Expand capabilities of amoCRM with APIs and widgets

Use APIs and widgets to obtain complete information about your clients. Integrate services and create widgets.

Widgets. Your services are visible to all users

 APIs. Become integrated with any types of systems and services

WEBhooks. Use an event model

Recent modifications

05 Nov. New capabilities for developers —[custom fields](https://developers.amocrm.ru/rest_api/fields_set.php) available only for APIs

17 Oct. [A new API for telephony](https://developers.amocrm.ru/rest_api/calls_set.php). We can track the emerging contacts ourselves

03 Oct. A new [WEBhooks](https://developers.amocrm.ru/rest_api/webhooks.php)-based event model

14 Sep. [The task result event](https://developers.amocrm.ru/rest_api/notes_set.php) appeared in event API

01 Sep. Now, widgets can work within the new FullAjax interface

Ready-made integrations

Expand capabilities with APIs

We strive to provide maximum opportunities for integration of amoCRM with other services and systems so that you, our clients and partners, could expand the opportunities of amoCRM for yourself and for other users (both yours and ours).

Writing your own widgets is not so difficult as it may seem at first. In order to make a developer’s life even easier, besides instructions, we also provide examples of using methods and have written a library for agile PHP-development.

Also, you can use the console to try working with APIs and test queries through an independent service.

What is widget?

If you only need to exchange data between amoCRM and external systems, then you can use REST API. This will allow you to receive, add and refresh data in amoCRM remotely.

In case your solution must be available to all clients of amoCRM, or you need to involve JS, interact with users within your browser and supplement system interfaces, all these capabilities are provided by widget system.

A widget is an array of settings, JS and PHP files, which can be added to any amoCRM account, when a client turns on a widget.

Practical examples of using widgets:

I want to output additional information about the contact (in the contact card) from my internal accounting system;

I want to provide employees of my company with an ability to send a request to the accounting department to form a payment document directly from the lead card;

I am a developer of an external service (telephony, direct e-mail), and I want to provide amoCRM clients with an ability to use my service by publishing a public widget and making integration easier and more transparent.

amoCRM glossary

| TERM | MEANING |
| --- | --- |
| Widget/ Add-on | An additional component for the system, which can be turned on and customized by the user for his account in order to extend functionality of amoCRM. A widget may contain JS file, settings file, PHP file, and images used in add-on's operation. |
| Languages/Localization | amoCRM is available in two languages: English (amocrm.com) and Russian (amocrm.ru). You may publish your widgets in both languages after indicating it clearly in the manifest. Don’t forget to make a translation and to consider time format |
| Manifest | A file, which must be uploaded with a widget, containing widget description and settings |
| Widget settings | Some widgets require to input settings during installation, e.g. account name in the external system, authorization parameters for this system, field settings, etc. |
| Account | A company’s amoCRM profile containing its payment details, billing information, proprietary data and the list of users, which will have access to these data. Each account has a subdomain in the system, e.g. company.amocrm.ru |

General rules of working with APIs

All communication with API is performed in encrypted form through the SSL protocol. It means that all links to API must contain the HTTPS protocol. It is particularly important to remember this fact in case of accessing through JS, when you are accessing some external resources, for example, WebSocket. Within the system, the user is always connected securely, and any attempt to access an HTTP image will be blocked, or the user’s browser will show a warning message.

All requests must be addressed not to the general domain http://www.amocrm.ru, but to the exact address of your account, for example, https://company.amocrm.ru or https://example.amocrm.ru.

Mechanisms for limiting the activity of working with API are provided – no more than one request per second. Also, some methods provide limitations on the amount of returned data for each data request (for more details, please, refer to descriptions of specific methods).

Main objects and methods

For a developer, amoCRM system is a relational database. The system contains main and auxiliary entities (in substance, data tables) that may be linked between each other. Access to them is provided through API.

Authorization

In order to access system data through both interfaces and APIs, user account authorization is required. All operations through API also take place with regard to access privileges of the user authorized in the account. All methods may be used only after authorization.

| API METHOD | DESCRIPTION |
| --- | --- |
| [POST /private/api/auth.php](https://developers.amocrm.ru/rest_api/auth.php) | In case of successful authorization, this method returns the user’s session ID, which must be used while accessing to other API methods. |

Account

Though API, you can obtain all necessary information about the account: name, paid period, account’s users and their privileges, guides on custom contact and lead fields, lead status guide, event types guide, task types guide and other account parameters.

| API METHOD | DESCRIPTION |
| --- | --- |
| [GET accounts/current](https://developers.amocrm.ru/rest_api/accounts_current.php) | Acquisition of information on the authorized account. |

Contact

This is one of the general entities of the system. It consists of a predefined set of fields and custom fields created by the account administrator. Each contact may take part in one or more leads, or may be not linked to any leads at all. Each contact can be linked with a single company.

