in event handlers ONIMBOTMESSAGEADD, ONIMBOTMESSAGEUPDATE, ONIMBOTMESSAGEDELETE	18
ENTITY_ID	LEAD|13 - for CRM, facebook|2|1647215182041969|862 - for open lines	**Yes**	The numeric ID of the entity. May be used to search chat and for easy definition of context in event handlers ONIMBOTMESSAGEADD, ONIMBOTMESSAGEUPDATE, ONIMBOTMESSAGEDELETE	18

Method call and response

PHP

```
$result = restCommand('im.chat.get', Array(

    'ENTITY_TYPE' => 'CRM',
    'ENTITY_ID' => 'LEAD|13',

), $_REQUEST["auth"]);
```

Response Example

```
{
    "result": 10
}
```

Execution result: returns chat ID CHAT_ID or null.

im.chat.mute

Disable chat notifications

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter Example	Required	Description	Revision
CHAT_ID 17	**Yes**	Chat ID	19
MUTE AND	**Yes**	Disable or enable notifications - Y|N	19

- Option key **MUTE:** Y or N

Method call and response

JavaScript

```
BX24.callMethod('im.chat.mute', {
    'CHAT_ID': 17,
    'MUTE': 'Y'
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
```

```
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.chat.mute', Array(
    'CHAT_ID' => 17,
    'MUTE' => 'Y'
), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	Chat ID not passed

Working with Dialogs

Method	Description of the method
im.dialog.messages.get	Getting a list of recent messages in chat.
im.dialog.read	Changing the fact that messages have been read: All messages up to the specified one (including the message itself) are marked as read.

im.dialog.unread	Changing the fact that messages were read: all messages after the specified one (including the message itself) are marked as unread.

Note! All of the above methods are specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Methods specified using javascript method BX24.callMethod:

Method	Description of the method
im.dialog.users.list	Getting information about chat participants (pagination is supported).
im.dialog.writing	Sending the status "you are being contacted...".
im.dialog.read.all	Setting the "read" mark for everyone dialogues.

im.dialog.messages.get

Getting a list of recent messages in a chat

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
DIALOG_ID	chat29 or 256	**Yes**	Dialog ID. Format: **chatXXX** -- chat recipient if the message is for chat or **XXX** is the recipient ID if the message is for private dialogue	19

LAST_ID	28561624	No	Identifier last loaded message	19
FIRST_ID	454322	No	ID of the first loaded message	19
LIMIT	20	No	Restriction on the selection of messages in dialogue	19

- If the LAST_ID and FIRST_ID keys are not passed, the last 20 chat messages will be loaded.
- To load the previous 20 messages, you must pass LAST_ID with the ID of the minimum message ID obtained from the last fetch.
- If you need to download the next 20 messages, you must pass FIRST_ID with the ID of the maximum message ID obtained from the last fetch.
- If you need to download the first 20 messages, you must pass FIRST_ID with ID 0, you will be returned the result with the very first message available to you in this chat.

Note! Due to the potentially large amount of data, this method does not support the standard Bitrix24 Rest API pagination.

Method call and response

JavaScript

```
BX24.callMethod('im.dialog.messages.get', {
    DIALOG_ID: 'chat29'
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.dialog.messages.get', Array(
    'DIALOG_ID': 'chat29'
), $_REQUEST["auth"]);
```

Response Example


```
{
    "result": {
        "messages": [
         {
            "id": 28561624,
            "chat_id": 29,
            "author_id": 1,
            "date": "2018-01-29T16:58:47+03:00",
            "text": "http:// yandex.ru",
            "params": {
              "URL_ID": [
                5
              ],
              "URL_ONLY": "Y",
              "ATTACH": [
                {
                  "ID": "5",
                  "BLOCKS": [
                    {
                      "RICH_LINK": [
                        {
                          "NAME": "yandex",
                          "LINK": "http:// yandex.ru/",
                          "DESC": "There is everything",
                          "PREVIEW": "https:// yastatic.net/ morda-logo/ i/ share-logo-ru.png"
                        }
                      ]
                    }
                  ],
                  "COLOR": "transparent"
```

```
        }
       ]
      }
     },
     {
      "id": 28561623,
      "chat_id": 29,
      "author_id": 28,
      "date": "2018-01-29T16:43:38+03:00",
      "text": "",
      "params": {
       "FILE_ID": [
        540
       ]
      }
     },
     {
      "id": 28561622,
      "chat_id": 29,
      "author_id": 1,
      "date": "2018-01-29T16:43:12+03:00",
      "text": "Yes it works :) :)",
      "params": {
       "IS_EDITED": "Y"
      }
     },
     {
      "id": 28561618,
      "chat_id": 29,
      "author_id": 0,
      "date": "2018-01-25T15:15:22+03:00",
      "text": "Evgeny Shelenkov changed the chat topic to \"Big Chat\"",
      "params": null
     },
    ],
    "users": [
     {
      "id": 1,
      "name": "Evgeny Shelenkov",
      "first_name": "Eugene",
      "last_name": "Shelenkov",
      "work_position": "",
      "color": "#df532d",
      "avatar": "http://192.168.2.232/upload/resize_cache/main/1d3/100_100_2/shelenkov.png",
      "gender": "M",
      "birthday": "",
      "extranet": false,
      "network": false,
      "bot": false,
      "connector": false,
      "external_auth_id": "default",
```

```
      "status": "online",
      "idle": false,
      "last_activity_date": "2018-01-29T17:35:31+03:00",
      "mobile_last_date": false,
      "departments": [
        50
      ],
      "absent": false,
      "phones": {
        "work_phone": "",
        "personal_mobile": "",
        "personal_phone": ""
      }
    },
    {
      "id": 28,
      "name": "Ivan Elpidin",
      "first_name": "Ivan",
      "last_name": "Elpidin",
      "work_position": "manager",
      "color": "#728f7a",
      "avatar": "http://192.168.2.232/upload/resize_cache/main/8b8/100_100_2/26.jpg",
      "gender": "M",
      "birthday": "26-01",
      "extranet": false,
      "network": false,
      "bot": false,
      "connector": false,
      "external_auth_id": "default",
      "status": "online",
      "idle": false,
      "last_activity_date": false,
      "mobile_last_date": false,
      "departments": [
        58
      ],
      "absent": false,
      "phones": {
        "work_phone": "8 (495) 222-33-55",
        "personal_mobile": "8 (495) 123-55-65",
        "personal_phone": ""
      }
    },
  ],
  "files": [
    {
      "id": 540,
      "chatId": 29,
      "date": "2018-01-29T16:43:39+03:00",
      "type": "image",
      "preview": "",
```

```
                "name": "1176297_698081120237288_696773366_n.jpeg",
                "size": 55013,
                "image": {
                    "width": 960,
                    "height": 640
                },
                "status": "done",
                "progress": 100,
                "authorId": 1,
                "authorName": "Evgeny Shelenkov",
                "urlPreview": "http://192.168.2.232/bitrix/components/bitrix/im.messenger/show.file.php?fileId=540&preview=Y&fileName=1176297_698081120237288_696773366_n.jpeg",
                "urlShow": "http://192.168.2.232/bitrix/components/bitrix/im.messenger/show.file.php?fileId=540&fileName=1176297_698081120237288_696773366_n.jpeg",
                "urlDownload": "http://192.168.2.232/bitrix/components/bitrix/im.messenger/download.file.php?fileId=540"
            }
        ],
        "chat_id": 29
    }
}
```


Description of keys:

- messages -- array of messages:
  - id -- message ID
  - chat_id -- chat ID
  - author_id -- the author of the message (0 - if the message is a system one)
  - date -- date of the message in ATOM format
  - text -- message text
  - params -- message parameters, parameter object if parameters are not passed null (basic types will be described below)
- users -- user data description objects:
  - id -- user ID
  - name -- user's first and last name
  - first_name -- username
  - last_name -- user last name
  - work_position -- position
  - color -- user color in hex format
  - avatar -- link to the avatar (if empty, no avatar has been set)
  - gender -- user's gender
  - birthday -- user's birthday in DD-MM format, if empty -- not set
  - extranet -- sign of an external extranet user (true/false)
  - network -- attribute of Bitrix24.Network user (true/false)
  - bot -- sign of a bot (true/false)

- connector -- indication of the user of open lines (true/false)
- external_auth_id -- external authorization code
- status -- selected user status
- idle -- date the user left the computer, in ATOM format (false if not set)
- last_activity_date -- date of the user's last activity in ATOM format
- mobile_last_date -- date of the last action in the mobile application in ATOM format (if not set, false)
- absent -- date by which date the user has a vacation, in ATOM format (if not set, false)

- files -- an object describing the files in the selected messages:
  - id -- file ID
  - chatId -- chat identifier
  - date -- date the file was modified
  - type -- file type: image -- image, file -- file
  - name -- name of the uploaded file
  - size -- the actual size of the image in bytes
  - image -- actual image size in px
  - width -- width in px
  - height -- height in px
  - status -- current status: done -- uploaded, upload -- uploading
  - progress -- download status in percent: 100 -- uploaded, -1 -- current status not available
  - authorId -- identifier of the user who uploaded the object
  - authorName -- first and last name of the user who uploaded the object
  - urlPreview -- link to image preview (available for images)
  - urlShow -- link to go to the object in "view" mode
  - urlDownload -- link to start downloading
- chat_id -- chat ID

Description of additional options:

- ATTACH -- an object containing a rich appearance
- KEYBOARD -- an object containing a description of the keyboard
- IS_DELETED -- flag indicating that the message has been deleted
- IS_EDITED -- flag indicating that the post has been edited
- FILE_ID -- array of file IDs
- LIKE -- array of user IDs that voted for the post

Example response when an error occurs

```
{
  "error": "DIALOG_ID_EMPTY",
  "error_description": "Dialog ID can't be empty"
```

```
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
DIALOG_ID_EMPTY	Dialog ID not passed
ACCESS_ERROR	The current user does not have access rights to dialogue

im.dialog.read

Changing the fact that messages have been read: all messages up to the specified one (including the message itself) are marked as read

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
DIALOG_ID	chat29 or 256	**Yes**	Dialog ID. Format: **chatXXX** -- the recipient's chat if the message is for chat or **XXX** -- the recipient's ID if the message is for private dialogue	21
MESSAGE_ID	12	**Yes**	ID of the last read message in the dialog	21

Method call and response

JavaScript

```
BX24.callMethod('im.dialog.read', {
  'DIALOG_ID': chat29,
  'MESSAGE_ID': 12,
}, function(result){
  if(result.error())
```

```
  {
    console.error(result.error().ex);
  }
  else
  {
    console.log(result.data());
  }
});
```

PHP

```
$result = restCommand('im.dialog.read', Array(
  'DIALOG_ID' => chat29,
  'MESSAGE_ID' => 12,
), $_REQUEST["auth"]);
```

Response Example

```
{
    "result":
    {
        dialogId: "chat76",
        chatId: 76,
        counter: 1,
        loadId: 6930
    }
}
```

- **dialogId** -- identifier of the read dialog
- **chatId** -- chat ID **counter** --
- number of unread messages after method execution **lastId** -- last message read
-

If the method failed to set a new read mark:

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

Example response when an error occurs

```
{
    "error": "MESSAGE_ID_ERROR",
    "error_description": "Message ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description

MESSAGE_ID_ERROR Invalid message ID specified	
DIALOG_ID_EMPTY	Invalid dialog ID specified

im.dialog.unread

Changing the fact that messages were read: all messages after specified (including the message itself) are marked as unread

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
DIALOG_ID	chat29 or 256	**Yes**	Dialog ID. Format: **chatXXX** -- the recipient's chat if the message is for chat or **XXX** -- the recipient's ID if the message is for private dialogue	21
MESSAGE_ID 12		**Yes**	First ID unread dialogue	21

Method call and response

JavaScript

```
BX24.callMethod('im.dialog.unread', {
  'DIALOG_ID': chat29,
  'MESSAGE_ID': 12,
}, function(result){
  if(result.error())
  {
    console.error(result.error().ex);
  }
  else
  {
    console.log(result.data());
  }
});
```

PHP

```
$result = restCommand('im.dialog.unread', Array(
  'DIALOG_ID' => chat29,
  'MESSAGE_ID' => 12,
), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
    "error": "MESSAGE_ID_ERROR",
    "error_description": "Message ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
MESSAGE_ID_ERROR	Invalid message ID specified
DIALOG_ID_EMPTY	Invalid dialog ID specified

im.dialog.get

Getting information about a dialog

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
DIALOG_ID	chat29 or 256	**Yes**	Dialog ID. Format: **chatXXX** -- the recipient's chat if the message is for a chat, or **XXX** -- the recipient's ID if the message is for a private conversation	24

Method call and response

JavaScript

```
BX24.callMethod('im.dialog.get', {
    DIALOG_ID: 'chat29'
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.dialog.get', Array(
    'DIALOG_ID': 'chat29'
), $_REQUEST["auth"]);
```

Response Example


```
{
    "result":
  {
    "id": "21191",
    "title": "Mint Chat #3",
    "owner": "2",
    "extranet": false,
    "avatar": "",
    "color": "#4ba984",
    "type": "chat",
    "entity_type": "",
    "entity_data_1": "",
    "entity_data_2": "",
    "entity_data_3": "",
    "date_create": "2017-10-14T12:15:32+02:00",
    "message_type": "C"
  }
}
```


Description of keys:

- id -- chat identifier
- title -- chat name
- owner -- user ID of the chat owner

- extranet -- sign of participation in the chat of an external extranet user (true/false)
- color -- chat color in hex format
- avatar -- link to the avatar (if empty, no avatar has been set)
- type -- chat type (group chat, call chat, open line chat, etc.)
- entity_type -- external code for chat -- type
- entity_id -- external code for chat -- identifier
- entity_data_1 -- external data for chat
- entity_data_2 -- external data for chat
- entity_data_3 -- external data for chat
- date_create -- chat creation date in ATOM format
- message_type -- chat message type

Example response when an error occurs

```
{
    "error": "DIALOG_ID_EMPTY",
    "error_description": "Dialog ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
DIALOG_ID_EMPTY	Dialog ID not passed
ACCESS_ERROR	The current user does not have access rights to the dialog

im.dialog.users.list

Getting information about chat participants (pagination supported)

Revision: 30

Options

Parameter	Example	Required	Description	Revision
DIALOG_ID	chat74	**Yes**	Dialog ID. Format: • **chatXXX** -- chat	30

			recipient, if message for chat or • **XXX** -- recipient ID, if message for private dialogue	
SKIP_EXTERNAL	N	**No**	Skip all systemic users - 'Y'|'N' (default 'N')	30
SKIP_EXTERNAL_EXCEPT_TYPES	'bot, email'	**No**	String with those types systemic users to keep in the selection	30

Method call and response

JavaScript

```
B24.callMethod(
  'im.dialog.users.list',
  {
    DIALOG_ID: 'chat74',
    SKIP_EXTERNAL: 'Y'
  },
  res => {
    if (res.error())
    {
      console.error(result.error().ex);
    }
    else
    {
      console.log(res.data())
    }
  }
)
```

Response Example

```
[
  {
```

```
    "id": 1019,
    "active": true,
    "name": "alexa shasha",
    "first_name": "alexa",
    "last_name": "shasha",
    "work_position": "",
    "color": "#df532d",
    "avatar": "",
    "gender": "M",
    "birthday": "",
    "extranet": false,
    "network": false,
    "bot": false,
    "connector": false,
    "external_auth_id": "default",
    "status": "online",
    "idle": false,
    "last_activity_date": "2021-10-30T11:24:12+02:00",
    "mobile_last_date": "2021-10-20T13:02:33+02:00",
    "absent": false,
    "departments": [
      1
    ],
    "phones": false
  },
  {
    "id": 1,
    "active": true,
    "name": "Aleksey Shakhvorostov",
    "first_name": "Aleksey",
    "last_name": "Shakhvorostov",
    "work_position": "",
    "color": "#df532d",
    "avatar": "",
    "gender": "M",
    "birthday": "",
    "extranet": false,
    "network": false,
    "bot": false,
    "connector": false,
    "external_auth_id": "default",
    "status": "online",
    "idle": false,
    "last_activity_date": "2021-10-30T13:36:34+02:00",
    "mobile_last_date": "2021-10-27T16:39:26+02:00",
    "absent": false,
    "departments": [
      1
    ],
    "phones": false
  }
```

```
]
```

Description of keys:

- id -- user ID
- active -- whether the user is active (not fired)
- name -- user's first and last name
- first_name -- username
- last_name -- user last name
- work_position -- position
- color -- user color in **hex** format
- avatar -- link to the avatar (if empty, no avatar has been set)
- gender -- user's gender
- birthday -- user's birthday in **DD-MM** format (if empty, not set)
- extranet -- sign of an external extranet user (true/false)
- network -- attribute of *Bitrix24.Network* user (true/false)
- bot -- sign of a bot (true/false)
- connector -- indication of the user of open lines (true/false)
- external_auth_id -- external authorization code
- status -- selected user status
- idle -- date the user left the computer, in **ATOM** format (false if not set)
- last_activity_date -- date of the user's last activity in **ATOM** format
- mobile_last_date -- date of the last action in the mobile application in **ATOM** format (if not set, then false)
- departments -- department identifiers
- absent -- date by which date the user has a vacation, in **ATOM** format (if not set, then false)
- phones -- array of phone numbers:
  - work_phone -- work phone
  - personal_mobile -- mobile phone
  - personal_phone -- home phone

Example response when an error occurs

```
{
  "error":"DIALOG_ID_EMPTY",
  "error_description":"Dialog ID can\u0027t be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
DIALOG_ID_EMPTY	The DIALOG_ID parameter is not set or does not match the format
ACCESS_ERROR	The current user does not have data access rights

im.dialog.writing

Sending the status "you are being contacted..."

Revision: 30

Options

Parameter	Example	Required	Description	Revision
DIALOG_ID	13	**Yes**	Dialog ID. Format: • **chatXXX** -- the recipient's chat if the message is for chat or • **XXX** -- recipient identifier if the message is for a private conversation	30

Method call and response

JavaScript

```
B24.callMethod(
  'im.dialog.writing',
  {},
  function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

Response Example

```
true
```

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID not passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
DIALOG_ID_EMPTY	The DIALOG_ID parameter was not passed or does not match the **XXX** format
CHAT_ID_EMPTY	The DIALOG_ID parameter was not passed or does not match the **chatXXX** format
ACCESS_ERROR	The user does not have rights to this chat

im.dialog.read.all

Setting the "read" mark for all dialogs

Revision: 30

Options

Parameters are not passed.

Method call and response

JavaScript

```
B24.callMethod(
  'im.dialog.read.all',
  {
  },
  res => console.log(res.data())
)
```

Response Example

```
true
```

Working with messages

Method	Description of the method
im.message.add	Sending a message to the chat.
im.message.update	Sending a chatbot message change.
im.message.delete	Delete a chatbot message.
im.message.like	"I like" setting.

Note! All of the above methods are specified using the function restCommand is a method for sending data to Bitrix24, this method is in the EchoBot example, And presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Methods specified using javascript method BX24.callMethod:

Method	Description of the method
im.message.command	Using a bot command.
im.message.share	Creating new entities by chat message: new chat, task, news post, event in calendar.

im.message.add

Sending a message to chat

Revision: 18

Note! The method specified using the restCommand function is the send method data in Bitrix24, this method is in the EchoBot example, and is presented here as an example. You you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
DIALOG_ID	chat13 or 256	**Yes**	Dialog ID. Format: **chatXXX** -- recipient's chat if chat message or **XXX** -- identifier recipient if the message is for private dialogue	18
MESSAGE	Text messages	**Yes**	Message text	18

SYSTEM	N	No	Display messages as system message or not, optional field, default 'N'	18
ATTACH		No	The attachment	18
URL_PREVIEW	Y	No	Convert links to rich links, optional field, by default 'Y'	18
KEYBOARD		No	Keyboard	18
MENU		No	Context menu	18

Method call and response

PHP

```
$result = restCommand('im.message.add', Array(

    'DIALOG_ID' => 'chat13',
    'MESSAGE' => 'Message text',
    'SYSTEM' => 'N',
    'ATTACH' => '',
    'URL_PREVIEW' => 'Y',
    'KEYBOARD' => '',
    'MENU' => '',

), $_REQUEST["auth"]);
```

Response Example

```
{
    "result": 11
}
```

Execution result: message identifier MESSAGE_ID or error.

Example response when an error occurs

```
{
    "error": "USER_ID_EMPTY",
    "error_description": "Recipient ID was not set when sending a message to
one-on-one chat
}
```

Description of keys:

- error -- code of the error that occurred

- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
USER_ID_EMPTY	Recipient ID not set in when sending a message in a one-on-one chat one
CHAT_ID_EMPTY	Recipient chat id not set when sending a message to the chat
ACCESS_ERROR	Insufficient rights to send messages
MESSAGE_EMPTY	Message text not sent
ATTACH_ERROR	The entire attachment object passed is not passed validation
ATTACH_OVERSIZE	Exceeded the maximum allowable attachment size (30 Kb)
KEYBOARD_ERROR	The entire passed keyboard object is not passed validation
KEYBOARD_OVERSIZE	Exceeded the maximum allowable keyboard size (30 Kb)
MENU_ERROR	The entire passed menu object is not passed validation
MENU_OVERSIZE	Exceeded the maximum allowable menu size (30 Kb)
PARAMS_ERROR	Something went wrong

Related links:

How to work with dialable keyboards
How to work with attachments
Message Formatting
Working with the context menu

im.message.update

Sending a chatbot message change

Revision: 18

Note! The method specified using the restCommand function is the send method data in Bitrix24, this method is in the EchoBot example, and is presented here as an example. You you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
MESSAGE_ID	1	**Yes**	Identifier messages	18
MESSAGE	Text messages	No	Text messages. If pass empty value, then the message will be deleted	18
ATTACH		No	The attachment	18
URL_PREVIEW Y		No	Transform links in rich links	18
KEYBOARD		No	Keyboard	18
MENU		No	Contextual menu	18

Related links:

How to work with dialable keyboards
How to work with attachments
Message Formatting

Method call and response

PHP

```
$result = restCommand('im.message.update', Array(

    'MESSAGE_ID' => 1,
    'MESSAGE' => 'Message text',
    'ATTACH' => '',
    'URL_PREVIEW' => 'Y',
    'KEYBOARD' => '',
    'MENU' => '',

), $_REQUEST["auth"]);
```

Response Example

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

Execution result: true or error.

Example response when an error occurs

```
{
    "error": "MESSAGE_ID_ERROR",
    "error_description": "Message ID not passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
MESSAGE_ID_ERROR	Message ID not passed
CANT_EDIT_MESSAGE	You do not have access to this post or time to modify it expired (more than 3 days have passed since the publication)
ATTACH_ERROR	The entire attachment object passed did not pass validation
ATTACH_OVERSIZE	Maximum attachment size exceeded (30 KB)
KEYBOARD_ERROR	The entire passed keyboard object failed validation
KEYBOARD_OVERSIZE	The maximum allowable keyboard size has been exceeded (30 KB)
MENU_ERROR	The entire passed menu object failed validation
MENU_OVERSIZE	Maximum allowable menu size exceeded (30 KB)

im.message.delete

Deleting a chatbot message

Revision: 18

Note! The method specified using the restCommand function is the send method data in Bitrix24, this method is in the EchoBot example, and is presented here as an example. You you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
MESSAGE_ID	1	**Yes**	Message ID	18

Method call and response

PHP

```
$result = restCommand('im.message.delete', Array(

    'MESSAGE_ID' => 1,

), $_REQUEST["auth"]);
```

Response Example

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

Execution result: true or error.

Example response when an error occurs

```
{
    "error": "MESSAGE_ID_ERROR",
    "error_description": "Message ID not passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
MESSAGE_ID_ERROR	Message ID not passed
CANT_EDIT_MESSAGE	You do not have access to this message

im.message.like

"Like" setting

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
MESSAGE_ID	1	**Yes**	Message ID (any a message sent in private conversations or in group chats where a chatbot is present)	18
ACTION	auto	No	Action associated with the method: plus - will put a "Like" label; minus - will remove the "I like" label; auto - automatically calculates whether to mark or unmark. If not specified, it will work in the mode auto	18

Method call and response

PHP

```
$result = restCommand('im.message.like', Array(

    'MESSAGE_ID' => 1,
    'ACTION' => 'auto',

), $_REQUEST["auth"]);
```

Response Example

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

Execution result: true or error.

Example response when an error occurs

```
{
    "error": "MESSAGE_ID_ERROR",
    "error_description": "Message ID not passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
MESSAGE_ID_ERROR	Message ID not passed
WITHOUT_CHANGES	After calling the status state "I like" has not changed

im.message.command

Using a bot command

Revision: 30

Options

Parameter	Example	Required	Description	Revision
MESSAGE_ID	278	**Yes**	Identifier messages with the ability give a command to the bot	30
BOT_ID	1	**Yes**	Bot ID in chat	30
COMMAND	'KEYBOARD'	**Yes**	The team to be execute bot	30
COMMAND_PARAMS	'stop'	**Yes**	Command Options	30

It is necessary to send a message containing a choice of bot commands.

Method call and response

JavaScript

```
B24.callMethod(
  'im.message.command',
  {
    MESSAGE_ID: 278,
    BOT_ID: 1,
    COMMAND: 'KEYBOARD',
```

```
    COMMAND_PARAMS: 'stop'
  },
  res => {
    if (res.error())
    {
      console.error(result.error().ex);
    }
    else
    {
      console.log(res.data())
    }
  }
)
```

Response Example

```
true
```

Example response when an error occurs

```
{
  "error":"PARAMS_ERROR",
  "error_description":"Incorrect params"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
MESSAGE_ID_ERROR	Parameter MESSAGE_ID not set or not is a number
BOT_ID_ERROR	The BOT_ID parameter is not set or is not number
COMMAND_ERROR	COMMAND parameter not set
PARAMS_ERROR	COMMAND_PARAMS is not set or not matches the bot command parameter

im.message.share

Create new entities by chat message: new chat,

task, news post, calendar event

Options

Parameter	Example	Required	Description	Revision
MESSAGE_ID 289		**Yes**	Message identifier by which the new entity will be created	30
DIALOG_ID	'chat74'	**Yes**	Dialog ID. Format: • **chatXXX** -- the recipient's chat if the message is for chat or • **XXX** -- recipient identifier if the message is for a private conversation	30
TYPE	'TASK'	**Yes**	Type of created entity: • 'CHAT' -- a new chat will be created on the message • 'TASK' -- according to the message will be task created • 'POST' -- the post will be created in the News • 'CALEND' -- the message will create a calendar event	30

It is necessary to send a message containing a choice of bot commands.

Method call and response

JavaScript

```
B24.callMethod(
  'im.message.share',
  {
    MESSAGE_ID: 289,
    DIALOG_ID: 'chat74',
    TYPE: 'CHAT',
  },
  res => {
```

```
    if (res.error())
    {
      console.error(result.error().ex);
    }
    else
    {
      console.log(res.data())
    }
  }
)
```

Response Example

```
true
```

Example response when an error occurs

```
{
  "error":"PARAMS_ERROR",
  "error_description":"Incorrect params"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
MESSAGE_ID_ERROR	The MESSAGE_ID parameter is not set or is not number
DIALOG_ID_EMPTY	DIALOG_ID is not set or does not match the format
ACCESS_ERROR	The current user does not have access rights to

	chat or dialogue
PARAMS_ERROR	The TYPE parameter is not set or does not match available

Working with files

Note! All methods are specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24- php-sdk.

Method	Description of the method
im.disk.folder.get	Getting information about the storage folder for chat files.
im.disk.file.commit	Publication of the uploaded file in the chat.
im.disk.file.delete	Deleting files inside the chat folder.
im.disk.file.save	Saving the file to your Bitrix24.Disk.

im.disk.folder.get

Getting information about a file storage folder for chat

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is available in

EchoBot ***example , and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24- php-sdk.***

Options

Parameter	Example	Required	Description	Revision
CHAT_ID	17	**Yes**	Identifier chat	18

Method call and response

JavaScript

```
BX24.callMethod('im.disk.folder.get', {
    'CHAT_ID': 17,
}, function(result)
    { if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.disk.folder.get', Array(
    'CHAT_ID' => 17,
), $_REQUEST["auth"]);
```

Response Example

```
{
    "result": {
```

```
        ID: 127
    }
}
```

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	Chat id not passed
ACCESS_ERROR	The current user does not have access rights to the dialog
INTERNAL_ERROR	Server error, contact technical support

im.disk.file.commit

Sharing an uploaded file in a chat

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can

use your own function or javascript method BX24.callMethod or bitrix24- php-sdk.

Options

Parameter	Example	Required	Description	Revision
CHAT_ID	17	**Yes**	Identifier chat	18
UPLOAD_ID	213	**Yes***	Identifier of the uploaded file via module methods DISK	18
DISK_ID	112	**Yes***	File ID available from local disk	18
MESSAGE	Important document	No	Description of the file will be published in chat	18
SILENT_MODE N		No	Parameter for Open Lines chat, indicates whether information about the file will be sent to the client or No	18

- For a successful API call, you need to specify CHAT_ID and one of two fields - - UPLOAD_ID or DISK_ID.

Note! The file must first be uploaded via method disk.folder.uploadfile.

Method call and response

JavaScript

```
BX24.callMethod('im.disk.file.commit', {
    'CHAT_ID': 17,
    'UPLOAD_ID': 112,
}, function(result){
    if(result.error()) {

        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.disk.file.commit', Array(
    'CHAT_ID' => 17,
    'UPLOAD_ID' => 112,
), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	Chat ID not passed
ACCESS_ERROR	The current user does not have access rights to dialogue
FILES_ERROR	No file id passed

im.disk.file.delete

Deleting files inside the chat folder

Revision: 18

Note! Method specified using function restCommand is a method for sending data to Bitrix24, this method is available in EchoBot ***example , and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.***

Options

Parameter	Example	Required	Description	Revision
CHAT_ID	17	**Yes**	Identifier chat	18
DISK_ID	112	**Yes**	Identifier file	18

Method call and response

JavaScript

```
BX24.callMethod('im.disk.file.delete', {
    'CHAT_ID': 17,
    'DISK_ID': 112,
}, function(result){
    if(result.error()) {

        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.disk.file.delete', Array(
    'CHAT_ID' => 17,
    'DISK_ID' => 112,
), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	Chat ID not passed
FILE_ID_EMPTY	File ID not passed

im.disk.file.save

Saving the file to your Bitrix24.Disk

Revision: 18

Note! Method specified using function restCommand is a method for sending data to Bitrix24, this method is available in EchoBot ***example , and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.***

Options

Parameter	Example	Required	Description	Revision
DISK_ID	112	**Yes**	Identifier file	21

Method call and response

JavaScript

```
BX24.callMethod('im.disk.file.save', {
```

```
  'DISK_ID': 112,
}, function(result){
  if(result.error())
  {
    console.error(result.error().ex);
  }
  else
  {
    console.log(result.data());
  }
});
```

PHP

```
$result = restCommand('im.disk.file.save', Array(
  'DISK_ID' => 112,
), $_REQUEST["auth"]);
```

Response Example

```
{
  "result": {
    "folder": {"id": 130, "name": "Saved files"},
    "file": {"id": 578, "name": "image.png"}
  }
}
```

Example response when an error occurs

```
{
  "error": "FILE_ID_EMPTY",
  "error_description": "File ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
FILE_SAVE_ERROR	File could not be saved
FILE_ID_EMPTY	File ID not passed

Working with users

Note! All methods are specified using the function restCommand is a method for sending data to Bitrix24, this method is available in EchoBot ***example , and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.***

Method	Description of the method
im.user.get	Getting user data.
im.user.list.get	Getting data about users.
im.user.business.list	Getting a list of business users.
im.user.status.get	Getting information about set user status.
im.user.status.set	Setting the user's status.
im.user.status.idle.start	Setting automatic status "Departed."
im.user.status.idle.end	Disabling automatic status "Departed."

im.user.get

Getting user data

Revision: 18

Note! All methods are specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24- php-sdk.

Options

Parameter	Example	Required	Description	Revision
ID	5	No	Identifier user	18
AVATAR_HR	N	No	Generate high resolution avatar	18

- If the ID key is not passed, the data for the current user.

Method call and response

JavaScript

```
BX24.callMethod('im.user.get', {ID: 5}, function(result){
   if(result.error())
   {
      console.error(result.error().ex);
   }
   else
   {
      console.log(result.data());
   }
```

```
});
```

PHP

```
$result = restCommand('im.user.get', Array( 'ID' => 5,


), $_REQUEST["auth"]);
```

Response Example


```
{
    "result": {
        "id": 5,
        "name": "Evgeny Shelenkov",
        "first_name": "Eugene",
        "last_name": "Shelenkov",
        "work_position": "",
        "color": "#df532d",
        "avatar": "http://192.168.2.232/upload/resize_cache/main/1d3/100_100_2/shelenkov.png",
        "gender": "M",
        "birthday": "",
        "extranet": false,
        "network": false,
        "bot": false,
        "connector": false,
        "external_auth_id": "default",
        "status": "online",
        "idle": false,
        "last_activity_date": "2018-01-29T17:35:31+03:00",
        "desktop_last_date": false,
        "mobile_last_date": false,
        "departments": [
          50
        ],
        "absent": false,
        "phones": {
          "work_phone": "",
          "personal_mobile": "",
          "personal_phone": ""
        }
```

```
    }
}
```

Description of keys:

- id -- user ID
- name -- user's first and last name
- first_name -- username
- last_name -- user last name
- work_position -- position
- color -- user color in hex format
- avatar -- link to the avatar (if empty, no avatar has been set)
- avatar_hr -- link to a high-res avatar (available only when requested with the AVATAR_HR = 'Y' parameter)
- gender -- user's gender
- birthday -- user's birthday in DD-MM format, if empty -- not set
- extranet -- sign of an external extranet user (true/false)
- network -- attribute of Bitrix24.Network user (true/false)
- bot -- sign of a bot (true/false)
- connector -- indication of the user of open lines (true/false)
- external_auth_id -- external authorization code
- status -- selected user status
- idle -- date the user left the computer, in ATOM format (false if not set)
- last_activity_date -- date of the user's last activity in ATOM format
- mobile_last_date -- date of the last action in the mobile application in ATOM format (if not set, false)
- departments -- department identifiers
- desktop_last_date -- date of the last action in the desktop application in ATOM format (false if not set)

- absent -- date by which date the user has a vacation, in ATOM format (if not set, false)
- phones -- array of phone numbers: work_phone -- work phone, personal_mobile -- mobile phone, personal_phone -- home phone

Example response when an error occurs

```
{
    "error": "ID_EMPTY",
    "error_description": "User ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
ID_EMPTY	User ID not passed
USER_NOT_EXISTS	The user with the specified id is not found
ACCESS_DENIED	The current user does not have data access rights

im.user.list.get

Getting data about users

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can

use your own function or javascript method BX24.callMethod or bitrix24- php-sdk.

Options

Parameter	Example	Required	Description	Revision
ID	[4,5]	**Yes**	User IDs	19
AVATAR_HR N		No	Generate high resolution avatar	19

- If the ID key is not passed, the data for the current user.