The contact’s e-mail address and phone number are used as unique identifiers in association with other systems. For example, information about calls made, about e-mail communications falls exactly within the contact’s events.

A responsible user may be assigned to each contact in order to assign access privileges among the account’s personnel.

| API METHOD | DESCRIPTION |
| --- | --- |
| [POST contacts/set](https://developers.amocrm.ru/rest_api/contacts_set.php) | This method allows to add contacts one by one or in packages, and also to update data for the already existing contacts. |
| [GET contacts/list](https://developers.amocrm.ru/rest_api/contacts_list.php) | A method for obtaining contact lists with possibility of filtration and page-by-page selection. |
| [GET contacts/links](https://developers.amocrm.ru/rest_api/contacts_links.php) | A method for obtaining a list of links between leads and contacts. |

Lead

This is one of the general entities of the system. It consists of a predefined set of fields and custom fields created by the account administrator. Each lead can be linked to one or more contacts, or to none.

A responsible user may be assigned to each lead in order to assign access privileges among the account’s personnel.

Each lead has a status, which specifies the current position of the lead within its lifecycle (business process). Status must be necessarily assigned to any lead. The list of statuses within the account may be changed except for two final system statuses.

| API METHOD | DESCRIPTION |
| --- | --- |
| [POST leads/set](https://developers.amocrm.ru/rest_api/leads_set.php) | This method allows to create new leads, and also to update information on the already existing leads |
| [GET leads/list](https://developers.amocrm.ru/rest_api/leads_list.php) | This method allows obtaining detailed information on already created leads with possibility of filtration and page-by-page selection. |
| [GET contacts/links](https://developers.amocrm.ru/rest_api/contacts_links.php) | A method for obtaining a list of links between leads and contacts. |

Company

This entity is identical with “contact” entity. It consists of a predefined set of fields and custom fields created by the account administrator. Each company may take part in one or more leads, or may be not linked to any leads at all.

The e-mail address and phone number are used as identifiers in association with other systems.

A responsible user may be assigned to each company in order to assign access privileges among the account’s personnel.

| API METHOD | DESCRIPTION |
| --- | --- |
| [POST company/set](https://developers.amocrm.ru/rest_api/company_set.php) | This method allows to add companies one by one or in packages, and also to update data for the already existing companies. |
| [GET company/list](https://developers.amocrm.ru/rest_api/company_list.php) | A method for obtaining company lists with possibility of filtration and page-by-page selection. |

Task

Each task must necessarily have a responsible user and date (day and time) assigned to it. Also, a task may be linked to a lead or contact, but not necessarily. It may be not linked to any object.

It is possible to set a result for a completed task by means of [POST notes/set](https://developers.amocrm.ru/rest_api/notes_set.php) method

| API METHOD | DESCRIPTION |
| --- | --- |
| [POST tasks/set](https://developers.amocrm.ru/rest_api/tasks_set.php) | This method allows to create new tasks, and also to update information on the already existing tasks. |
| [GET tasks/list](https://developers.amocrm.ru/rest_api/tasks_list.php) | A method for obtaining a list of already created tasks with possibility of filtration and page-by-page selection. |

Event (note)

Events provide possibility of adding additional structured or unstructured information to a contact or a lead. Events may be of two types: system events (calls, SMS-messages, etc.) or user created events (notes, files). Events are displayed in cards along with tasks, since they are not linked to a responsible user or date.

Commonly, events are used by widgets for adding additional information to a lead or a contact, when it is inconvenient to use custom fields. Events can be easily used as a log, since they are always displayed in chronological order within the stream, and if your information is linked to a date (chronology), it is advisable to use events here.

| API METHOD | DESCRIPTION |
| --- | --- |
| [POST notes/set](https://developers.amocrm.ru/rest_api/notes_set.php) | This method allows to add events one by one or in packages , and also to update information on the already existing events. |
| [GET notes/list](https://developers.amocrm.ru/rest_api/notes_list.php) | A method for obtaining a list of events with possibility of filtration and page-by-page selection. |

Custom fields

This method allows creating or deleting custom fields one by one or in packages. The user will not be able to change their values from the interface, but can browse them and use them for filtering.

Creation or deletion of a field is also possible from the interface.

| API METHOD | DESCRIPTION |
| --- | --- |
| [POST fields/set](https://developers.amocrm.ru/rest_api/fields_set.php) | This method allows to add fields one by one or in packages, and also to delete them. |

Call

Calls provide possibility to add information to a client or company. Events are displayed in cards along with tasks, since they are not linked to a responsible user or date. If the call event has the link to the file containing the record of a telephone conversation, then a media player will be added to the note for playback of this record.

| API METHOD | DESCRIPTION |
| --- | --- |
| [POST calls/add](https://developers.amocrm.ru/rest_api/calls_set.php) | This method allows adding calls one by one or in packages. |

WebHooks

Each amoCRM account has ability to communicate with a web-server. These WebHooks can be used for updating information about leads in your web-store, sending SMS-notifications, or automation of lead conducting process. Each WebHook can be adjusted for certain operations and events. The account’s administrator can adjust hooks at “Settings->API” page.  
The description of WebHook mechanism can be found [here](https://developers.amocrm.ru/rest_api/webhooks.php).

Authorization through API

This method performs user authorization in the system. All API methods may be used only after authorization.

In case of successful login, a cookie file containing session key is returned in response to a request alongside with response body (as in the case of work with a web-browser). On further requests to API-methods, it is necessary to send the received cookies back. Session lifetime is 15 minutes.

All requests to API are made from the user, whose credentials were used during authorization through this method. At that, we take account of all of the user’s privileges, i.e. it is impossible to receive more data through API than the user can browse through the system’s interfaces. We recommend creating a separate user for API in order to allow more specific adjustment of privileges for plug-in application.

Parameters

| PARAMETER | DESCRIPTION |
| --- | --- |
| USER\_LOGINrequire | User login. E-mail is used as a login in the system. |
| USER\_HASHrequire | User key, which can be obtained at user profile editing page. |

The method also can accept an optional parameter for input.

| PARAMETER | DESCRIPTION |
| --- | --- |
| type | If type = json, then the response will be in the form of json array instead of XML document |

Example Request

| PARAMETER | DATA |
| --- | --- |
| POST | https://example.amocrm.ru/private/api/auth.php |
| POST DATA | USER\_LOGIN=example@amocrm.com& USER\_HASH=c123ae456cd7891246bffb1e654abb9d |

Example Response

<root>

  <auth>true <!-- (or “false” in case of error) --></auth>

</root>

You can use [API Console](https://developers.amocrm.ru/console.php) for testing. For testing of authorization, select Authentication = No Auth, API method Authentication. Enter your account’s subdomain on Template tab, and authorization parameters on Query tab.

In order to use authorization in API Console when calling other methods, it is necessary to select Authentication = Custom Token and transmit authorization credentials. USER\_HASH is an API-key of the user. You may specify type=json in the Custom Key/Value parameter, if necessary.

Example Code

First, we have to initialize data necessary for request compilation.

Account data

Obtaining information on the account used for authorization: name, paid period, account’s users and their privileges, guides on custom contact and lead fields, lead status guide, event types guide, task types guide and other account parameters.

Resource URL

/private/api/v2/json/accounts/current

Parameters

This method doesn’t accept parameters.