Method call and response

JavaScript

```
BX24.callMethod('im.user.list.get', {ID: [4,5]}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.user.list.get', Array(
    'ID' => [4,5],
), $_REQUEST["auth"]);
```

Response Example


```
{
    "result": {
        4: null,
        5: {
            "id": 5,
            "name": "Evgeny Shelenkov",
            "first_name": "Eugene",
            "last_name": "Shelenkov",
            "work_position": "",
            "color": "#df532d",
            "avatar": "http://192.168.2.232/upload/resize_cache/main/1d3/100_100_2/shelenkov.png",
            "gender": "M",
            "birthday": "",
            "extranet": false,
            "network": false,
            "bot": false,
            "connector": false,
            "external_auth_id": "default",
            "status": "online",
            "idle": false,
            "last_activity_date": "2018-01-29T17:35:31+03:00",
            "desktop_last_date": false,
            "mobile_last_date": false,
            "departments": [
                50
            ],
            "absent": false,
            "phones": {
                "work_phone": "",
                "personal_mobile": "",
                "personal_phone": ""
            }
        }
    }
}
```

Description of keys:

- id -- user ID
- name -- user's first and last name
- first_name -- username
- last_name -- user last name
- work_position -- position
- color -- user color in hex format
- avatar -- link to the avatar (if empty, no avatar has been set)
- avatar_hr -- link to a high-res avatar (available only when requested with the AVATAR_HR = 'Y' parameter)
- gender -- user's gender
- birthday -- user's birthday in DD-MM format, if empty -- not set
- extranet -- sign of an external extranet user (true/false)
- network -- attribute of Bitrix24.Network user (true/false)
- bot -- sign of a bot (true/false)
- connector -- indication of the user of open lines (true/false)
- external_auth_id -- external authorization code
- status -- selected user status
- idle -- date the user left the computer, in ATOM format (false if not set)
- last_activity_date -- date of the user's last activity in ATOM format
- mobile_last_date -- date of the last action in the mobile application in ATOM format (if not set, false)
- departments -- department identifiers
- desktop_last_date -- date of the last action in the desktop application in ATOM format (false if not set)
- absent -- date by which date the user has a vacation, in ATOM format (if not set, false)

- phones -- array of phone numbers: work_phone -- work phone, personal_mobile -- mobile phone, personal_phone -- home phone

Example response when an error occurs

```
{
    "error": "INVALID_FORMAT",
    "error_description": "A wrong format for the ID field is passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
INVALID_FORMAT	User ID not passed

im.user.business.list

Get a list of business users

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24- php-sdk.

Options

Parameter	Example	Required	Description	Revision
USER_DATA	N	No	Upload information about	19

			users	
OFFSET	0	No	User sampling bias	19
LIMIT	10	No	User sampling limit	19

- If the USER_DATA = Y parameter is passed, then in the response, instead of an array of identifiers, an array of objects with information about user.
- If business users are not available at the current rate, a false response will be returned.
- The method supports the standard Bitrix24 Rest Api pagination , but in addition to it, it is possible to build navigation using the OFFSET and LIMIT parameters.

Method call and response

JavaScript

```
BX24.callMethod('im.user.business.list', {USER_DATA: 'Y'}, function(result){
    if(result.error()) {

        console.error(result.error().ex);
    }
    else
    {
        console.log('users', result.data());
        console.log('total', result.total());
    }
});
```

PHP

```
$result = restCommand('im.user.business.list', Array(
```

```
    'USER_DATA' => 'Y'
), $_REQUEST["auth"]);
```

Response Example

With options USER_DATA = N:


```
{
    "result": [1],
    "total": 1
}
```


With options USER_DATA = Y:


```
{
    "result": [
        {
            "id": 1,
            "name": "Evgeny Shelenkov",
            "first_name": "Eugene",
            "last_name": "Shelenkov",
            "work_position": "",
            "color": "#df532d",
            "avatar": "http://192.168.2.232/upload/resize_cache/main/1d3/100_100_2/shelenkov.png",
            "gender": "M",
            "birthday": "",
            "extranet": false,
            "network": false,
            "bot": false,
            "connector": false,
            "external_auth_id": "default",
            "status": "online",
            "idle": false,
            "last_activity_date": "2018-01-29T17:35:31+03:00",
            "desktop_last_date": false,
            "mobile_last_date": false,
            "departments": [
              50
            ],
            "absent": false,
```

```
            "phones": {
              "work_phone": "",
              "personal_mobile": "",
              "personal_phone": ""
            }
        }
    ],
    "total": 1
}
```


Description of keys:

- id -- user ID
- name -- user's first and last name
- first_name -- username
- last_name -- user last name
- work_position -- position
- color -- user color in hex format
- avatar -- link to the avatar (if empty, no avatar has been set)
- avatar_hr -- link to a high-res avatar (available only when requested with the AVATAR_HR = 'Y' parameter)
- gender -- user's gender
- birthday -- user's birthday in DD-MM format, if empty -- not set
- extranet -- sign of an external extranet user (true/false)
- network -- attribute of Bitrix24.Network user (true/false)
- bot -- sign of a bot (true/false)
- connector -- indication of the user of open lines (true/false)
- external_auth_id -- external authorization code
- status -- selected user status
- idle -- date the user left the computer, in ATOM format (false if not set)

- last_activity_date -- date of the user's last activity in ATOM format
- mobile_last_date -- date of the last action in the mobile application in ATOM format (if not set, false)
- departments -- department identifiers
- desktop_last_date -- date of the last action in the desktop application in ATOM format (false if not set)
- absent -- date by which date the user has a vacation, in ATOM format (if not set, false)
- phones -- array of phone numbers: work_phone -- work phone, personal_mobile -- mobile phone, personal_phone -- home phone

im.user.status.get

Getting information about installed user status

Revision: 18

Options

No

Method call

JavaScript

```
BX24.callMethod('im.user.status.get', {}, function(result){
   if(result.error())
   {
      console.error(result.error().ex);
   }
   else
   {
      console.log(result.data());
   }
});
```

PHP

```
$result = restCommand('im.user.status.get', Array(
```

```
), $_REQUEST["auth"]);
```

Response Example

```
{
    "result": "dnd"
}
```

Description of the result:

- online -- Online
- dnd -- do not disturb
- away -- absent

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24- php-sdk.

im.user.status.set

Setting user status

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24- php-sdk.

Options

Parameter	Example	Required	Description	Revision
STATUS	online	**Yes**	New status user	18

The following statuses are available:

- online -- Online
- dnd -- do not disturb
- away -- absent

Method call and response

JavaScript

```
BX24.callMethod('im.user.status.set', {}, function(result){
    if(result.error()) {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.user.status.set', Array( ),
$_REQUEST["auth"]);
```

Response Example

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

im.user.status.idle.start

Setting the automatic status "Away"

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can

use your own function or javascript method BX24.callMethod or bitrix24- php-sdk.

Options

Parameter	Example	Required	Description	Revision
AGO	10	No	How many minutes ago did you leave	18

Method call and response

JavaScript

```
BX24.callMethod('im.user.status.idle.start', {
    'AGO': 10
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.user.status.idle.start', Array(
    'AGO' => 10
), $_REQUEST["auth"]);
```

Response Example

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

im.user.status.idle.end

Disabling automatic status "Away"

Revision: 18

Options

No

Method call

JavaScript

```
BX24.callMethod('im.user.status.idle.end', {}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.user.status.idle.start', Array( ),
$_REQUEST["auth"]);
```

Response Example

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

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24- php-sdk.

Working with departments

Note! All methods are specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24- php-sdk.

Method	Description of the method
im.department.get	Retrieve department information.
im.department.colleagues.list	Getting a list of users in your department.
im.department.managers.get	Getting a list of department heads.
im.department.employees.get	Getting a list of employees in subdivision.

im.department.get

Retrieving department data

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24- php-sdk.

Options

Parameter	Example	Required	Description	Revision
ID	[51]	**Yes**	Department ID	18
USER_DATA N		No	Upload information about users	18

- If the previous parameter USER_DATA = Y, then the result will be the support data and the manager.

Method call and response

JavaScript

```
BX24.callMethod('im.department.get', {ID: [51]}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.department.get', Array(
    'ID' => [51],
), $_REQUEST["auth"]);
```

Response Example

```
{
    "result": [
        {
            "id": 51,
            "name": "Moscow branch",
```

```
        "full_name": "Moscow branch / Bitrix"
        "manager_user_id": 11,
      }
   ]
}
```


Description of keys:

- id -- department ID
- name -- subdivision short name
- full_name -- full name of the department
- manager_user_data -- manager data description object (not available if USER_DATA != 'Y'):
  - id -- user ID
  - name -- user's first and last name
  - first_name -- username
  - last_name -- user last name
  - work_position -- position
  - color -- user color in hex format
  - avatar -- link to the avatar (if empty, no avatar has been set)
  - gender -- user's gender
  - birthday -- user's birthday in DD-MM format, if empty -- not set
  - extranet -- sign of an external extranet user (true/false)
  - network -- attribute of Bitrix24.Network user (true/false)
  - bot -- sign of a bot (true/false)
  - connector -- indication of the user of open lines (true/false)
  - external_auth_id -- external authorization code
  - status -- selected user status
  - idle -- date the user left the computer, in ATOM format (false if not set)

- last_activity_date -- date of the user's last activity in ATOM format
- mobile_last_date -- date of the last action in the mobile application in ATOM format (if not set, false)
- absent -- date by which date the user has a vacation, in ATOM format (if not set, false)

Example response when an error occurs

```
{
    "error": "INVALID_FORMAT",
    "error_description": "A wrong format for the ID field is passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
INVALID_FORMAT	Invalid identifier format passed

im.department.colleagues.list

Getting a list of users in your department

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24- php-sdk.

Options

Parameter	Example	Required	Description	Revision
USER_DATA N		No	Upload information about users	19
OFFSET	0	No	User sampling bias	19
LIMIT	10	No	User sampling limit	19

- If the USER_DATA = Y parameter is passed, then in the response, instead of an array of identifiers, an array of objects with information about user.
- The method supports the standard Bitrix24 Rest Api pagination , but in addition to it, it is possible to build navigation using the OFFSET and LIMIT parameters.

Method call and response

JavaScript

```
BX24.callMethod('im.department.colleagues.list', {USER_DATA: 'Y'}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log('users', result.data());
        console.log('total', result.total());
    }
});
```

PHP

```
$result = restCommand('im.department.colleagues.list', Array(
```

```
    'USER_DATA' => 'Y'
), $_REQUEST["auth"]);
```

Response Example

With options USER_DATA = N:

```
{
    "result": [1],
    "total": 1
}
```

With options USER_DATA = Y:

```
{
    "result": [
        {
            "id": 1,
            "name": "Evgeny Shelenkov",
            "first_name": "Eugene",
            "last_name": "Shelenkov",
            "work_position": "",
            "color": "#df532d",
            "avatar": "http://192.168.2.232/upload/resize_cache/main/1d3/100_100_2/shelenkov.png",
            "gender": "M",
            "birthday": "",
            "extranet": false,
            "network": false,
            "bot": false,
            "connector": false,
            "external_auth_id": "default",
            "status": "online",
            "idle": false,
            "last_activity_date": "2018-01-29T17:35:31+03:00",
            "desktop_last_date": false,
            "mobile_last_date": false,
            "departments": [
              50
            ],
            "absent": false,
```

```
            "phones": {
              "work_phone": "",
              "personal_mobile": "",
              "personal_phone": ""
            }
        }
    ],
    "total": 1
}
```


Description of keys:

- id -- user ID
- name -- user's first and last name
- first_name -- username
- last_name -- user last name
- work_position -- position
- color -- user color in hex format
- avatar -- link to the avatar (if empty, no avatar has been set)
- gender -- user's gender
- birthday -- user's birthday in DD-MM format, if empty -- not set
- extranet -- sign of an external extranet user (true/false)
- network -- attribute of Bitrix24.Network user (true/false)
- bot -- sign of a bot (true/false)
- connector -- indication of the user of open lines (true/false)
- external_auth_id -- external authorization code
- status -- selected user status
- idle -- date the user left the computer, in ATOM format (false if not set)
- last_activity_date -- date of the user's last activity in ATOM format

- mobile_last_date -- date of the last action in the mobile application in ATOM format (if not set, false)
- desktop_last_date -- date of the last action in the desktop application in ATOM format (false if not set)
- absent -- date by which date the user has a vacation, in ATOM format (if not set, false)
- phones -- array of phone numbers: work_phone -- work phone, personal_mobile -- mobile phone, personal_phone -- home phone

im.department.managers.get

Getting a list of department heads

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24- php-sdk.

Options

Parameter	Example	Required	Description	Revision
ID	[105]	**Yes**	Department IDs	19
USER_DATA N		No	Upload information about users	19

- If the USER_DATA = Y parameter is passed, then in the response, instead of an array of identifiers, an array of objects with information about user.

Method call and response

JavaScript

```
BX24.callMethod('im.department.managers.get', {USER_DATA: 'Y'}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log('users', result.data());
    }
});
```

PHP

```
$result = restCommand('im.department.managers.get', Array(
    'USER_DATA' => 'Y'
), $_REQUEST["auth"]);
```

Response Example

With options USER_DATA = N:

```
{
    "result": {
        105: [1]
    },
}
```

With options USER_DATA = Y:

```
{
    "result": {
        105: {
            "id": 1,
            "name": "Evgeny Shelenkov",
            "first_name": "Eugene",
            "last_name": "Shelenkov",
            "work_position": "",
            "color": "#df532d",
            "avatar": "http://192.168.2.232/upload/resize_cache/main/1d3/100_100_2/shelenkov.png",
```

```
            "gender": "M",
            "birthday": "",
            "extranet": false,
            "network": false,
            "bot": false,
            "connector": false,
            "external_auth_id": "default",
            "status": "online",
            "idle": false,
            "last_activity_date": "2018-01-29T17:35:31+03:00",
            "desktop_last_date": false,
            "mobile_last_date": false,
            "departments": [
              50
            ],
            "absent": false,
            "phones": {
              "work_phone": "",
              "personal_mobile": "",
              "personal_phone": ""
            }
        }
    }
}
```


Description of keys:

- id -- user ID
- name -- user's first and last name
- first_name -- username
- last_name -- user last name
- work_position -- position
- color -- user color in hex format
- avatar -- link to the avatar (if empty, no avatar has been set)
- gender -- user's gender

- birthday -- user's birthday in DD-MM format, if empty -- not set
- extranet -- sign of an external extranet user (true/false)
- network -- attribute of Bitrix24.Network user (true/false)
- bot -- sign of a bot (true/false)
- connector -- indication of the user of open lines (true/false)
- external_auth_id -- external authorization code
- status -- selected user status
- idle -- date the user left the computer, in ATOM format (false if not set)
- last_activity_date -- date of the user's last activity in ATOM format
- mobile_last_date -- date of the last action in the mobile application in ATOM format (if not set, false)
- desktop_last_date -- date of the last action in the desktop application in ATOM format (false if not set)
- absent -- date by which date the user has a vacation, in ATOM format (if not set, false)
- phones -- array of phone numbers: work_phone -- work phone, personal_mobile -- mobile phone, personal_phone -- home phone

Example response when an error occurs

```
{
    "error": "ID_EMPTY",
    "error_description": "Department ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description

ID_EMPTY No list of identifiers passed

im.department.employees.get

Getting a list of employees in a department

Revision: 18

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24- php-sdk.

Options

Parameter Example	Required	Description

Revision				
ID	[105]	**Yes**	Department IDs	19
USER_DATA N		No	Upload user data	19

- If the USER_DATA = Y parameter is passed, then in the response instead of an array identifiers will be passed an array of objects with information about user.

Method call and response

JavaScript

```
BX24.callMethod('im.department.employees.get', {USER_DATA: 'Y'}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log('users', result.data());
    }
});
```

PHP

```
$result = restCommand('im.department.employees.get', Array(
    'USER_DATA' => 'Y'
), $_REQUEST["auth"]);
```

Response Example

With options USER_DATA = N:

```
{
    "result": {
        105: [1]
    }
}
```

With options USER_DATA = Y:

```
{
    "result": {
```

```
        105: {
            "id": 1,
            "name": "Evgeny Shelenkov",
            "first_name": "Eugene",
            "last_name": "Shelenkov",
            "work_position": "",
            "color": "#df532d",
            "avatar": "http://192.168.2.232/upload/resize_cache/main/1d3/100_100_2/shelenkov.png",
            "gender": "M",
            "birthday": "",
            "extranet": false,
            "network": false,
            "bot": false,
            "connector": false,
            "external_auth_id": "default",
            "status": "online",
            "idle": false,
            "last_activity_date": "2018-01-29T17:35:31+03:00",
            "desktop_last_date": false,
            "mobile_last_date": false,
            "departments": [
              50
            ],
            "absent": false,
            "phones": {
              "work_phone": "",
              "personal_mobile": "",
              "personal_phone": ""
            }
        }
    }
}
```


Description of keys:

- id -- user ID
- name -- user's first and last name
- first_name -- username
- last_name -- user last name
- work_position -- position

- color -- user color in hex format
- avatar -- link to the avatar (if empty, no avatar has been set)
- gender -- user's gender
- birthday -- user's birthday in DD-MM format, if empty -- not set
- extranet -- sign of an external extranet user (true/false)
- network -- attribute of Bitrix24.Network user (true/false)
- bot -- sign of a bot (true/false)
- connector -- indication of the user of open lines (true/false)
- external_auth_id -- external authorization code
- status -- selected user status
- idle -- date the user left the computer, in ATOM format (false if not set)
- last_activity_date -- date of the user's last activity in ATOM format
- mobile_last_date -- date of the last action in the mobile application in ATOM format (if not set, false)
- desktop_last_date -- date of the last action in the desktop application in ATOM format (false if not set)
- absent -- date by which date the user has a vacation, in ATOM format (if not set, false)
- phones -- array of phone numbers: work_phone -- work phone, personal_mobile -- mobile phone, personal_phone -- home phone

Example response when an error occurs

```
{
    "error": "ID_EMPTY",
    "error_description": "Department ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
ID_EMPTY	No list of identifiers passed

Working with search

Note! All methods are specified using the restCommand function - this is a method for sending data to Bitrix24, this method is in the EchoBot example, And presented here as an example. You can use your function or javascript-ÿÿÿÿÿ BX24.callMethod or bitrix24-php-sdk.

Method	Description of the method
im.search.user.list	User search.
im.search.chat.list	Chat search.
im.search.department.list	Search for divisions.
im.search.last.get	Getting a list of elements last search.
im.search.last.add	Adding a story element last search.
im.search.last.delete	Deleting a history item last search.

im.search.user.list

User search

Revision: 18

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, And presented here as an example. You can use your function or javascript-ÿÿÿÿÿ BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
FIND	Eugene	**Yes**	Search phrase	19
BUSINESS	N	No	Search among business users	19
AVATAR_HR	N	No	Generate high resolution avatar	19
OFFSET	0	No	User sampling bias	19
LIMIT	10	No	User sampling limit	19

- The search is carried out on the following fields: **First name, Last name, Position, Department.**
- The method supports the standard Bitrix24 Rest Api pagination , but in addition to it, it is possible to build navigation using the OFFSET and LIMIT parameters.

Method call and response

JavaScript

```
BX24.callMethod('im.search.user.list', {
    FIND: 'Eugene'
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log('users', result.data());
        console.log('total', result.total());
    }
```

```
});
```

PHP

```
$result = restCommand('im.search.user.list', Array(
    'FIND' => 'Eugene'
), $_REQUEST["auth"]);
```

Response Example


```
{
    "result": {
        1: {
            "id": 1,
            "name": "Evgeny Shelenkov",
            "first_name": "Eugene",
            "last_name": "Shelenkov",
            "work_position": "",
            "color": "#df532d",
            "avatar": "http://192.168.2.232/upload/resize_cache/main/1d3/100_100_2/shelenkov.png",
            "gender": "M",
            "birthday": "",
            "extranet": false,
            "network": false,
            "bot": false,
            "connector": false,
            "external_auth_id": "default",
            "status": "online",
            "idle": false,
            "last_activity_date": "2018-01-29T17:35:31+03:00",
            "desktop_last_date": false,
            "mobile_last_date": false,
            "departments": [
              50
            ],
            "absent": false
        }
    },
    "total": 1
}
```


Description of keys:

- id -- user ID
- name -- user's first and last name
- first_name -- username
- last_name -- user last name
- work_position -- position
- color -- user color in hex format
- avatar -- link to the avatar (if empty, no avatar has been set)
- avatar_hr -- link to a high-res avatar (available only when requested with the AVATAR_HR = 'Y' parameter)
- gender -- user's gender
- birthday -- user's birthday in DD-MM format, if empty -- not set
- extranet -- sign of an external extranet user (true/false)
- network -- attribute of Bitrix24.Network user (true/false)
- bot -- sign of a bot (true/false)
- connector -- indication of the user of open lines (true/false)
- external_auth_id -- external authorization code
- status -- selected user status
- idle -- date the user left the computer, in ATOM format (false if not set)
- last_activity_date -- date of the user's last activity in ATOM format
- mobile_last_date -- date of the last action in the mobile application in ATOM format (if not set, false)
- desktop_last_date -- date of the last action in the desktop application in ATOM format (false if not set)
- absent -- date by which date the user has a vacation, in ATOM format (if not set, false)

Example response when an error occurs

```
{
    "error": "FIND_SHORT",
    "error_description": "Too short a search phrase."
}
```

Description of keys:

- error -- code of the error that occurred

- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
FIND_SHORT	Too short search phrase, search carried out from three characters.

im.search.chat.list

Chat Search

Revision: 18

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, And presented here as an example. You can use your function or javascript-ÿÿÿÿÿ BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
FIND	Mint	**Yes**	Search phrase	19
OFFSET	0	No	Sample bias users	19
LIMIT	10	No	Sample limit users	19

- The search is carried out on the following fields: **Title, First Name** and **Last Name** chat participants.
- The method supports the standard Bitrix24 Rest API pagination , but in addition to it, it is possible to build navigation using OFFSET and LIMIT parameters .

Method call and response

JavaScript

```
BX24.callMethod('im.search.chat.list', {
    FIND: 'Mint'
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log('users', result.data());
        console.log('total', result.total());
    }
});
```

PHP

```
$result = restCommand('im.search.chat.list', Array(
    'FIND' => 'Mint'
), $_REQUEST["auth"]);
```

Response Example


```
{
    "result": {
        21191: {
            "id": 21191,
            "title": "Mint Chat #3",
            "owner": 2,
            "extranet": false,
            "avatar": "",
            "color": "#4ba984",
            "type": "chat",
            "entity_type": "",
            "entity_data_1": "",
            "entity_data_2": "",
            "entity_data_3": "",
            "date_create": "2017-10-14T12:15:32+02:00",
            "message_type": "C"
        }
    },
    "total": 1
}
```


Description of keys:

- id -- chat identifier
- title -- chat name
- owner -- user ID of the chat owner
- color -- chat color in hex format
- avatar -- link to the avatar (if empty, no avatar has been set)
- type -- chat type (group chat, call chat, open line chat, etc.)
- entity_type -- external code for chat -- type
- entity_id -- external code for chat -- identifier
- entity_data_1 -- external data for chat
- entity_data_2 -- external data for chat
- entity_data_3 -- external data for chat
- date_create -- chat creation date in ATOM format
- message_type -- chat message type

Example response when an error occurs

```
{
    "error": "FIND_SHORT",
    "error_description": "Too short a search phrase."
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
FIND_SHORT	The search phrase is too short, the search is performed from three characters.

im.search.department.list

Search for divisions

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, And presented here as an example. You can use your function or javascript-ÿÿÿÿÿ BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
FIND	Moscow	**Yes**	Search phrase	19
USER_DATA N		No	Upload data about users	19
OFFSET	0	No	Sample bias users	19
LIMIT	10	No	Sample limit users	19

- If the previous parameter USER_DATA = Y, then the result will be the backup data leader.
- The search is carried out on the following fields: **Full name divisions.**
- The method supports the standard Bitrix24 Rest API pagination , but in addition to it, it is possible to build navigation using OFFSET and LIMIT parameters .

Method call and response

JavaScript

```
BX24.callMethod('im.search.department.list', {
    FIND: 'Moscow'
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log('users', result.data());
```

```
        console.log('total', result.total());
    }
});
```

PHP

```
$result = restCommand('im.search.department.list', Array(
    'FIND' => 'Moscow'
), $_REQUEST["auth"]);
```

Response Example

```
{
    "result": [
        {
            id: 51,
            name: "Moscow branch",
            full_name: "Moscow branch / Bitrix",
            manager_user_id: 11
        }
    ],
    "total": 1
}
```

Description of keys:

- id -- department ID
- name -- subdivision short name
- full_name -- full name of the department
- manager_user_data -- manager data description object (not available if USER_DATA != 'Y')
    - id -- user ID
    - name -- user's first and last name
    - first_name -- username
    - last_name -- user last name
    - work_position -- position
    - color -- user color in hex format
    - avatar -- link to the avatar (if empty, no avatar has been set)
    - gender -- user's gender

- birthday -- user's birthday in DD-MM format, if empty -- not set
- extranet -- sign of an external extranet user (true/false)
- network -- attribute of Bitrix24.Network user (true/false)
- bot -- sign of a bot (true/false)
- connector -- indication of the user of open lines (true/false)
- external_auth_id -- external authorization code
- status -- selected user status
- idle -- date the user left the computer, in ATOM format (false if not set)
- last_activity_date -- date of the user's last activity in ATOM format
- mobile_last_date -- date of the last action in the mobile application in ATOM format (if not set, false)
- absent -- date by which date the user has a vacation, in ATOM format (if not set, false)

Example response when an error occurs

```
{
    "error": "FIND_SHORT",
    "error_description": "Too short a search phrase."
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
FIND_SHORT	The search phrase is too short, the search is performed from three characters.

im.search.last.get

Getting a list of last search items

Revision: 18

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, And presented here as an example. You can use your function or javascript-ÿÿÿÿÿ BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
SKIP_OPENLINES	N	No	Skip chats open lines	18
SKIP_CHAT	N	No	Skip chats	18
OFFSET	N	No	Miss one-on-one dialogues one	18

Method call and response

JavaScript

```
BX24.callMethod('im.search.last.get', {}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.user.business.list', Array(
), $_REQUEST["auth"]);
{
    "result": [
        {
            "id": 1,
```

```
                "type": "user",
                "title": "Evgeny Shelenkov",
                "avatar": {
                    "url": "http://192.168.2.232/upload/resize_cache/main/1d3/100_100_2/shelenkov.png",
                    "color": "#df532d"
                }
                "user": {
                    "id": 1,
                    "name": "Evgeny Shelenkov",
                    "first_name": "Eugene",
                    "last_name": "Shelenkov",
                    "work_position": "",
                    "color": "#df532d",
                    "avatar":
"http://192.168.2.232/upload/resize_cache/main/1d3/100_100_2/shelenkov.png",
                    "gender": "M",
                    "birthday": "",
                    "extranet": false,
                    "network": false,
                    "bot": false,
                    "connector": false,
                    "external_auth_id": "default",
                    "status": "online",
                    "idle": false,
                    "last_activity_date": "2018-01-29T17:35:31+03:00",
                    "desktop_last_date": false,
                    "mobile_last_date": false,
                    "departments": [
                      50
                    ],
                    "absent": false,
                    "phones": {
                      "work_phone": "",
                      "personal_mobile": "",
                      "personal_phone": ""
                    }
                }
            }
        ]
    }
```


Description of keys:

- id -- dialog ID (number if user, chatXXX if chat)
- name -- post type (user -- if user, chat -- if chat)
- avatar -- post avatar description object:
    - url -- link to the avatar (if empty, then the avatar is not set)
    - color -- dialog color in hex format
- title -- record title
- user -- user data description object (not available if post type is chat):
    - id -- user ID
    - name -- user's first and last name
    - first_name -- username
    - last_name -- user last name
    - work_position -- position
    - color -- user color in hex format
    - avatar -- link to the avatar (if empty, no avatar has been set)
    - gender -- user's gender
    - birthday -- user's birthday in DD-MM format, if empty -- not set
    - extranet -- sign of an external extranet user (true/false)
    - network -- attribute of Bitrix24.Network user (true/false)
    - bot -- sign of a bot (true/false)
    - connector -- indication of the user of open lines (true/false)
    - external_auth_id -- external authorization code
    - status -- selected user status
    - idle -- date the user left the computer, in ATOM format (false if not set)
    - last_activity_date -- date of the user's last activity in ATOM format
    - mobile_last_date -- date of the last action in the mobile application in ATOM format (if not set, false)
    - absent -- date by which date the user has a vacation, in ATOM format (if not set, false)
- chat -- chat data description object (not available if post type is user):

- id -- chat identifier
- title -- chat name
- owner -- user ID of the chat owner
- extranet -- sign of participation in the chat of an external extranet user (true/false)
- color -- chat color in hex format
- avatar -- link to the avatar (if empty, no avatar has been set)
- type -- chat type (group chat, call chat, open line chat, etc.)
- entity_type -- external code for chat -- type
- entity_id -- external code for chat -- identifier
- entity_data_1 -- external data for chat
- entity_data_2 -- external data for chat
- entity_data_3 -- external data for chat
- date_create -- chat creation date in ATOM format
- message_type -- chat message type

im.search.last.add

Adding a last search history item

Revision: 18

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
DIALOG_ID	chat17 or 256	**Yes**	Dialog ID. Format: **chatXXX** -- the recipient's chat if the message is for chat, or **XXX** -- the recipient's ID if the message is for private dialogue	18

Method call and response

JavaScript

```
BX24.callMethod('im.search.last.add', {
    'DIALOG_ID': 'chat17'
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.search.last.add', Array(
    'DIALOG_ID' => 'chat17'
), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
    "error": "DIALOG_ID_EMPTY",
    "error_description": "Dialog ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description

DIALOG_ID_EMPTY No dialog ID passed.

im.search.last.delete

Removing a Last Search History Item

Revision: 18

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
DIALOG_ID	chat17 or 256	**Yes**	Dialog ID. Format: **chatXXX** -- the recipient's chat if the message is for chat, or **XXX** -- the recipient's ID if the message is for private dialogue	18

Method call and response

JavaScript

```
BX24.callMethod('im.search.last.delete', {
    'DIALOG_ID': 'chat17'
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
```

```
    }
});
```

PHP

```
$result = restCommand('im.search.last.delete', Array(
    'DIALOG_ID' => 'chat17'
), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
    "error": "DIALOG_ID_EMPTY",
    "error_description": "Dialog ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
DIALOG_ID_EMPTY	No dialog ID passed.

Working with the list of recent chats

Method	Description of the method
im.recent.get	List of the user's recent conversations.
im.recent.pin	Fixing the dialogue in favorites.

im.recent.hide	Remove a conversation from the list of recent chats.

Note! All of the above methods are specified using restCommand functions is a method for sending data to Bitrix24, this method is available in EchoBot ***example , and is presented here as an example. you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.***

Methods specified using javascript method BX24.callMethod:

Method	Description of the method
im.recent.unread	Setting the mark "not read" on a chat or dialogue.
im.recent.list	Getting a list of the user's recent conversations (with pagination support).

im.recent.get

List of recent user conversations

Revision: 18

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, And presented here as an example. You can use your function or javascript-ÿÿÿÿÿ BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
SKIP_OPENLINES	N	No	Miss open chats lines	18
SKIP_CHAT	N	No	Miss chats	18
SKIP_DIALOG	N	No	Miss dialogues one	18

			one-on-one	
LAST_UPDATE	2019-07-11T10:45:31+02:00	No	Sample limit for minimization transferred data, date in ATOM format	23
ONLY_OPENLINES N		No	Sample only chats open lines	29
LAST_SYNC_DATE	2019-07-11T10:45:31+02:00	No	The date of the previous fetch to load the changes that occurred in the list from this time. Sample returns data is not older than 7 days. Date in ATOM format	29

Method call and response

JavaScript

```
BX24.callMethod('im.recent.get', {
    'SKIP_OPENLINES': 'Y'
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
```

```
});
```

PHP

```
$result = restCommand('im.user.status.idle.start', Array(
    'SKIP_OPENLINES' => 'Y'
), $_REQUEST["auth"]);
```

Response Example


```
{
    "result": [
        {
            "id": "1",
            "type": "user",
            "avatar": {
                "url": "http://www.hazz/upload/resize_cache/main/1af/100_100_2/1464255149.png",
                "color": "#df532d"
            },
            "title": "Evgeny Shelenkov",
            "message": {
                "id": "30468",
                "text": "1",
                "file": false,
                "attach": false,
                "author_id": "1"
            },
            "counter": "3",
            "date": "2017-10-17T11:12:56+02:00",
            "user": {
                "id": "1",
                "name": "Evgeny Shelenkov",
                "first_name": "Eugene",
                "last_name": "Shelenkov",
                "work_position": "IT Specialist",
                "color": "#df532d",
                "avatar": "http://www.hazz/upload/resize_cache/main/1af/100_100_2/1464255149.png",
                "gender": "M",
                "birthday": false,
                "extranet": false,
                "network": false,
                "bot": false,
                "connector": false,
```

```
            "external_auth_id": "default",
            "status": "online",
            "idle": false,
            "last_activity_date": "2017-10-17T11:16:01+02:00",
            "mobile_last_date": "2017-05-26T12:04:58+02:00",
            "absent": "2017-11-01T00:00:00+02:00"
        }
    },
    {
        "id": "chat21191",
        "type": "chat",
        "avatar": {
            "url": "",
            "color": "#4ba984"
        },
        "title": "Mint Chat #3",
        "message": {
            "id": "30467",
            "text": "Permission to update Bitrix24 received from [Attachment]",
            "file": false,
            "attach": true,
            "author_id": "2"
        },
        "counter": "0",
        "date": "2017-10-17T10:38:20+02:00",
        "chat": {
            "id": "21191",
            "title": "Mint Chat #3",
            "owner": "2",
            "extranet": false,
            "avatar": "",
            "color": "#4ba984",
            "type": "chat",
            "entity_type": "",
            "entity_data_1": "",
            "entity_data_2": "",
            "entity_data_3": "",
            "date_create": "2017-10-14T12:15:32+02:00",
            "message_type": "C"
        }
    }
]
```

```
}
```

Description of keys:

- id -- dialog ID (number if user, chatXXX if chat)
- name -- post type (user -- if user, chat -- if chat)
- avatar -- post avatar description object:
  - url -- link to the avatar (if empty, then the avatar is not set)
  - color -- dialog color in hex format
- title -- post title (First name, last name -- for user, chat title -- for chat)
- messages -- message description object:
  - id -- message ID
  - text -- message text (without bb-codes and line breaks)
  - file -- files present (true/false)
  - attach -- attachments present (true/false)
  - author_id -- post author
  - date -- date of the message in ATOM format
- counter -- counter of unread messages
- user -- user data description object (not available if post type is chat):
  - id -- user ID
  - name -- user's first and last name
  - first_name -- username
  - last_name -- user last name
  - work_position -- position
  - color -- user color in hex format
  - avatar -- link to the avatar (if empty, no avatar has been set)
  - gender -- user's gender
  - birthday -- user's birthday in DD-MM format, if empty -- not beat
  - extranet -- sign of an external extranet user (true/false)
  - network -- attribute of Bitrix24.Network user (true/false)
  - bot -- sign of a bot (true/false)

- connector -- indication of the user of open lines (true/false)
- external_auth_id -- external authorization code
- status -- selected user status
- idle -- date the user left the computer, in ATOM format (false if not set)
- last_activity_date -- date of the user's last activity in ATOM format
- mobile_last_date -- date of the last action in the mobile application in ATOM format (if not set, false)
- absent -- date by which date the user has a vacation, in ATOM format (if not set, false)

- chat -- chat data description object (not available if post type is user):
  - id -- chat identifier
  - title -- chat name
  - owner -- user ID of the chat owner
  - extranet -- sign of participation in the chat of an external extranet user (true/false)
  - color -- chat color in hex format
  - avatar -- link to the avatar (if empty, no avatar has been set)
  - type -- chat type (group chat, call chat, open line chat, etc.)
  - entity_type -- external code for chat -- type
  - entity_id -- external code for chat -- identifier
  - entity_data_1 -- external data for chat
  - entity_data_2 -- external data for chat
  - entity_data_3 -- external data for chat
  - date_create -- chat creation date in ATOM format
  - message_type -- chat message type

im.recent.pin

Pinning a conversation to favorites

Revision: 18

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
DIALOG_ID	chat17 or 256	**Yes**	Dialog ID. Format: **chatXXX** -- the recipient's chat if the message is for chat, or **XXX** -- the recipient's ID if the message is for private dialogue	19
PIN	AND	No	Pin or unpin a dialog	19

- If you specify the PIN = N parameter, then the pinned dialog will be unpinned.