Response Field Guide

| PARAMETER | DESCRIPTION |
| --- | --- |
| id | A unique account identifier |
| name | Account name |
| subdomain | Unique subdomain of this account |
| currency | Account currency (used during operations with budget of leads.). Unrelated to billing information of the account itself. |
| paid\_from | Paid from (the response is received in timestamp format; if the account is unpaid – false) |
| paid\_till | Paid until (the response is received in timestamp format; if the account is unpaid – false) |
| timezone | Time zone |
| date\_pattern | Date format (see format description [here](https://developers.amocrm.ru/rest_api/accounts_current.php#dateformat)) |
| language | Account language. ru – Russian, en – English |
| users | List of account’s users and their privileges |
| users/id | Unique user’s identifier |
| users/name | User’s name |
| users/last\_name | User’s last name |
| users/rights\_lead\_add | User’s privileges for adding new leads (see the list of values [here](https://developers.amocrm.ru/rest_api/accounts_current.php#leadmap)) |
| users/rights\_lead\_view | User’s privileges for browsing the existing leads (see the list of values [here](https://developers.amocrm.ru/rest_api/accounts_current.php#leadmap)) |
| users/rights\_lead\_edit | User’s privileges for editing the existing leads (see the list of values [here](https://developers.amocrm.ru/rest_api/accounts_current.php#leadmap)) |
| users/ rights\_lead\_delete | User’s privileges for deleting the existing leads (see the list of values [here](https://developers.amocrm.ru/rest_api/accounts_current.php#leadmap)) |
| users/ rights\_lead\_export | User’s privileges for export of leads (see the list of values [here](https://developers.amocrm.ru/rest_api/accounts_current.php#leadmap)) |
| users/ rights\_contact\_add | User’s privileges for adding new contacts (see the list of values [here](https://developers.amocrm.ru/rest_api/accounts_current.php#leadmap)) |
| users/ rights\_contact\_view | User’s privileges for browsing the existing contacts (see the list of values [here](https://developers.amocrm.ru/rest_api/accounts_current.php#leadmap)) |
| users/ rights\_contact\_edit | User’s privileges for editing the existing contacts (see the list of values [here](https://developers.amocrm.ru/rest_api/accounts_current.php#leadmap)) |
| users/ rights\_contact\_delete | User’s privileges for deleting the existing contacts (see the list of values [here](https://developers.amocrm.ru/rest_api/accounts_current.php#leadmap)) |
| users/ rights\_contact\_export | User’s privileges for export of contacts (see the list of values [here](https://developers.amocrm.ru/rest_api/accounts_current.php#leadmap)) |
| leads\_statuses | List of lead statuses |
| leads\_statuses/name | Name of status |
| leads\_statuses/id | Unique identifier of a status |
| leads\_statuses/color | Color of a status |
| leads\_statuses/editable | Indicates possibility of changing this status (the response is in the form of "N" string – can not be edited through account settings, or "Y” string – can be edited) |
| custom\_fields | Contains custom fields for different entities |
| custom\_fields/contacts | Custom contact fields |
| custom\_fields/contacts/id | Unique identifier of an custom field |
| custom\_fields/contacts/name | Name of an custom field |
| custom\_fields/contacts/code | Field code. Set only for predefined fields |
| custom\_fields/contacts/disabled | Indicates possibility of field value editing from the interface (0 – possible, 1 – impossible; see [custom fields](https://developers.amocrm.ru/rest_api/fields_set.php)) |
| custom\_fields/contacts/multiple | Designations answering for existence of enum |
| custom\_fields/contacts/type\_id | Field type (see the list of values [here](https://developers.amocrm.ru/rest_api/accounts_current.php#type_id)) |
| custom\_fields/contacts/enums | List values |
| custom\_fields/leads | Custom lead fields (the structure is similar to contacts) |
| note\_types | The list of note types used in the system (see detailed description of types [here](https://developers.amocrm.ru/rest_api/notes_list.php#notetypes)) |
| note\_types/id | Unique note identifier |
| note\_types/name | Note name |
| note\_types/code | Code |
| note\_types/editable | Indicates possibility of note editing (the response is in the form of "N" string – can not be edited, or "Y" string – can be edited) |
| task\_types | Types of tasks available for this account |
| task\_types/id | Unique task identifier |
| task\_types/name | Task name |
| task\_types/code | Code (user only in standard tasks) |
| server\_time | Time stamp of the current server time with adjustment to the account’s time zone (transmitted in timestamp format) |

Data format

| PARAMETER | DESCRIPTION |
| --- | --- |
| D | Textual representation of weekday, 3 symbols |
| d | Day of month, 2 numbers with leading zero |
| M | Short name of month, 3 symbols |
| m | Order number of month with leading zero |
| Y | Order number of year, four numbers |
| H | Hours in 24-hour format with leading zero |
| i | Minutes with leading zero |
| s | Seconds with leading zero |

User privilege values

| CODE SYMBOL | DESCRIPTION |
| --- | --- |
| A | Permitted for all leads and contracts |
| M | Permitted only if the user is a responsible user |
| D | Forbidden |

You can use [API Console](https://developers.amocrm.ru/console.php) for testing. For testing, it is necessary to select Authentication = Custom Token and transmit authorization credentials. Enter your account’s subdomain on Template tab.

Example Code

First, you need to initialize data necessary for request compilation.

You have to initiate a request to server. Let’s use cURL library (included with PHP). You can also use a cross-platform cURL program, if you don’t work with PHP.

Guide on API errors & responses

In case of performance of incorrect request to the system, our API can return an error code. In case of correct request, API will return a response. Of course, you have already processed server response at debugging your widgets or writing scripts, which interact with our system. For your convenience, we have decided to systematize all possible responses and errors given by our system and to place them on a separate page. We hope that this is going to simplify and speed up integration of your projects with amoCRM.