Method call and response

JavaScript

```
BX24.callMethod('im.recent.pin', {
    'DIALOG_ID': 'chat17'
    'PIN': 'Y'
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.recent.pin', Array(
    'DIALOG_ID' => 'chat17',
```

```
    'PIN' => 'Y'
), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
    "error": "DIALOG_ID_EMPTY",
    "error_description": "Dialog ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
DIALOG_ID_EMPTY	No dialog ID passed.

im.recent.hide

Removing a Conversation from the Recent Chats List

Revision: 18

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision

DIALOG_ID chat17 or 256		**Yes**	Dialog ID. Format: **chatXXX** -- the recipient's chat if the message is for chat, or **XXX** -- the recipient's ID if the message is for private dialogue	18

Method call and response

JavaScript

```
BX24.callMethod('im.recent.hide', {
    'DIALOG_ID': 'chat17'
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.recent.hide', Array(
    'DIALOG_ID' => 'chat17'
), $_REQUEST["auth"]);
```

Response Example

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

Example response when an error occurs

```
{
    "error": "DIALOG_ID_EMPTY",
    "error_description": "Dialog ID can't be empty"
```

```
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
DIALOG_ID_EMPTY	No dialog ID passed.

im.recent.unread

Setting an "unread" mark on a chat or conversation

Revision: 30

Options

Parameter	Example	Required	Description	Revision
DIALOG_ID	'chat74'	**Yes**	Dialog ID. Format: • **chatXXX** -- chat recipient, if message for chat or • **XXX** -- identifier recipient, if message for private dialogue	30
ACTION	'AND'	**No**	Mark | uncheck "unread" on dialog - 'Y'|'N'	30

Method call and response

JavaScript

```
B24.callMethod(
  'im.recent.unread',
  {
    DIALOG_ID: 'chat74',
    ACTION: 'Y'
  },
  res => {
    if (res.error())
    {
      console.error(result.error().ex);
    }
    else
    {
      console.log(res.data())
    }
  }
)
```

Response Example

```
true // if successful|uncheck mark
```

Example response when an error occurs

```
{
  "error":"DIALOG_ID_EMPTY",
  "error_description":"Dialog ID can\u0027t be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
DIALOG_ID_EMPTY	The DIALOG_ID parameter was not passed or is not

conforms to the format

im.recent.list

Getting a list of the user's recent conversations (with pagination support)

Revision: 30

Options

Parameter	Example	Required	Description	Revision
SKIP_OPENLINES	'AND'	**No**	Miss open chats lines	30
SKIP_DIALOG	'N'	**No**	Miss one-on-one dialogues one"	30
SKIP_CHAT	'AND'	**No**	Miss chats	30
LAST_MESSAGE_DATE	'2021-10-30'	**No**	date from last element previous samples	30

Method call and response

JavaScript

```
B24.callMethod(
  'im.recent.list',
  {
    LAST_MESSAGE_DATE: '2021-10-30'
  },
  res => {
    if (res.error())
```

```
    {
      console.error(result.error().ex);
    }
    else
    {
      console.log(res.data())
    }
  }
)
```

Response Example


```
{
  "items": [
    {
      "id": "chat71",
      "chat_id": 71,
      "type": "chat",
      "avatar": {
        "url": "",
        "color": "#8474c8"
      },
      "title": "Purple Guest #3 - Open Line 2",
      "message": {
        "id": 267,
        "text": "Submitted form \"Contact form for open lines\"
[The attachment]",
        "file": false,
        "author_id": 0,
        "attach": true,
        "date": "2021-10-27T18:20:45+02:00",
        "status": "received"
      },
      "counter": 3,
      "pinned": false,
      "unread": false,
      "date_update": "2021-10-27T18:20:45+02:00",
      "chat": {
        "id": 71,
        "name": "Purple Guest #3 - Open Line 2",
        "owner": 0,
        "extranet": false,
        "avatar": "",
```

```
    "color": "#8474c8",
    "type": "lines",
    "entity_type": "LINES",
    "entity_id": "livechat|2|70|2016",
    "entity_data_1": "N|NONE|0|N|N|7|1635351612|0|0|0",
    "entity_data_2": "",
    "entity_data_3": "",
    "mute_list": [],
    "manager_list": [],
    "date_create": "2021-10-27T18:20:12+02:00",
    "message_type": "L"
  },
  "lines": {
    "id": 7,
    "status": 5,
    "date_create": "2021-10-27T18:20:12+02:00"
  },
  "user": {
    "id": 0
  },
  "options": []
},
{
  "id": "chat69",
  "chat_id": 69,
  "type": "chat",
  "avatar": {
    "url": "",
    "color": "#3e99ce"
  },
  "title": "Blue Guest #3 - Open Line 2",
  "message": {
    "id": 258,
    "text": "Submitted \"Open lines contact form\" [Attachment]",

    "file": false,
    "author_id": 0,
    "attach": true,
```

```
      "date": "2021-10-27T18:16:22+02:00",
      "status": "received"
    },
    "counter": 3,
    "pinned": false,
    "unread": false,
    "date_update": "2021-10-27T18:16:22+02:00",
    "chat": {
      "id": 69,
      "name": "Blue guest #3 - Open line 2",
      "owner": 0,
      "extranet": false,
      "avatar": "",
      "color": "#3e99ce",
      "type": "lines",
      "entity_type": "LINES",
      "entity_id": "livechat|2|68|2015",
      "entity_data_1": "N|NONE|0|N|N|6|1635351343|0|0|0",
      "entity_data_2": "",
      "entity_data_3": "",
      "mute_list": [],
      "manager_list": [],
      "date_create": "2021-10-27T18:15:43+02:00",
      "message_type": "L"
    },
    "lines": {
      "id": 6,
      "status": 5,
      "date_create": "2021-10-27T18:15:43+02:00"
    },
    "user": {
      "id": 0
    },
    "options": []
  },
  {
    "id": "chat67",
    "chat_id": 67,
    "type": "chat",
    "avatar": {
      "url": "",
      "color": "#29619b"
```

```
},
 "title": "Azure Guest #2 - Open Line 2",
 "message": {
  "id": 250,
  "text": "hi",
  "file": false,
  "author_id": 2014,
  "attach": false,
  "date": "2021-10-27T17:11:36+02:00",
  "status": "received"
},
 "counter": 5,
 "pinned": false,
 "unread": false,
 "date_update": "2021-10-27T17:11:36+02:00",
 "chat": {
  "id": 67,
  "name": "Azure Guest #2 - Open Line 2",
  "owner": 0,
  "extranet": false,
  "avatar": "",
  "color": "#29619b",
  "type": "lines",
  "entity_type": "LINES",
  "entity_id": "livechat|2|66|2014",
  "entity_data_1": "N|NONE|0|N|N|5|1635346774|0|0|0",
  "entity_data_2": "",
  "entity_data_3": "",
  "mute_list": [],
  "manager_list": [],
  "date_create": "2021-10-27T16:59:34+02:00",
  "message_type": "L"
},
 "lines": {
  "id": 5,
  "status": 5,
  "date_create": "2021-10-27T16:59:34+02:00"
},
 "user": {
  "id": 2014,
  "active": true,
  "name": "Guest",
```

```
    "first_name": "Guest",
    "last_name": "",
    "work_position": "",
    "color": "#4ba5c3",
    "avatar": "",
    "gender": "M",
    "birthday": "",
    "extranet": true,
    "network": false,
    "bot": false,
    "connector": true,
    "external_auth_id": "imconnector",
    "status": "online",
    "idle": false,
    "last_activity_date": "2021-10-27T17:11:36+02:00",
    "mobile_last_date": false,
    "absent": false,
    "departments": [],
    "phones": false,
    "desktop_last_date": false
  },
  "options": []
},
{
  "id": "chat65",
  "chat_id": 65,
  "type": "chat",
  "avatar": {
    "url": "",
    "color": "#df532d"
  },
  "title": "Maxim Tester - Open Line 2",
  "message": {
    "id": 225,
    "text": "New deal [Attachment] created",
    "file": false,
    "author_id": 0,
    "attach": true,
    "date": "2021-10-25T14:48:28+02:00",
    "status": "received"
  },
  "counter": 6,
```

```
    "pinned": false,
    "unread": false,
    "date_update": "2021-10-25T14:48:28+02:00",
    "chat": {
      "id": 65,
      "name": "Maxim Tester - Open line 2",
      "owner": 0,
      "extranet": false,
      "avatar": "",
      "color": "#df532d",
      "type": "lines",
      "entity_type": "LINES",
      "entity_id": "livechat|2|64|2011",
      "entity_data_1": "Y|DEAL|12|N|N|4|1635166080|0|0|0",
      "entity_data_2": "LEAD|0|COMPANY|0|CONTACT|9|DEAL|12",
      "entity_data_3": "",
      "mute_list": [],
      "manager_list": [],
      "date_create": "2021-10-25T14:48:00+02:00",
      "message_type": "L"
    },
    "lines": {
      "id": 4,
      "status": 5,
      "date_create": "2021-10-25T14:48:00+02:00"
    },
    "user": {
      "id": 0
    },
    "options": []
  },
  {
    "id": "chat39",
    "chat_id": 39,
    "type": "chat",
    "avatar": {
      "url": "",
      "color": "#4ba984"
    },
    "title": "Chat is not for robots",
    "message": {
      "id": 161,
```

```
    "text": "here's a question",
    "file": false,
    "author_id": 1,
    "attach": false,
    "date": "2021-10-08T13:19:33+02:00",
    "status": "delivered"
  },
  "counter": 0,
  "pinned": false,
  "unread": false,
  "date_update": "2021-10-08T13:19:33+02:00",
  "chat": {
    "id": 39,
    "name": "Chat is not for robots",
    "owner": 1018,
    "extranet": false,
    "avatar": "",
    "color": "#4ba984",
    "type": "chat",
    "entity_type": "",
    "entity_id": "",
    "entity_data_1": "",
    "entity_data_2": "",
    "entity_data_3": "",
    "mute_list": [],
    "manager_list": [],
    "date_create": "2021-09-27T15:57:53+02:00",
    "message_type": "C"
  },
  "user": {
    "id": 1,
    "active": true,
    "name": "Aleksey Shakhvorostov",
    "first_name": "Aleksey",
    "last_name": "Shakhvorostov",
    "work_position": "",
    "color": "#df532d",
    "avatar": "",
    "gender": "M",
    "birthday": "",
    "extranet": false,
    "network": false,
```

```
    "bot": false,
    "connector": false,
    "external_auth_id": "default",
    "status": "online",
    "idle": false,
    "last_activity_date": "2021-10-30T15:52:40+02:00",
    "mobile_last_date": "2021-10-27T16:39:26+02:00",
    "absent": false,
    "departments": [
      1
    ],
    "phones": false,
    "desktop_last_date": "2021-10-21T11:07:54+02:00"
  },
  "options": []
},
{
  "id": 1018,
  "chat_id": 28,
  "type": "user",
  "avatar": {
    "url": "",
    "color": "#df532d"
  },
  "title": "Lev Tregubov",
  "message": {
    "id": 160,
    "text": "Push notifications are coming",
    "file": false,
    "author_id": 1018,
    "attach": false,
    "date": "2021-10-08T13:19:32+02:00",
    "status": "delivered"
  },
  "counter": 0,
  "pinned": false,
  "unread": false,
  "date_update": "2021-10-08T13:20:59+02:00",
  "user": {
    "id": 1018,
    "active": true,
    "name": "Lev Tregubov",
```

```
        "first_name": "Lev",
        "last_name": "Tregubov",
        "work_position": "",
        "color": "#df532d",
        "avatar": "",
        "gender": "M",
        "birthday": "",
        "extranet": false,
        "network": false,
        "bot": false,
        "connector": false,
        "external_auth_id": "default",
        "status": "online",
        "idle": false,
        "last_activity_date": "2021-10-29T16:29:17+02:00",
        "mobile_last_date": "2021-10-08T13:24:22+02:00",
        "absent": false,
        "departments": [
          1
        ],
        "phones": false,
        "desktop_last_date": "2021-10-18T16:51:09+02:00"
      },
      "options": []
    }
  ],
  "hasMore": false
}
```


Description of keys:

- id -- conversation ID (if digit is user; if **chatXXX** is chat)
- type -- record type (if **user** is a user, if **chat** is a chat)
- avatar -- post avatar description object:
  - url -- link to the avatar (if empty, then the avatar is not set)
  - color -- dialog color in **hex** format
- title -- post title (first and last name -- for user, chat name -- for chat)
- messages -- message description object:
  - id -- message ID

- text -- message text (no BB codes or line breaks)
- file -- files present (true/false)
- attach -- attachments present (true/false)
- author_id -- post author
- date -- message date

- counter -- counter of unread messages
- user -- user data description object (not available if post type is chat):
  - id -- user ID
  - name -- user's first and last name
  - first_name -- username
  - last_name -- user last name
  - work_position -- position
  - color -- user color in **hex** format
  - avatar -- link to the avatar (if empty, no avatar has been set)
  - gender -- user's gender
  - birthday -- user's birthday in **DD-MM format,** if empty -- not beat
  - extranet -- sign of an external extranet user (true/false)
  - network -- attribute of *Bitrix24.Network* user (true/false)
  - bot -- sign of a bot (true/false)
  - connector -- indication of the user of open lines (true/false)
  - external_auth_id -- external authorization code
  - status -- selected user status
  - idle -- date the user left the computer, in **ATOM** format (false if not set)
  - last_activity_date -- date of the user's last activity in **ATOM** format
  - mobile_last_date -- date of the last action in the mobile application in **ATOM** format (if not set, then false)
  - absent -- date by which date the user has a vacation, in **ATOM** format (if not set, then false)
- chat -- chat data description object (not available if post type is user):

- id -- chat identifier
- title -- chat name
- owner -- identifier of the user-owner of the chat
- extranet -- sign of participation in the chat of an external extranet user (true/false)
- color -- chat color in **hex** format
- avatar -- link to the avatar (if empty, no avatar has been set)
- type -- chat type (group chat, call chat, open line chat, etc.)
- entity_type -- external code for chat -- type
- entity_id -- external code for chat -- identifier
- entity_data_1 -- external data for chat
- entity_data_2 -- external data for chat
- entity_data_3 -- external data for chat
- date_create -- chat creation date in **ATOM** format
- message_type -- chat message type

- pinned -- chat is pinned or not
- unread -- whether there is a manual flag that the chat is not read
- date_update -- date of last change in related entities

Working with counters

Note! All methods are specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Method	Description of the method
im.counters.get	Getting counters.

im.counters.get

Getting counters

Options

No

Method call

JavaScript

```
BX24.callMethod('im.counters.get', {}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.counters.get', Array(
), $_REQUEST["auth"]);
```

Response Example

```
{
    "result": {
        "CHAT": {18: 1},
        "DIALOG": {1: 3, 5: 1},
        "LINES": {},
        "TYPE" {
            "ALL": 5,
            "CHAT": 1,
            "DIALOG": 4,
            "LINES": 0,
            "NOTIFY": 0
        }
    }
}
```

Description of keys:

- CHAT -- an object containing a list of chats that have unread messages
- DIALOG -- an object containing a list of dialogs that have unread messages
- LINES -- an object containing a list of open line chats that have unread messages
- TYPE -- object, contains total counters
  - ALL -- total count of all entities
  - CHAT -- total chat counter
  - DIALOG -- total dialog counter
  - LINES -- total counter of open lines
  - NOTIFY -- total notification counter

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Working with notifications

Method	Description of the method
im.notify.personal.add	Sending a personal notification.
im.notify.system.add	Sending a personal notification.
im.notify.delete	Removing a notification.
im.notify.read	Setting the cancellation of read notifications.

Note! All of the above methods are specified using the restCommand function - this is a method for sending data to Bitrix24, this method is available in

EchoBot ***example , and is presented here as an example. you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.***

Methods specified using javascript method BX24.callMethod:

Method	Description of the method
im.notify.read.list	"Reading" the list of notifications, excluding CONFIRM type notifications .
im.notify.answer	Reply to a notification that supports fast answer.
im.notify.confirm	Interaction with notification buttons.

im.notify.personal.add

Sending a personal notification

Revision: 18

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, And presented here as an example. You can use your function or javascript-ÿÿÿÿÿ BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
USER_ID	1	**Yes**	Identifier user to whom will be addressed notification	18
MESSAGE	Personal notification	**Yes**	Text notifications	18
MESSAGE_OUT	Text personal notifications for mail	No	Text notifications for mail. If not given, then	18

			used field MESSAGE	
TAG	TEST	No	Tag notifications unique within the system. When adding a notification with an existing tag, other notifications will be removed	18
SUB_TAG	SUB|TEST	No	Additional tag, without checking for uniqueness	18
ATTACH		No	The attachment	18

Related links

How to work with attachments

Method call and response

PHP

```
$result = restCommand('im.notify.personal.add', Array(

    'USER_ID' => 1,
    'MESSAGE' => 'Personal notification',
    'MESSAGE_OUT' => 'Text of personal notification for mail',
    'DAY' => 'TEST',
    'SUB_TAG' => 'SUB|TEST',
    'ATTACH' =>

), $_REQUEST["auth"]);
```

Response Example

```
{
    "result": 123
```

```
}
```

Execution result: notification ID or error.

Example response when an error occurs

```
{
    "error": "USER_ID_EMPTY",
    "error_description": "Recipient ID not set"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
USER_ID_EMPTY	Recipient ID not set
MESSAGE_EMPTY	Message text not sent
ATTACH_ERROR	The entire attachment object passed is not passed validation
ATTACH_OVERSIZE	Maximum attachment size exceeded (30 KB)

im.notify.system.add

Sending a system notification

Revision: 18

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
USER_ID	1	**Yes**	Identifier user to whom will be addressed notification	18
MESSAGE	Systemic notification	**Yes**	Notification text	18
MESSAGE_OUT	Text systemic notifications for mail	No	Notification text for mail. If not given, then used field MESSAGE	18
TAG	TEST	No	notification tag, unique within systems. At adding notifications from existing tag other notifications will be removed	18
SUB_TAG	SUB|TEST	No	Additional tag, no check for uniqueness	18
ATTACH	Array()	No	The attachment	18

Related links

How to work with attachments

Method call and response

PHP

```
$result = restCommand('im.notify.system.add', Array(
```

```
    'USER_ID' => 1,
    'MESSAGE' => 'System notification',
    'MESSAGE_OUT' => 'System notification text for mail',
    'DAY' => 'TEST',
    'SUB_TAG' => 'SUB|TEST',
    'ATTACH' => Array()

), $_REQUEST["auth"]);
```

Response Example

```
{
    "result": 123
}
```

Execution result: notification ID or error.