Responses during work with an account

[See this page](https://developers.amocrm.ru/rest_api/#account) for detailed information on working with an account

| CODE | DESCRIPTION |
| --- | --- |
| 101 | Account not found |
| 102 | POST-parameters must be transmitted in JSON format |
| 103 | Parameters were not transmitted |
| 104 | The requested API method was not found |

Responses during work with contacts

[See this page](https://developers.amocrm.ru/rest_api/#contact) for detailed information on working with contacts

| CODE | DESCRIPTION |
| --- | --- |
| 201 | Adding contacts: empty array |
| 202 | Adding contacts: no privileges |
| 203 | Adding contacts: system failure during work with custom fields |
| 204 | Adding contacts: custom field not found |
| 205 | Adding contacts: contact was not created |
| 206 | Adding/updating contacts: empty request |
| 207 | Adding/updating contacts: incorrect method requested |
| 208 | Updating contacts: empty array |
| 209 | Updating contacts: “id” and “last\_modified” parameters are required |
| 210 | Updating contacts: system failure during work with custom fields |
| 211 | Updating contacts: custom field not found |
| 212 | Updating contacts: the contact was not updated |

Responses during work with leads

[See this page](https://developers.amocrm.ru/rest_api/#lead) for detailed information on working with leads

| CODE | DESCRIPTION |
| --- | --- |
| 213 | Adding leads: empty array |
| 214 | Adding/Updating leads: empty array |
| 215 | Adding/Updating leads: incorrect method requested |
| 216 | Updating leads: empty array |
| 217 | Updating leads: “id”, “last\_modified”, “status\_id”, “name” parameters needed |
| 240 | Adding/Updating leads: incorrect “id” parameter of custom field |

Responses during work with events

[See this page](https://developers.amocrm.ru/rest_api/#event) for detailed information on working with events

| CODE | DESCRIPTION |
| --- | --- |
| 218 | Adding events: empty array |
| 221 | Event list: type required |
| 222 | Adding/Updating events: empty request |
| 223 | Adding/Updating events: incorrect method requested |
| 224 | Updating events: empty array |
| 225 | Updating events: events not found |

Responses during work with tasks

[See this page](https://developers.amocrm.ru/rest_api/#tasks) for detailed information on working with tasks

| CODE | DESCRIPTION |
| --- | --- |
| 227 | Adding tasks: empty array |
| 228 | Adding/Updating tasks: empty request |
| 229 | Adding/Updating tasks: incorrect method requested |
| 230 | Updating tasks: empty array |
| 231 | Updating tasks: tasks not found |
| 232 | Adding events: element ID or element type is empty or incorrect |
| 233 | Adding events: some contacts for this element ID were not found |
| 234 | Adding events: some leads for this element ID were not found |
| 235 | Adding tasks: element type was not specified |
| 236 | Adding tasks: some contacts for this element ID were not found |
| 237 | Adding tasks: some leads for this element ID were not found |
| 238 | Adding contacts: custom field has no value |
| 244 | Adding leads: no privileges. |

Other responses

Miscellaneous errors and responses

| CODE | DESCRIPTION | NOTE |
| --- | --- | --- |
| 2002 | No results found for your request | HTTP-code #204 “No Content” is given with this response |

Adding and updating contacts

This method allows to add contacts one by one or in packages, and also to update data for the already existing contacts.

In order to obtain the list of links between contacts and leads, [GET contacts/links](https://developers.amocrm.ru/rest_api/contacts_links.php) method must be used.

Resource URL

POST /private/api/v2/json/contacts/set

Parameters

| PARAMETER | TYPE | DESCRIPTION |
| --- | --- | --- |
| add | Numerated array | The list of contacts to be added |
| update | Numerated array | Update of an existing contact  All parameters described in “add” also work in “update” |
| add/namerequire | String | Contact name |
| add/request\_id | Number | Unique identifier of a record in client-side program (optional parameter) |
| add/date\_create | timestamp | Date of creation of this contact (optional parameter) |
| add/last\_modified | timestamp | Date of the last modification of this contact (optional parameter) |
| add/responsible\_user\_id | Number | Unique identified of a responsible user (see users in [Account information](https://developers.amocrm.ru/rest_api/accounts_current.php)) |
| add/linked\_leads\_id | Numerated array | List of linked leads |
| add/linked\_leads\_id/ | Number | Lead ID |
| add/company\_name | String | Company name |
| add/custom\_fields | Numerated array | Custom field of contact |
| add/custom\_fields// | Numerated array | Description of each custom field will be inside |
| add/custom\_fields//id | Number | Unique identifier of the filled custom field (see [Account information)](https://developers.amocrm.ru/rest_api/accounts_current.php) |
| add/custom\_fields//values | Numerated array | Another array with “value” description will be inside |
| add/custom\_fields//values// | Numerated array | Values of the custom field and, if necessary, an additional type will be described here (for “multilist” type fields, we need only to specify IDs of the selected values) |
| add/custom\_fields//values//value | String | Custom field value |
| add/custom\_fields//values//enum | String | Selectable type of custom field (e.g. home/work phone, etc.) |
| add/tags | String | Tag names separated by commas. |
| update/idrequire | Number | Unique contact identifier, which is specified to update the contact |
| update/last\_modifiedrequire | timestamp | Date of the last modification of the current entity. If the parameter is empty, or it is less than the one in the database, then the update will not take place, and information from amoCRM Database will be sent in response (mandatory parameter) |

Adding contacts

To add a contact, it is necessary to describe an array containing information about the contact and place it into an array of the following form: $contacts['request']['contacts']['add'] Our API also supports simultaneous adding of several contacts. For this purpose, we must place several arrays into $contacts['request']['contacts']['add'] array, each describing the data necessary for creation of the corresponding contact.

Now, let’s prepare the data necessary for making request to the server

Updating contacts

In order to update information about contact, it is necessary to describe an array containing information about the contact and place it into an array of the following form: $contacts['request']['contacts']['update'] Our API also supports simultaneous updating of several contacts at one time. For this purpose, we must place several arrays into $contacts['request']['contacts']['update'] array, each describing the data necessary to update the corresponding contact.

List of contacts

This method allows obtaining a list of contacts with possibility of filtration and page-by-page selection. The limit of data returned within a single page (offset) is 500 contacts.

In order to obtain the list of links between contacts and leads, [GET contacts/links](https://developers.amocrm.ru/rest_api/contacts_links.php) method must be used.

Parameters

| PARAMETER | DESCRIPTION |
| --- | --- |
| if-modified-since (изменено с) | Mon, 22 Jul 2013 10:35:00 Data must be transmitted in “D, d M Y H:i:s” format through HTTP header |
| limit\_rows | Number of selected rows (system limit is 500) |
| limit\_offset | Selection offset (from which row to select) (Works only if limit\_rows parameter is also indicated) |
| id | Select an element with the specified ID (if this parameter is indicated, all other parameters will be ignored) (It is possible to transmit an array containing several IDs) |
| query | Searched element, by a text query (Performs search in such fields as e-mail, phone and others; does not perform search in notes and tasks) |
| responsible\_user\_id | Additional search filter, by a responsible user  (May be transmitted in the form of array) |
| type | Contact type: contact (by default), company or all |

Response Field Guide

| PARAMETER | DESCRIPTION |
| --- | --- |
| company\_name | Company name |
| type | Contact type: сontact or company |
| created\_user\_id | ID of the user, who has created the contact |
| linked\_leads\_id | An array of IDs of linked leads (leads themselves are located in “leads” element, which is on the same level as “contacts”) |
| tags | Tag array |
| tags/id | Unique tag identifier |
| tags/name | Tag name (it is also the text) |
| custom\_fields//id | Unique custom field identifier |
| custom\_fields//name | Custom field name |
| custom\_fields//code | Field code. It is set only for predefined fields |
| custom\_fields//values | An array of values of the current custom field |
| custom\_fields//values//id | Unique identifier of the value of the current custom field |
| custom\_fields//values//value | Value of the current custom field |
| custom\_fields//values//enum | Value of list |
| custom\_fields//values//last\_modified | Date of the last modification (transmitted in timestamp format) |
| server\_time | Time stamp of the current server time with adjustment to the account’s time zone (transmitted in timestamp format) |

Links between leads and contacts

This method allows obtaining a list of links between leads and contacts. Leads and contacts are linked as many-to-many, i.e. one lead can contain a lot of contacts, and one contact can be linked to many leads.

Resource URL

/private/api/v2/json/contacts/links

Parameters

| PARAMETER | DESCRIPTION |
| --- | --- |
| if-modified-since (изменено с) | Mon, 22 Jul 2013 10:35:00 Data must be transmitted in “D, d M Y H:i:s” format through HTTP-header |
| contacts\_link / deals\_link | Array of IDs of contacts and leads correspondingly  Used for obtaining lead IDs linked with the transmitted list of contact IDs (or vice versa) |
| limit\_rows | Number of selected rows (system limit is 500) |
| limit\_offset | Selection offset (from which row to select) (Works only if limit\_rows parameter is also indicated) |

Response Field Guide

| PARAMETER | DESCRIPTION |
| --- | --- |
| "contact\_id" | Unique identifier of a contact, which is linked with a lead |
| "lead\_id" | Unique identifier of a lead, which is linked with a contact |
| "last\_modified" | Date of the last modification of this entry in timestamp format |
| "server\_time" | Time stamp of the current server time with adjustment to the account’s time zone (transmitted in timestamp format) |

Adding and updating leads

This method allows to add leads one by one or in packages, and also to update data for the already existing leads.