Example response when an error occurs

```
{
    "error": "USER_ID_EMPTY",
    "error_description": "Recipient ID not set"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
USER_ID_EMPTY	Recipient ID not set
MESSAGE_EMPTY	Message text not sent
ATTACH_ERROR	The entire attachment object passed is not passed validation
ATTACH_OVERSIZE	Maximum attachment size exceeded (30 KB)

im.notify.delete

Removing a notification

Revision: 18

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, And presented here as an example. You can use your function or javascript-ÿÿÿÿÿ BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
ID	123	**Yes***	Identifier notifications	18
TAG	TEST	**Yes***	notification tag, unique within systems.	18
SUB_TAG	SUB|TEST	**Yes***	Additional tag, without uniqueness checks	18

* You need to specify **one of three** required parameters to choose from: ID (notification identifier), TAG (notification tag), or SUB_TAG (sub-tag).

Related links

How to work with attachments

Method call and response

PHP

```
$result = restCommand('im.notify.delete', Array(

    'ID' => 13,
    'DAY' => 'TEST',
```

```
    'SUB_TAG' => 'SUB|TEST'

), $_REQUEST["auth"]);
```

Response Example

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

Execution result: true or error.

Example response when an error occurs

```
{
    "error": "PARAMS_ERROR",
    "error_description": "Error deleting notification"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
PARAMS_ERROR	Error deleting notification

im.notify.read

Setting the cancellation of read notifications

Revision: 18

Note! The method specified using the restCommand function is method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
ID	17	**Yes**	Notification ID	18
ONLY_CURRENT N		No	Read only the specified notification	18

- If the ONLY_CURRENT parameter is passed as Y, the read mark will be set only for the specified ID. Otherwise, the notification will be checked if it is equal to or greater than the specified ID.

Method call and response

JavaScript

```
BX24.callMethod('im.notify.read', {
    'ID': 17,
}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('im.notify.read', Array(
    'ID' => '17'
), $_REQUEST["auth"]);
```

Response Example

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

im.notify.read.list

"Reading" the list of notifications, excluding notifications of type CONFIRM

Revision: 30

Options

Parameter	Example	Required	Description	Revision
IDS	[1,2,3]	**Yes**	Array of notification IDs	30
ACTION	'AND'	**No**	Mark as read| unread (Y|N)	30

Method call and response

JavaScript

```
B24.callMethod(
  'im.notify.read.list',
 {
   IDS: [1,2,3],
   ACTION: 'Y'
 },
 res => {
  if (res.error())
  {
    console.error(result.error().ex);
  }
  else
  {
    console.log(res.data())
  }
}
```

Response Example

```
true
```

Example response when an error occurs

```
{
  "error": "PARAMS_ERROR",
  "error_description": "No IDS param or it is not an array"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
PARAMS_ERROR	IDS parameter was not passed or is not an array

im.notify.answer

Reply to a notification that supports fast answer

Revision: 30

Options

Parameter	Example	Required	Description	Revision
ID	270	**Yes**	Identifier notifications, supportive quick response	30
ANSWER_TEXT	'Hello'	**Yes**	Quick text response	30

Method call and response

JavaScript

```
B24.callMethod(
  'im.notify.answer',
  {
    ID: 270,
    ANSWER_TEXT: 'Hello'
  },
  res => {
    if (res.error())
    {
      console.error(result.error().ex);
    }
    else
    {
      console.log(res.data())
    }
  }
)
```

Response Example

```
{
  "result_message": [
    "Your response has been sent successfully."
  ]
}
```

An array of messages is returned for your response.

Example of a response if passing a notification ID that does not support quick response

```
{
  "result_message": false
}
```

Example response when an error occurs

```
{
  "error":"NOTIFY_ID_ERROR",
  "error_description":"Notification ID can\u0027t be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
ID_ERROR	The ID parameter was not passed or it is not a number
ANSWER_TEXT_ERROR	The ANSWER_TEXT parameter was not passed or is not non-empty string

im.notify.confirm

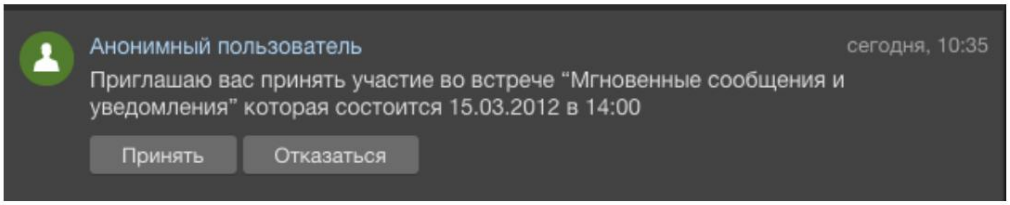
Interaction with notification buttons

Revision: 30

Options

Parameter	Example	Required	Description	Revision
ID	288	**Yes**	Identifier notifications, supportive choice of answer by pressing on buttons	30
NOTIFY_VALUE	'Y'	**Yes**	Meaning selected answer (button value)	30

For example, consider a notification:



- button **Accept** value 'Y'
- the **Reject** button has a value of 'N'

Method call and response

JavaScript

```
B24.callMethod(
  'im.notify.confirm',
  {
    ID: 288,
    NOTIFY_VALUE: 'Y'
  },
  res => {
    if (res.error())
    {
      console.error(result.error().ex);
    }
    else
    {
      console.log(res.data())
    }
  }
)
```

Response Example

```
{
  "result_message": [
    "Invitation Accepted"
  ]
}
```

Example response when an error occurs

```
{
  "error":"NOTIFY_VALUE_ERROR",
  "error_description":"Notification Value can\u0027t be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
ID_ERROR	The ID parameter was not passed or it is not a number
NOTIFY_VALUE_ERROR	NOTIFY_VALUE was not specified or is empty

Full list of Chat API methods

Platform version:

Method	Description of the method
im.revision.get	Getting information about API revisions.

Working with chats:

Method	Description of the method
im.chat.add	Create a chat.
im.chat.user.list	Getting a list of participants.
im.chat.user.add	Participant invitation.
im.chat.user.delete	Exclusion of participants.
im.chat.leave	Leave chat.
im.chat.sendTyping	Sending the status "you are being contacted...".

im.chat.updateTitle	Chat title update.
im.chat.updateColor	Chat color update.
im.chat.updateAvatar	Chat avatar update.
im.chat.setOwner	Change the owner of the chat.
im.chat.get	Get the chat ID.
im.chat.mute	Disable chat notifications.

Working with dialogues:

Method	Description of the method
im.dialog.messages.get	Getting a list of recent messages in chat.
im.dialog.read	Change the fact that messages were read: all messages up to the specified (including the message) are marked as read.
im.dialog.unread	Change the fact that messages were read: all messages after the specified (including the message) are marked as unread.
im.dialog.get	Getting information about the dialogue.

Working with messages:

Method	Description of the method
im.message.add	Sending a message to the chat.

im.message.update	Sending a chatbot message change.
im.message.delete	Delete a chatbot message.
im.message.like	"I like" setting.

Working with files:

Method	Description of the method
im.disk.folder.get	Getting information about a storage folder chat files.
im.disk.file.commit	Publication of the uploaded file in the chat.
im.disk.file.delete	Deleting files inside the chat folder.
im.disk.file.save	Saving the file to your Bitrix24.Disk.

Working with users:

Method	Description of the method
im.user.get	Getting user data.
im.user.list.get	Getting data about users.
im.user.business.list	Get a list of business users.
im.user.status.get	Getting information about installed user status.
im.user.status.set	Setting the user's status.
im.user.status.idle.start	Setting automatic status "Departed."

im.user.status.idle.end	Disabling automatic status "Departed."

Working with departments:

Method	Description of the method
im.department.get	Retrieve department information.
im.department.colleagues.list	Getting a list of users in your department.
im.department.managers.get	Getting a list of leaders divisions.
im.department.employees.get	Getting a list of employees subdivision.

Working with search:

Method	Description of the method
im.search.user.list	User search.
im.search.chat.list	Chat search.
im.search.department.list	Search for divisions.
im.search.last.get	Getting a list of the elements of the last asked.
im.search.last.add	Adding a recent history element asked.
im.search.last.delete	Removing a recent history item asked.

Working with the list of recent chats:

Method	Description of the method
im.recent.get	List of the user's recent conversations.
im.recent.pin	Fixing the dialogue in favorites.
im.recent.hide	Remove a conversation from the list of recent chats.

Working with counters:

Method	Description of the method
im.counters.get	Getting counters.

Working with notifications:

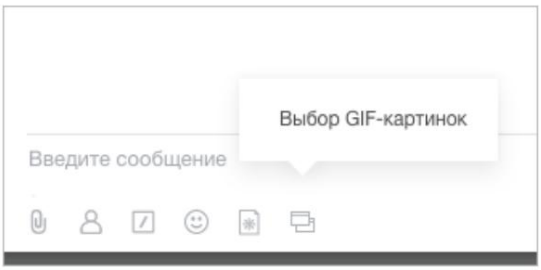
Method	Description of the method
im.notify.personal.add	Sending a personal notification.
im.notify.system.add	Sending a personal notification.
im.notify.delete	Removing a notification.
im.notify.read	Setting undo about read notifications.

Chat Applications

What it is?

Chat Applications

Developers now have the opportunity to integrate into Messenger by adding their own icon in the text input panel:



If the application does not load the picture, **the Chat application's** service icon will be displayed , clicking on which will show the text version of the icon.

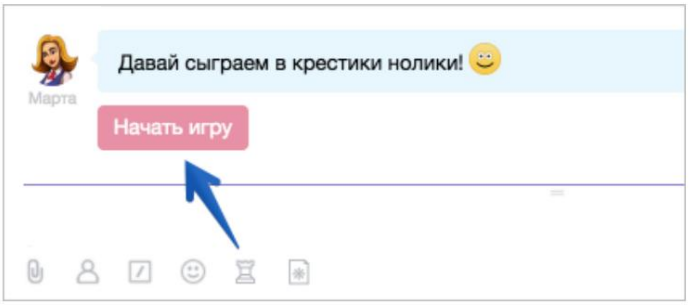
There are two types of chat application - **JS command** or **IFRAME application.**

JS command

Clicking on the icon will insert a command for the chatbot into the input field, or send a command to the chat, or start a phone call, or open an open support line.

Using this format, developers in their chatbots can make a button for connections with them.

An example of such a command for the chat bot Marta - there will be an icon for playing in tic-tac-toe:



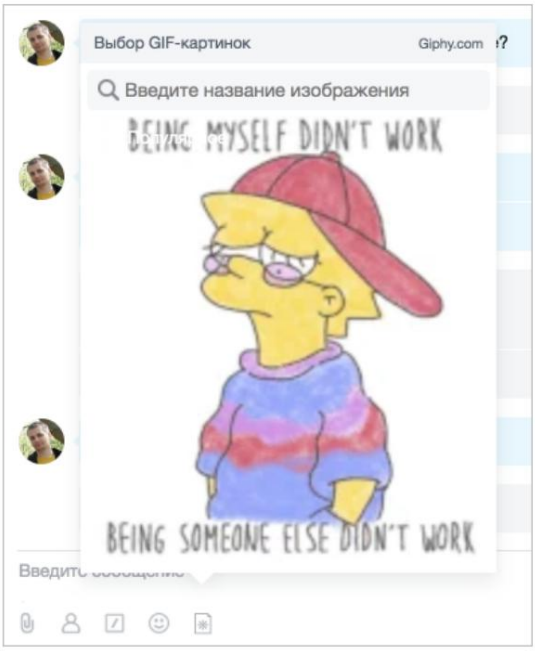
IFRAME application

This is an improved format, clicking on the icon will open the IFRAME an application in which the developer can do whatever he wants.

The application can interact with the chat using JS commands:

- to insert a message into the input field
- to send a message on behalf of the user
- to close the dialog
- opening a technical support chat (OL consultant)
- making a call

You can see an example of such an implementation in the example of the GIphy chat bot:



Feel the difference - before you wrote a command in the form of a message, and GIphy gave you a random picture on the topic. Now you see what you send.

Please note that the icons are context aware, which means that the application can only be displayed in those chats in which you require it.

For example, to communicate with technical support, it would be more correct to place the application in the context of your chat bot, because in other chats it will be superfluous. Or you can make a special application for open lines - it should only be shown in the context of open lines.

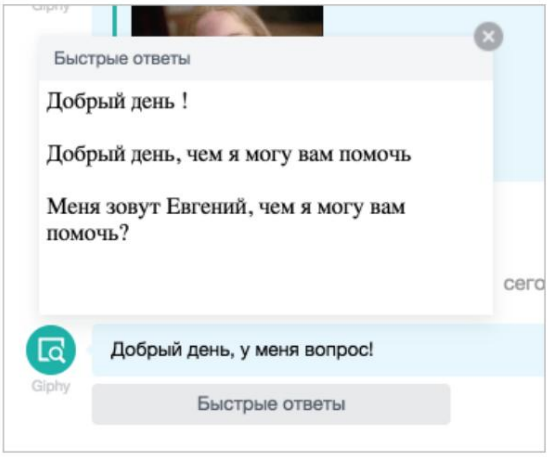
Available contexts: all, chat, bot, lines, user, call.

You can add a postfix -admin to each context - *then the icon will be shown in the right context only to administrators.*

Context applications

Contextual applications are designed to help you interact a user with a chatbot within a specific dialogue (message).

For example, a client writes to an open line, a chatbot for open lines analyzes the message and prepares response options. In order not to interfere with the operators' work and not show them the entire flow of information, we carefully form a button, by clicking on which the IFRAME application will open.



Create an application

Applications are divided into two types: JS command and IFRAME application.

Let's talk about API methods for working with applications.

- imbot.app.register - registration application for chat.
- imbot.app.unregister - deleting the chat application.
- imbot.app.update - updating app data in chat.

Registering a Chat App

Rest-Method: imbot.app.register

Method call for JS command:

```
$result = restCommand('imbot.app.register', Array(

    'BOT_ID' => 62, // chat application owner bot ID
    'CODE' => 'echo', // chat application code
    'JS_METHOD' => 'SEND',
    'JS_PARAM' => '/help',
    'ICON_FILE' => '/* base64 image */', // Your app's icon is base64
    'CONTEXT' => 'BOT', // context applications
    'EXTRANET_SUPPORT' => 'N', // is the command available to extranet users, by default N
    'LIVECHAT_SUPPORT' => 'N', // live chat support
```

```
        'IFRAME_POPUP' => 'N', // the iframe will be opened and moved inside
messenger, the transition between dialogs will not close such a window.
        'LANG' => Array( // array of translations, it is desirable to specify at least for RU and EN
          Array('LANGUAGE_ID' => 'en', 'TITLE' => 'Echobot BUTTON', 'DESCRIPTION' => 'Send help
command'),


        )


    ), $_REQUEST["auth"]);
```

Options for JS_METHOD:

- PUT - inserting a command to a chatbot in a textarea, JS_PARAM can contain only the text of the command and its parameters (allowed characters: az 0-9 - + _ /).
- SEND - sending a command to a chatbot, JS_PARAM can contain only the text of the command and its parameters (allowed characters az 0-9 - + _ /).
- CALL - call a phone number.
- SUPPORT - opening a technical support chatbot working through Open Lines, in JS_PARAM you must specify the open line code (32 characters).

Application icon format ICON_FILE:

- Base64 from your icon must be passed to this field, the icon format is described in detail in the documentation.
- Please note that if this field is not specified, then the application will be available in the "Applications for Chat" system dialog.

Context applications CONTEXT:

- ALL - the application will be available in all chats.
- USER - The application will only be available in One-to-One chats.
- CHAT - the application will be available only in group chats.
- BOT - the application will be available only to the chatbot that installed the application.
- LINES - the application will be available only in Open Lines chats.
- CALL - the application will be available only in chats created within Telephony.

Postfix -admin can be added to each context , then the application will be available in the desired context only for Bitrix24 administrators.

Calling a method for an IFRAME application:

```
    $result = restCommand('imbot.app.register', Array(


        'BOT_ID' => 62, // chat application owner bot ID
        'CODE' => 'echo', // chat application code
        'IFRAME' => 'https:// marta.bitrix.info/ iframe/ echo.php',
        'IFRAME_WIDTH' => '350', // desired frame width. Minimum value - 250px
        'IFRAME_HEIGHT' => '150', // desired frame height. Minimum value - 50px
        'HASH' => 'd1ab17949a572b0979d8db0d5b349cd2', // token to access your frame
for signature verification, 32 characters.
        'ICON_FILE' => '/* base64 image */', // Your app's icon is base64
        'CONTEXT' => 'BOT', // context applications
```

```
        'HIDDEN' => 'N', // hidden application or not
        'EXTRANET_SUPPORT' => 'N', // is the command available to extranet users, by
default N
        'LIVECHAT_SUPPORT' => 'N', // live chat support
        'IFRAME_POPUP' => 'N', // the iframe will be opened and moved inside
messenger, the transition between dialogs will not close such a window.
        'LANG' => Array( // array of translations, it is desirable to specify at least for RU and EN
          Array('LANGUAGE_ID' => 'en', 'TITLE' => 'Echobot IFRAME', 'DESCRIPTION' => 'Open Echobot
IFRAME app', 'COPYRIGHT' => 'Bitrix24'),
        )

    ), $_REQUEST["auth"]);
```

Note:

1. ***Although you set the size of the window when registering the IFRAME application, it can be reduced by the application to actually available sizes. The ideal option is to make the application in such a way that it fits into the minimum dimensions and can freely adapt to increase and decrease, this will be very important for mobile devices.***

2. *A link to an IFRAME must necessarily lead to a site with an active HTTPS certificate. You can read the rules for developing an IFRAME handler and restrictions in the documentation.*

3. ***Be sure to use hash checks before performing any operations.***

4. ***Applications are available only after a page reload or after a service hit on server (on different installations in different ways, from 15 to 26 minutes). The application will appear instantly only when interacting with the context button.***

5. *In the IFRAME_WIDTH and IFRAME_HEIGHT* values *you specify the desired width and height of the application window. If the messenger window is smaller than the application window, then the application window will be reduced to the size of the messenger window. If the messenger window is larger than the application window, then the desired dimensions for the application will be used.*

6. *If the application is called from the context (keyboard or menu), the application will be called in context mode, in this mode the IFRAME_POPUP* value *will be ignored.*

Execution result: Numeric identifier of the created application APP_ID or an error.

Possible mistakes:

Error code	Error Description
BOT_ID_ERROR	Chatbot not found.
APP_ID_ERROR	The chat application does not belong to this rest application, you can only work with chat applications installed within the current rest application.
IFRAME_HTTPS	The link to the IFRAME must be on a site with an active HTTPS certificate.
HASH_ERROR	You have not provided a hash for the application. We recommend using unique hash for each iframe and each installation.

PARAMS_ERROR	Error when specifying JS command or IFRAME parameters.
NO ERROR	No language phrases were passed for the visible command.
WRONG_REQUEST	Something went wrong.

Uninstalling a chat app

Rest-Method: imbot.app.unregister

Method call:

```
$result = restCommand('imbot.app.unregister', Array(
    'APP_ID' => 13, // command ID to delete
), $_REQUEST["auth"]);
```

Execution result: true or error.

Possible mistakes:

Error code	Error Description
CHAT_APP_ID_ERROR	Application not found.
APP_ID_ERROR	Chat application does not belong to this rest application, work only possible with chat apps installed within the current rest app.
WRONG_REQUEST	Something went wrong.

Update app data in chat

Please note that the required fields are the application ID and one of the required fields for editing. If you specify the JS and IFRAME methods in the same command, only JS will be used.

Remainder: imbot.app.update

Method call:

```
$result = restCommand('imbot.app.update', Array(
    'APP_ID' => 13, // chat ID
```

```
        'FIELDS' => Array(
            'IFRAME' => 'https:// marta.bitrix.info/ iframe/ echo.php',
            'IFRAME_WIDTH' => '350', // desired frame width. Minimum value - 250px
            'IFRAME_HEIGHT' => '150', // desired frame height. Minimum value - 50px
            'JS_METHOD' => 'SEND',
            'JS_PARAM' => '/ help',
            'HASH' => 'register', // token to access your frame, 32 characters.
            'ICON_FILE' => '/ * base64 image */', // Your app's icon is base64
            'CONTEXT' => 'BOT', // context applications
            'EXTRANET_SUPPORT' => 'N', // is the command available to extranet users, default is N

            'LIVECHAT_SUPPORT' => 'N', // live chat support
            'IFRAME_POPUP' => 'N', // the iframe will be opened and moved inside
messenger, the transition between dialogs will not close such a window
            'LANG' => Array( // array of translations, it is desirable to specify at least for RU and EN
                Array('LANGUAGE_ID' => 'en', 'TITLE' => 'Echobot IFRAME', 'DESCRIPTION' => 'Open Echobot
IFRAME app', 'COPYRIGHT' => 'Bitrix24'),
            )
        )
    ), $_REQUEST["auth"]);
```

Execution result: true or error.

Possible mistakes:

Error code	Error Description
CHAT_APP_ID_ERROR	Application not found.
APP_ID_ERROR	Chat application does not belong to this rest application, work only possible with chat apps installed within the current rest app.
IFRAME_HTTPS	The link to the IFRAME must be on a site with an active HTTPS certificate.
WRONG_REQUEST	Something went wrong.

Context applications

You can read about what context applications are here. To work with the context, you need to perform several actions:

1. Register the application. You can register a hidden application, then it will not be displayed on the text input panel. 2. Send (or update) any message with a keyboard attached or from the menu. 3. In the parameters of a keyboard button or menu item, you must pass an identifier applications.

Method call:

```
restCommand('imbot.message.add', Array(
    "DIALOG_ID" => 2,
    "BOT_ID" => 17,
    "MESSAGE" => "Hello! My name is EchoBot :)",
    "KEYBOARD" => [{"TEXT":"Open App","APP_ID":11}],
    "MENU" => [{"TEXT":"Open App","APP_ID":11}]

), $_REQUEST["auth"]);
```

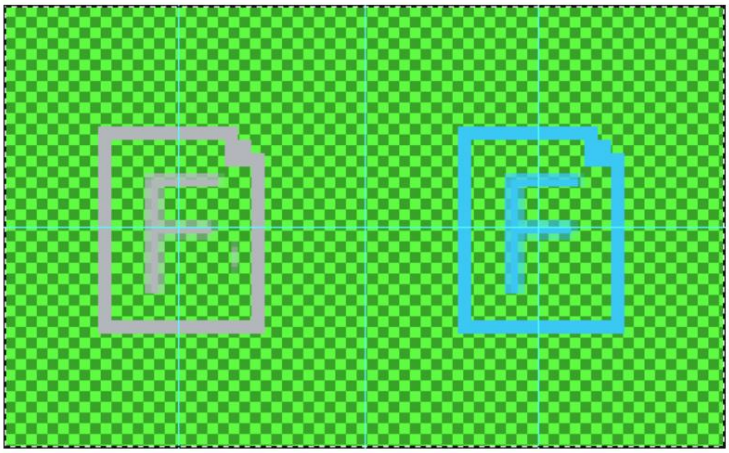
Besides APP_ID, you can pass any string to APP_PARAMS when opening your IFRAME
the data will be passed to the BUTTON_PARAMS parameter.

You can read the rules for developing an IFRAME handler and restrictions in the documentation. At
when creating a message, you can use one of two options -- keyboard (KEYBOARD) or context menu (MENU).

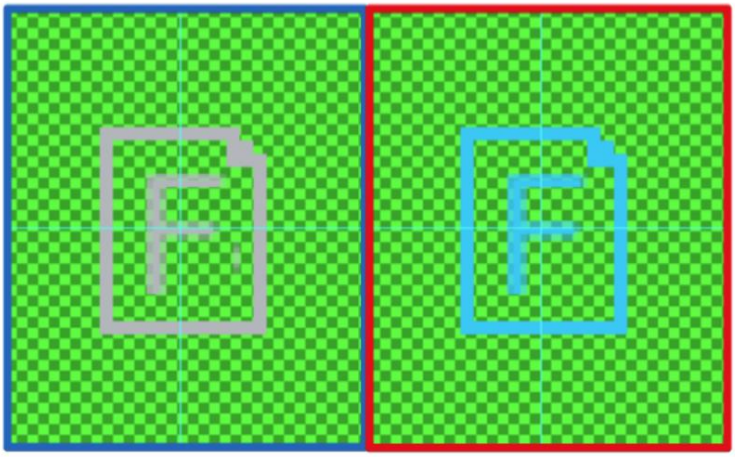
Creating a Chat Application Icon

When creating a **chat application** icon , you need to follow a few simple rules:

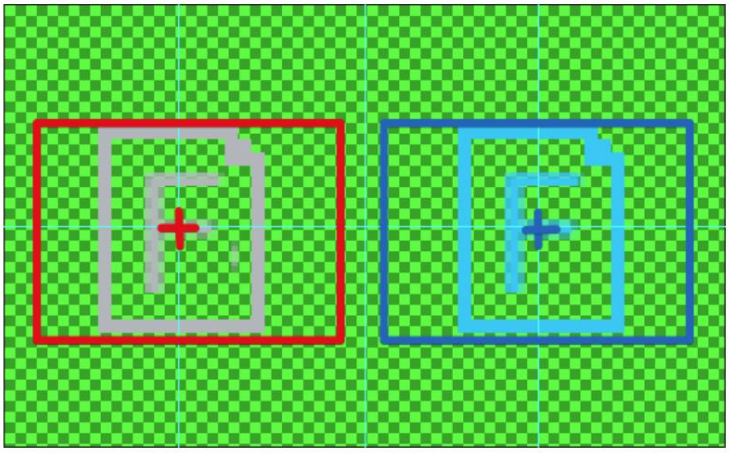
1. The icon must be on a transparent background, in .PNG format.



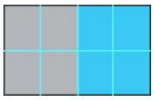
2. Canvas size: 108x66 px, the left side is used for an inactive icon, the second one is for display active:



3. The main part of the icon should be no more than 45x32 px and must be centered on middle of each part:



4. The main color of the inactive icon is #b3b7bc, the active one is #2fc7f7:



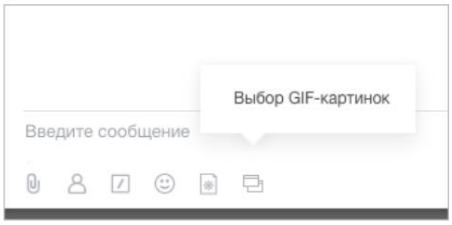
To make it easier for you to create them, we attach several templates:

textarea_icon_f.psd marta_icon.psd textarea_icon.psd

5. To upload a picture, when registering or updating the application, you need to pass the IMAGE_FILE key. The value of this key must be the base64 of your image:

```
'ICON_FILE' => base64_encode(file_get_contents(__DIR__.'/icon_button.png')),
```

Please note that you do not need to create an icon for the application, in which case it will be grouped into a service button:



Creating an IFRAME Handler

iframe handler

After registering the chat application, you need to make **an IFRAME handler.** ***Note that although you specify the size of the window when you register, it can be reduced by the application to the actual size available. The ideal option is to make the application in such a way that it fits into the minimum dimensions and can freely adapt to increase and decrease, this will be very important for mobile devices.***

Your page will open with the following data

```
Array
(
    [BOT_ID] => 17
    [ BOT_CODE ] = > echoBot
    [APP_ID] => 11
    [APP_CODE] => echoFrame
    [DOMAIN] => https:// test.bitrix24.ru
    [DOMAIN_HASH] => 0e9c40cee01d6f182e9261b38b30b5c3
    [USER_ID] => 2
    [USER_HASH] => 7e23ac8b6f6c7044076301c7f81cd745
    [DIALOG_ID] => 950
    [CONTEXT] => textarea
    [LANG] => ru
    [IS_CHROME] => Y
    [MESSAGE_ID] => 12333
    [BUTTON_PARAMS] => test
    [DIALOG_ENTITY_ID] => telegrambot|8|173633217|569
    [DIALOG_ENTITY_DATA_1] => Y|LEAD|56|N|N|1765
    [DIALOG_CONTEXT] => lines

)
```

More about options:

- BOT_ID and BOT_CODE - data about the chatbot under which the application is running.
- APP_ID and APP_CODE are app data for the running app's chat.
- DOMAIN - the address of the portal from which the application was launched.
- DOMAIN_HASH is the HASH field passed when the icon was registered. Used to cut off unauthorized requests.
- USER_ID - user ID.
- USER_HASH - hash string for validating the correctness of requests on the client portal.

- DIALOG_ID - ID of the user's running dialog when the frame was opened.
- CONTEXT is the context for calling the dialog, it can be a textarea or a button.
- LANG - current client interface language.
- IS_CHROME - whether this frame is running in the Google Chrome browser or not.
- DIALOG_CONTEXT - **this parameter is not yet available.** Can be all, chat, bot, lines, user, call .
- DIALOG_ENTITY_ID - additional data about the chat **(this parameter is not yet available).** For Open Lines, the data is separated by |. The value of the fields for Open lines: 1 - the channel through which the user wrote, 2 - the identifier of the OL, 3 and 4 - service data (the parameter is not yet available).
- DIALOG_ENTITY_DATA_1 - additional data about the chat **(this parameter is not yet available).** For Open Lines, the data is separated by |. Field values for Open lines: 1 - saved in CRM, 2 - CRM entity type, 3 - CRM entity identifier, 4 and 5 - service data, 6 - open line session identifier.

If the frame is running in context mode:

- MESSAGE_ID - message identifier.
- BUTTON_PARAMS - button parameter set when submitting.

Interaction with parent window (with messenger)

You need to make functions for interacting with the main window, initialization and dispatch data.

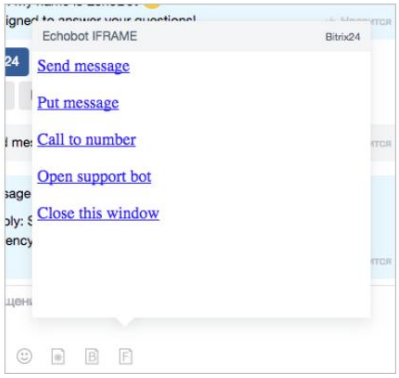
Implementation examples:

```
<script type="text/javascript">
  // function to initialize communication with the main window
  function frameCommunicationInit()
  {
    if (!window.frameCommunication)
    {
      window.frameCommunication = {timeout: {}};
    }
    if(typeof window.postMessage === 'function')
    {
      window.addEventListener('message', function(event){
        var data = {};
        try { data = JSON.parse(event.data); } catch (err){}

        if (data.action == 'init')
        {
          frameCommunication.uniqueLoadId = data.uniqueLoadId;
          frameCommunication.postMessageSource = event.source;
          frameCommunication.postMessageOrigin = event.origin;
        }
      });
    }
  }

  // function to send data to the main window
  function frameCommunicationSend(data)
```

```
    {
      data['uniqueLoadId'] = frameCommunication.uniqueLoadId;
      var encodedData = JSON.stringify(data);
      if (!frameCommunication.postMessageOrigin)
      {
        clearTimeout(frameCommunication.timeout[encodedData]);
        frameCommunication.timeout[encodedData] = setTimeout(function(){
          frameCommunicationSend(data);
        }, 10);
        return true;
      }

      if(typeof window.postMessage === 'function')
      {
        if(frameCommunication.postMessageSource)
        {
          frameCommunication.postMessageSource.postMessage(
            encodedData,
            frameCommunication.postMessageOrigin
          );
        }
      }
    }
    frameCommunicationInit();
</ script>
```

After that, the following functions will be available to you:

- Command to send text on behalf of the user:
  frameCommunicationSend({'action': 'send', 'message': 'Send message'})
- Command to insert text into input field:
  frameCommunicationSend({'action': 'put', 'message': 'Put message'})
- Call start command:
  frameCommunicationSend({'action': 'call', 'number': '123456'})
- Command to open a dialogue with your open line: frameCommunicationSend({'action': 'support', 'code': '6a4cdbcf753addac1a573ea64be826ca'})
- Frame close command:
  frameCommunicationSend({'action': 'close'})



Registration and security

You can choose not to check incoming requests, but we strongly recommend that you do so. Based on the incoming data, you can implement **the necessary checks:**

1. We recommend checking REFERER for a match with the specified domain.
2. We recommend checking the domain and its hash.
3. We recommend checking the user and the hash from him. 4. Use a unique HASH for each new registration.

With all of these factors combined, you should be able to make a fairly secure application. When registering a team, you must specify the HASH string that the domain will be signed with and user:

```
$hash = '0e9c40cee01d6f182e9261b38b30b5c3'; // hash specified during registration applications

$check = parse_url($_GET['DOMAIN']);

// check for validity of the specified domain
if ($_GET['DOMAIN_HASH'] == md5($check['host'].$hash))
{
  echo 'OK';
}

// check for validity of the specified user
if ($_GET['USER_HASH'] == md5($_GET['USER_ID'].$hash))
{
  echo 'OK';
}

// check REFERER
if (strpos($_SERVER['HTTP_REFERER'], $_GET['DOMAIN'])!== 0)
{
  echo 'OK';
}
```

Working in this mode is described in detail in the demo, which you can download here.

Rest API

Bot Rest APIs are an integral part of the overall REST system. Documentation for all other methods is located in the REST API Documentation.

Rest API Features

Rest API is designed to create applications for the ***Bitrix24 cloud service.***

Rest API allows you to access and manage such tools of the ***Bitrix24*** cloud service How:

- CRM,
- Business processes,
- disc,

- Social network,
- Telephony,
- Universal Lists,
- Data warehouses (infoblocks),
- notifications,
- Tasks,
- work with users and departments,
- live tape,
- Calendars.

To use Rest API methods by external applications, the application must be registered in the Marketplace. In this case, the application has all the data it needs to get the OAuth 2.0 token.

In general, calling a REST method looks like this:

```
https://domain_B24.bitrix24.{ru|en|de}/rest/method_name.transport?
method_parameters&auth=authorization_key
```

where **transport** is an optional parameter that can be either **json** or **xml.** The default is **json.**

The request can be sent using the GET or POST method. Method

parameter values are accepted in UTF-8 encoding. ***Attention! There***

is a limit on the number of requests. Two requests per second are allowed. If the limit exceeded, the limit starts to work after 50 requests. Note: If your application/

webhook creates an abnormal load on the portal, it may blocking work:

```
[
'error' => 'OVERLOAD_LIMIT',
'error_description' => 'REST API is blocked due to overload.'
]
```

To unlock it, you need to contact technical support.

Bot Rest APIs are an integral part of the overall REST system. Documentation for all other methods is located in the REST API Documentation.

Rest API Open Lines

In order to use the IMOPENLINES REST methods, in addition to the rights to imbot (Creating and managing chatbots) have access to scope **imopenlines** (Open lines).

Platform version

imopenlines.revision.get -- getting information about Open Lines API revisions

Revision: 2

Options

No.

Call example

JavaScript

```
BX24.callMethod('imopenlines.revision.get', {}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('imopenlines.revision.get', Array(
), $_REQUEST["auth"]);
```

Response Example

```
{
    "result": {
        "rest": 2,
        "web": 1,
        "mobile": 1,
    }
}
```

Description of keys:

- rest -- api revision for rest clients
- web -- API revision for web/desktop client
- mobile -- API revision for mobile client

Note! The method is specified using the restCommand function - this is the method of sending data to Bitrix24, this method is in the EchoBot example, and is presented here as an example. You can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Working with open lines

Attention! In order to use the IMOPENLINES REST methods, in addition to the rights to imbot (Creating and managing chatbots) have access to scope imopenlines (Open lines).

Method	Description of the method
imopenlines.network.join	Connecting an open line by code.

imopenlines.network.message.add	Sending a message on behalf of open line to the selected user.

imopenlines.network.join

Connecting an open line by code

Revision: 1

Attention! In order to use the IMOPENLINES REST methods, in addition to the rights to imbot (Creating and managing chatbots) have access to scope imopenlines (Open lines).

Options

Parameter	Example	Required	Description	Revision
CODE	ab515f5d85a8b844d484f6ea75a2e494 **ÿÿ**		Search Code connector pages	1

Method call and response

PHP

```
$result = restCommand('imopenlines.network.join', Array(

    'CODE' => 'ab515f5d85a8b844d484f6ea75a2e494'

), $_REQUEST["auth"]);
```

Response Example

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

Execution result: true or error.

Example response when an error occurs

```
{
    "error": "NOT_FOUND",
    "error_description": "Openline is not found"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
IMBOT_ERROR	Bot management module not installed
NOT_FOUND	Open line not found
INACTIVE	The open line is currently unavailable

imopenlines.network.message.add

Sending a message on behalf of an open line to the selected user

Revision: 1

Attention! In order to use the IMOPENLINES REST methods, in addition to the rights to imbot (Creating and managing chatbots) have access to scope imopenlines (Open lines).

Options

Parameter	Example	Required	Description	Revisions
CODE	ab515f5d85a8b844d484f6ea75a2e494 **ÿÿ**		The code registered open line	1
USER_ID	2	**Yes**	Identifier user the recipient messages	1
MESSAGE	message text	**Yes**	Message text	1
ATTACH		No	The attachment	1
KEYBOARD		No	Keyboard	1
URL_PREVIEW Y		No	Transform links in rich links, by default 'Y'	1

Method call and response

PHP

```
$result = restCommand('imopenlines.network.message.add', Array(

    'CODE' => 'ab515f5d85a8b844d484f6ea75a2e494',
    'USER_ID' => 2,
    'MESSAGE' => 'message text',
    'ATTACH' => '',
    'KEYBOARD' => '',
    'URL_PREVIEW' => 'Y'

), $_REQUEST["auth"]);
```

Response Example

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

Execution result: true or error.