In order to obtain the list of links between contacts and leads, [GET contacts/links](https://developers.amocrm.ru/rest_api/contacts_links.php) method must be used.

Resource URL

POST /private/api/v2/json/leads/set

Parameters

| PARAMETER | TYPE | DESCRIPTION |
| --- | --- | --- |
| add | Numerated array | The list of leads to be added |
| update | Numerated array | Updating the current leads  All parameters that are described in “add” also work in “update” |
| add/namerequire | String | Lead name |
| add/date\_create | timestamp | Date of creation of the current lead (optional parameter) |
| add/last\_modified | timestamp | Date of the last modification of the current lead (optional parameter) |
| add/status\_idrequire | Number | Lead status (for description of statuses, see [Account information](https://developers.amocrm.ru/rest_api/accounts_current.php)) |
| add/price | Number | Lead budget |
| add/responsible\_user\_id | Number | Unique identifier of a responsible user (see users in [Account information](https://developers.amocrm.ru/rest_api/accounts_current.php)) |
| add/request\_id | Number | Unique identifier of a record in client-side program (optional parameter) (information about request\_id will not be saved anywhere) |
| add/custom\_fields | Numerated array | Custom fields |
| add/custom\_fields// | Numerated array | Description of each custom field will be inside |
| add/custom\_fields//id | Number | Unique identifier of the filled custom field (see [Account information](https://developers.amocrm.ru/rest_api/accounts_current.php)) |
| add/custom\_fields//values | Numerated array | Another array with “value” description will be inside |
| add/custom\_fields//values// | Numerated array | Values of the custom field and, if necessary, an additional type will be described here (for “multilist” type fields, we need only to specify IDs of the selected values) |
| add/custom\_fields//values//value | String | Custom field value |
| add/custom\_fields//values//enum | String | Selectable type of custom field (e.g. home/work phone, etc.) |
| add/tags | String | Tag names separated by commas. |
| update/idrequire | Number | Unique lead identifier, which is specified to update the lead |
| update/last\_modifiedrequire | timestamp | Date of the last modification of the current entity. If the parameter is empty, or it is less than the one in the database, then the update will not take place, and information from amoCRM Database will be sent in response (mandatory parameter) |

Example Code

To add a lead, it is necessary to describe an array containing information about the lead and place it into an array of the following form: $leads['request']['leads']['add'] Our API also supports simultaneous adding of several leads. For this purpose, we must place several arrays into $leads['request']['leads']['add'] array, each describing the data necessary for creation of the corresponding lead.

Updating leads

In order to update lead data, it is necessary to describe an array containing information about the lead and place it into an array of the following form: $leads['request']['leads']['update'] Our API also supports simultaneous updating of several leads at one time. For this purpose, we must place several arrays into $leads['request']['leads']['update'] array, each describing the data necessary to update the corresponding lead.

List of leads

This method allows obtaining a list of leads with possibility of filtration and page-by-page selection. The limit of data returned within a single page (offset) is 500 leads.

In order to obtain the list of links between contacts and leads, [GET contacts/links](https://developers.amocrm.ru/rest_api/contacts_links.php) method must be used.

Resource URL

/private/api/v2/json/leads/list

Parameters

| PARAMETER | DESCRIPTION |
| --- | --- |
| if-modified-since (изменено с) | Mon, 22 Jul 2013 10:35:00 Data must be transmitted in “D, d M Y H:i:s” format through HTTP-header |
| limit\_rows | Number of selected rows (system limit is 500) |
| limit\_offset | Selection offset (from which row to select) (Works only if limit\_rows parameter is also indicated) |
| id | Select an element with the specified ID (if this parameter is indicated, all other parameters will be ignored) (It is possible to transmit an array containing several IDs) |
| query | Searched element, by a text query (Performs search in such fields as e-mail, phone and others; does not perform search in notes and tasks) |
| responsible\_user\_id | Additional search filter, by a responsible user  (May be transmitted in the form of array) |
| status | Filter by lead status ID (See [this page](https://developers.amocrm.ru/rest_api/accounts_current.php) to know the list of available IDs) (May be transmitted in the form of array) |