Limits

- You can send a message no more than once per user during weeks.

Note: There are no restrictions on portals with the Partner tariff (NFR-license).

- You can only use the keyboard to form a link button to an external site.

Example response when an error occurs

```
{
    "error": "CODE_ERROR",
    "error_description": "Incorrect open line code specified"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CODE_ERROR	Incorrect open source code specified lines.
USER_ID_EMPTY	The recipient's user ID was not passed.
MESSAGE_EMPTY	Message text not sent.
ATTACH_ERROR	The entire attachment object passed is not has been validated.

ATTACH_OVERSIZE	Exceeded the maximum allowable attachment size (30 Kb).
KEYBOARD_ERROR	The entire passed keyboard object is not has been validated.
KEYBOARD_OVERSIZE	Maximum OVERSIZE exceeded. keyboard size (30 Kb).
USER_MESSAGE_LIMIT	Message send limit exceeded for a specific user.
WRONG_REQUEST	Something went wrong.

Related links:

- How to work with dialable keyboards
- How to work with attachments
- Message Formatting

Working with Dialogs

Attention! In order to use the IMOPENLINES REST methods, in addition to the rights to imbot (Creating and managing chatbots) have access to scope imopenlines (Open lines).

Method	Description of the method
imopenlines.dialog.get	Getting information about a dialog (chat) open line operator.

imopenlines.dialog.get

Obtaining information about the dialogue (chat) of the operator of the open lines

Revision: 2

Note! The method specified using the restCommand function is the send method data in Bitrix24, this method is in the EchoBot example, and is presented here as an example. You you can use your function or javascript method BX24.callMethod or bitrix24-php-sdk.

Options

Parameter	Example	Required	Description	Revision
CHAT_ID	13	No	Numerical identifier	2

			chat	
DIALOG_ID	chat29 or 256	No	Dialog ID. Format: **chatXXX** -- chat recipient if the message is for chat or **XXX** -- recipient ID if the message is for private dialogue	2
SESSION_ID	1743	No	Session ID within an open line	2
USER_CODE	livechat|1|1373|211	No	String identifier user open line from CRM example livechat|1|1373|211 or imol|livechat|1|1373|211	2

Can be used to call any of the options.

Method call and response

JavaScript

```
BX24.callMethod('imopenlines.dialog.get', {USER_CODE: 'livechat|1|1373|211'}, function(result){
    if(result.error())
    {
        console.error(result.error().ex);
    }
    else
    {
        console.log(result.data());
    }
});
```

PHP

```
$result = restCommand('imopenlines.dialog.get', Array(
    'DIALOG_ID': 'chat29'
), $_REQUEST["auth"]);
```

Response Example

```
{
  "result":
  {
    avatar:
    color: "#4ba984"
    date_create: "2020-05-12T17:40:55+02:00"
    dialog_id: null
    entity_data_1: "N|NONE|0|N|N|0|1591872180|1|0|"
    entity_data_2:
    entity_data_3:
    entity_id: "livechat|1|1363|203"
    entity_type: "LINES"
    extranet: false
    id: 1364
    manager_list: []
    message_type: "L"
    name: "Eugene Perekopsky - Priority Support"
    owner: 0
    type: "lines"
  }
}
```


Description of keys:

- avatar -- link to the avatar (if empty, no avatar has been set)
- color -- chat color in hex format
- date_create -- chat creation date in ATOM format
- dialog_id -- dialog ID
- entity_data_1 -- external data for chat
- entity_data_2 -- external data for chat
- entity_data_3 -- external data for chat
- entity_id -- external code for chat -- identifier
- entity_type -- external code for chat -- type
- extranet -- sign of participation in the chat of an external extranet user (true/false)
- id -- chat identifier
- manager_list -- list of operators
- message_type -- chat message type
- name -- open line name
- owner -- identifier of the user-owner of the chat
- type -- chat type (group chat, call chat, open line chat, etc.)

Example response when an error occurs


```
{
  "error": "DIALOG_ID_EMPTY",
  "error_description": "Dialog ID can't be empty"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
DIALOG_ID_EMPTY	Dialog ID not passed
ACCESS_ERROR	The current user does not have access rights to the dialog

Working with chatbots

Attention! In order to use the IMOPENLINES REST methods, in addition to the rights to imbot _____ (Creating and managing chatbots) have access to scope imopenlines (Open lines).

Method	Description of the method
imopenlines.bot.session.operator_	Switching the conversation to free operator.
imopenlines.bot.session.transfer_	Switching the conversation to specific operator.
imopenlines.bot.session.finish_	Ending the current session.
imopenlines.bot.session.message.send	Chatbot Send Method automatic message.

imopenlines.bot.session.operator

Switching the conversation to a free operator

Revision: 1

Attention! In order to use the IMOPENLINES REST methods, in addition to the rights to imbot _____ (Creating and managing chatbots) have access to scope imopenlines (Open lines).

Options

Parameter Example	Required	Description	Revision
CHAT_ID 12	**Yes**	Chat ID	1

Method call and response

PHP

```
$result = restCommand('imopenlines.bot.session.operator', Array(

    'CHAT_ID' => 12

), $_REQUEST["auth"]);
```

Response Example

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

Execution result: true or error.

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID not passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	No chat ID passed.
WRONG_CHAT	The specified chat is not controlled by a bot.
BOT_ID_ERROR	Incorrect chatbot ID.

imopenlines.bot.session.transfer

Switching the conversation to a specific operator

Revision: 1

Attention! In order to use the IMOPENLINES REST methods, in addition to the rights to imbot (Creating and managing chatbots) have access to scope imopenlines (Open lines).

Options

Parameter	Example	Required	Description	Revision
CHAT_ID	112	**Yes**	Chat ID	1
USER_ID	12	**Yes**	User ID, on which is being redirected	1
LEAVE	N	**Yes**	Y/N. If N is specified - the chatbot will not leave this chat after redirection and will be present until the user confirms	1

Note: *Instead of USER_ID, you can specify QUEUE_ID to switch to another open line.*

Method call and response

PHP

```
$result = restCommand('imopenlines.bot.session.transfer', Array(

    'CHAT_ID' => 112,
    'USER_ID' => 12,
    'LEAVE' => 'N'

), $_REQUEST["auth"]);
```

Response Example

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

Execution result: true or error.

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID not passed"
}
```

Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	No chat ID passed.
USER_ID_EMPTY	The user ID for which the forward the conversation.
WRONG_CHAT	Invalid user id specified or this user is a chatbot or extranet user.
BOT_ID_ERROR	Incorrect chatbot ID.

imopenlines.bot.session.finish

Ending the current session

Revision: 1

Attention! In order to use the IMOPENLINES REST methods, in addition to the rights to imbot (Creating and managing chatbots) have access to scope imopenlines (Open lines).

Options

Parameter	Example	Required	Description	Revision
CHAT_ID	112	**Yes**	Identifier chat	1

Method call and response

PHP

```
$result = restCommand('imopenlines.bot.session.finish', Array(

    'CHAT_ID' => 112

), $_REQUEST["auth"]);
```

Response Example

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

Execution result: true or error.

Example response when an error occurs

```
{
    "error": "CHAT_ID_EMPTY",
    "error_description": "Chat ID not passed"
}
```


Description of keys:

- error -- code of the error that occurred
- error_description -- a short description of the error that occurred

Possible error codes

The code	Description
CHAT_ID_EMPTY	No chat ID passed.
BOT_ID_ERROR	Incorrect chatbot ID.

imopenlines.bot.session.message.send

Sending an automatic message by the chatbot

Revision: 1

Attention! In order to use the IMOPENLINES REST methods, in addition to the rights to imbot (Creating and managing chatbots) have access to scope imopenlines (Open lines).

Options

Parameter	Example	Required	Description	Revision
CHAT_ID	2	**Yes**	Identifier chat, which takes the current operator	1
MESSAGE	message text	**Yes**	sent message	1
NAME	WELCOME	**Yes**	Message type: WELCOME - welcome message, DEFAULT - usual message. By default, empty meaning	1

Method call and response

PHP

```
$result = restCommand('imopenlines.bot.session.message.send', Array(

    'CHAT_ID' => 2,
    'MESSAGE' => 'message text',
    'NAME' => 'WELCOME'
), $_REQUEST["auth"]);
```

Response Example

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

Execution result: true or error.

Full list of Rest API methods

Platform version:

Method	Description of the method
imopenlines.revision.get	Getting information about API revisions.

Working with open lines:

Method	Description of the method
imopenlines.network.join	Connecting an open line by code.
imopenlines.network.message.add	Sending a message on behalf of open line to the selected user.

Working with dialogues:

Method	Description of the method
imopenlines.dialog.get	Getting information about a dialog (chat) of an open line operator.

Working with chatbots:

Method	Description of the method
imopenlines.bot.session.operator	Switching the conversation to a free operator.
imopenlines.bot.session.transfer	Switching the conversation to a specific operator.
imopenlines.bot.session.finish	Ending the current session.
imopenlines.bot.session.message.send	Chatbot Send Method automatic message.

Sending commands and renewing authorization

Interaction with PHP

As part of our course, we interact with ***Bitrix24*** using **PHP.** All examples are specified using the **restCommand functions:**

```
restCommand('imbot.message.add',
   Array( "DIALOG_ID" => $_REQUEST['data']['PARAMS']['DIALOG_ID'],
   "MESSAGE" => "[put=/search]Enter search string[/put]",
), $_REQUEST["auth"]);
```

Where:

The first parameter is the name of the API (imbot.message.add); **The second parameter** is the data passed to the API (Array(...)) **The third parameter** is the data for request authorization ($_REQUEST["auth"]).

Note: This function is used as an example. You can use your a function based on the interaction protocol, or javascript method BX24.callMethod, or partner development bitrix24-php-sdk. **restCommand**

function

```
/**
 * 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')
 * @param boolean $authRefresh - If authorize is expired, refresh token
 * @return mixed
 */
function restCommand($method, array $params = Array(), array $auth = Array(), $authRefresh = true)
```

```
{
    $queryUrl = "https://".$auth["domain"]."/rest/".$method; $queryData =
    http_build_query(array_merge($params, array("auth" => $auth["access_token"])));

    $curl = curl_init();

    curl_setopt_array($curl, array(
        CURLOPT_POST => 1,
        CURLOPT_HEADER => 0,
        CURLOPT_RETURNTRANSFER => 1,
        CURLOPT_SSL_VERIFYPEER => 1,
        CURLOPT_URL => $queryUrl,
        CURLOPT_POSTFIELDS => $queryData,
    ));

    $result = curl_exec($curl);
    curl_close($curl);

    $result = json_decode($result, 1);

    if ($authRefresh && isset($result['error']) && in_array($result['error'], array('expired_token', 'invalid_token')))


    { $auth = restAuth($auth);
        if ($auth)
        {
            $result = restCommand($method, $params, $auth, false);
        }
    }

    return $result;
}
```

function restAuth

```
/**
 * Get new authorize data if you authorize is expire.
 *
 * @param array $auth - Authorize data, ex: Array('domain' => 'https://test.bitrix24.com',
'access_token' => '7inpwszbuu8vnwr5jmabqa467rqur7u6')
 * @return bool|mixed
 */
function restAuth($auth)
{
    if (!CLIENT_ID || !CLIENT_SECRET)
        return false;

    if(!isset($auth['refresh_token']) || !isset($auth['scope']) || !isset($auth['domain']))
        return false;

    $queryUrl = 'https://'.$auth['domain'].'/oauth/token/';
```

```
    $queryData = http_build_query($queryParams = array(
        'grant_type' => 'refresh_token',
        'client_id' => CLIENT_ID,
        'client_secret' => CLIENT_SECRET,
        'refresh_token' => $auth['refresh_token'],
        'scope' => $auth['scope'],
    ));

    $curl = curl_init();

    curl_setopt_array($curl, array(
        CURLOPT_HEADER => 0,
        CURLOPT_RETURNTRANSFER => 1,
        CURLOPT_URL => $queryUrl.'?'.$queryData,
    ));

    $result = curl_exec($curl);
    curl_close($curl);

    $result = json_decode($result, 1);

    return $result;
}
```

Event subscription

Server event handler

The REST API interface allows you to set your own server event handlers. The event handler is **the URL** that will be called after the user clicks the requested action on the ***Bitrix24*** portal where the application is installed.

The handler receives the following data as input:

- **event** - the name of the fired event.
- **data** - array of event data.
- **auth** - a set of data for authorization.

The handler can:

- Use the received authorization data to make requests to the REST API.
- Be installed only by a user with portal administration rights.

Attention! If an event handler needs to call the Rest API to work with data, then it is strongly recommended to use authorization data stored on the application side for this, and not rely on obtaining an access_token, since it may not be.

When setting an event, you can specify the user under which the event handler will be authorized. By default, the event handler will be given the authorization of the user whose actions triggered the event.

The event handler will not be called immediately after the event is triggered, but after some time, load dependent.

The list of available events can be obtained using the events REST method.

Installing a handler

Setting the event handler is done:

- using the REST-method event.bind based on **restCommand:**

```
$handlerBackUrl = 'http://www.my-domain.ru/handler/';
$result = restCommand('event.bind', Array(
  'EVENT' => 'OnAppUpdate',
  'HANDLER' => $handlerBackUrl
), $_REQUEST["auth"]);
```

- or using the BX24.callBind function js libraries:

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

Getting a list of registered event handlers is done using the events.get REST method.

Unbinding a registered event handler is done using the event.unbind REST method or using the BX24.callUnbind function js libraries.

To access each event, when registering an application version, it must be requested access right corresponding to the event.

An application can install an arbitrary number of handlers for the same event, but all handlers must be installed with different user authorizations. Also, calling an event handler may depend on the access of the user whose authorization will be given to the handler. Event names are case insensitive. ***Attention! When an application***

is uninstalled or a new version is

installed, all event handlers installed by the application will be removed.

Install and update events

During the installation and update of the application, 2 events are used: ONAPPINSTALL and ONAPPUPDATE. Both events have the same data set, and differ only in the parameter - [PREVIOUS_VERSION] => 1, which indicates **the old version of the application.**

Application installation event ONAPPINSTALL

```
[data] => Array(
  [LANGUAGE_ID] = ru // Base language for the portal
  [VERSION] = 1 // Applications version
)
[auth] => Array(
    [access_token] => lh8ze36o8ulgrljbyscr36c7ay5sinva // Key to send requests to
REST service
    [scope] => imbot // Allowed access levels
    [domain] => b24.hazz // Bitrix24 portal domain where the
application
    [application_token] => c917d38f6bdb84e9d9e0bfe9d585be73 // The application token will
help you "beat off" unnecessary requests to the event handler, this field is in all events


    [expires_in] => 3600 // Token expiration time after which
request a new one
    [member_id] => d41d8cd98f00b204e9800998ecf8427e // Unique portal ID, required to renew
authorization
```

```
        [refresh_token] => 5f1ih5tsnsb11sc5heg3kp4ywqnjhd09 // Key to renew
    authorization
    )
```

Application update event ONAPPUPDATE

```
    [data] => Array(
      [LANGUAGE_ID] = ru // Base language for the portal
      [VERSION] = 2 // New application version
      [PREVIOUS_VERSION] => 1 // Old version apps
    )
    [auth] => Array(
        [access_token] => lh8ze36o8ulgrljbyscr36c7ay5sinva // Key to send requests to
    REST service
        [scope] => imbot // Allowed access levels
        [domain] => b24.hazz // Bitrix24 portal domain where the
    application
        [application_token] => c917d38f6bdb84e9d9e0bfe9d585be73 // The application token will
    help you "beat off" unnecessary requests to the event handler, this field is in all events
        [expires_in] => 3600 // Token expiration time after which a new one will need to be requested
        [member_id] => d41d8cd98f00b204e9800998ecf8427e // Unique portal ID, required to renew
    authorization
        [refresh_token] => 5f1ih5tsnsb11sc5heg3kp4ywqnjhd09 // Key to renew
    authorization
)
```

Note! In the basic version of working with a chatbot, the expires_in, member_id, refresh_token fields are not required. But, if it is necessary for your application, then you can read how to work with them here. The example bot contains the option to renew.

JS methods for iframe applications

Within iframe applications, you can use methods such as opening a dialog with users or controlling window sizes. The following functions exist for working with the web messenger: BX24.im.openMessenger - opening the messenger

- window.
- BX24.im.openHistory - opening the history window.
- BX24.im.callTo - internal call.
- BX24.im.phoneTo - a call to a telephone number.

Opening the messenger window

JavaScript-Method: BX24.im.openMessenger

Method call:

```
    <script>>
      BX24.im.openMessenger('dialogId');
    </script>
```

Options:

dialogId - Dialog ID:

- userId or chatXXX - chat, where XXX is the chat ID, it can be just a number.
- sgXXX - group chat, where XXX - social network group number (chat must be allowed in this group).
- imol|XXXX is an open line, where XXX is the code received via the imopenlines.network.join Rest method.

If nothing is passed, the chat interface will be opened with the last opened dialog.

Opening the history window

JavaScript method: BX24.im.openHistory

Method call:

```
<script>>
  BX24.im.openHistory('dialogId');
</script>
```

Options:

dialogId - Dialog ID:

- userId or chatXXX - chat, where XXX is the chat ID, it can be just a number.
- imol|XXXX - open line, where XXX is the open line session number.

Intercom call

JavaScript method: BX24.im.callTo

Method call:

```
<script>>
  BX24.im.callTo('userId[, video=true]');
</script>
```

Options:

dialogId - Portal user ID.
video - true - video call, false - audio call. Optional parameter.

Call to phone number

JavaScript method: BX24.im.phoneTo

Method call:

```
<script>>
  BX24.im.phoneTo('number');
</script>
```

Options:

number - Phone number (string). The number can be in the format: 84012112233 or 8 (495) 711-22-33.

Additional materials

These materials may be useful when developing chatbots for ***Bitrix24:***

- Examples of chatbots
- bitrix24-php-sdk

- Developer portal group
- Documentation on Rest
- API Training course "Marketplace Bitrix24"
- Video course "Marketplace Bitrix24"