Response Field Guide

| PARAMETER | DESCRIPTION |
| --- | --- |
| id | Unique lead identifier |
| name | Lead name |
| created\_user\_id | ID of the user, who has created the lead |
| date\_create | Date of creation (timestamp) |
| last\_modified | Date of the last modification of this lead in timestamp format |
| status\_id | Unique identifier of lead status |
| price | Budget |
| responsible\_user\_id | Unique identifier of a responsible user |
| account\_id | Unique account identifier |
| tags | Tag array |
| tags/id | Unique tag identifier |
| tags/name | Tag name (it is also the text) |
| custom\_fields | Custom fields |
| custom\_fields/id | Unique identifier of the custom field |
| custom\_fields/name | Custom field name |
| custom\_fields/values | Array of values of the current custom field |
| custom\_fields/values/ value | Custom field value |
| server\_time | Time stamp of the current server time with adjustment to the account’s time zone (transmitted in timestamp format) |

Adding and updating companies

This method allows to add companies one by one or in packages, and also to update data for the already existing companies.

In order to obtain the list of links between contacts, companies and leads, [GET company/links](https://developers.amocrm.ru/rest_api/contacts_links.php) method must be used.

Resource URL

POST /private/api/v2/json/company/set

Parameters

| PARAMETER | TYPE | DESCRIPTION |
| --- | --- | --- |
| add | Numerated array | The list of companies to be added |
| update | Numerated array | Updating the current leads  All parameters that are described in “add” also work in “update” |
| add/namerequire | String | Company name |
| add/request\_id | Number | Unique identifier of a record in client-side program (optional parameter) |
| add/date\_create | timestamp | Date of creation of this company (optional parameter) |
| add/last\_modified | timestamp | Date of the last modification of this company (optional parameter) |
| add/responsible\_user\_id | Number | Unique identifier of a responsible user (see users in [Account information](https://developers.amocrm.ru/rest_api/accounts_current.php)) |
| add/linked\_leads\_id | Numerated array | The list of linked leads |
| add/linked\_leads\_id/ | Number | Lead ID |
| add/custom\_fields | Numerated array | Custom fields of the company |
| add/custom\_fields/ | Numerated array | Description of each custom field will be inside |
| add/custom\_fields//id | Number | Unique identifier of the filled custom field (see [Account information)](https://developers.amocrm.ru/rest_api/accounts_current.php) |
| add/custom\_fields//values | Numerated array | Another array with “value” description will be inside |
| add/custom\_fields//values/ | Numerated array | Values of the custom field and, if necessary, an additional type will be described here |
| add/custom\_fields//values//value | String | Custom field value |
| add/custom\_fields//values//enum | String | Selectable type of custom field (e.g. home/work phone, etc.) |
| add/tags | String | Tag names separated by commas. |
| update/idrequire | Number | Unique company identifier, which is specified to update the company |
| update/last\_modifiedrequire | timestamp | Date of the last modification of the current entity. If the parameter is empty, or it is less than the one in the database, then the update will not take place, and information from amoCRM Database will be sent in response (mandatory parameter) |

List of companies

Working with companies is completely the same as working with contacts (except for different sets of user fields and absence of company\_name parameter)

This method allows obtaining a list of companies with possibility of filtration and page-by-page selection. The limit of data returned within a single page (offset) is 500 companies.

In order to obtain the list of links between companies and leads, [GET contacts/links](https://developers.amocrm.ru/rest_api/contacts_links.php) method must be used.

Resource URL

/private/api/v2/json/company/list

Parameters

| PARAMETER | DESCRIPTION |
| --- | --- |
| if-modified-since (изменено с) | Mon, 22 Jul 2013 10:35:00 Data must be transmitted in “D, d M Y H:i:s” format through HTTP header |
| limit\_rows | Number of selected rows (system limit is 500) |
| limit\_offset | Selection offset (from which row to select) (Works only if limit\_rows parameter is also indicated) |
| id | Select an element with the specified ID (if this parameter is indicated, all other parameters will be ignored) (It is possible to transmit an array containing several IDs) |
| query | Searched element, by a text query (Performs search in such fields as e-mail, phone and others; does not perform search in notes and tasks) |
| responsible\_user\_id | Additional search filter, by a responsible user  (May be transmitted in the form of array) |

Response Field Guide

| PARAMETER | DESCRIPTION |
| --- | --- |
| created\_user\_id | ID of the user, who has created the contact |
| linked\_leads\_id | An array of IDs of linked leads (leads themselves are located in “leads” element, which is on the same level as “contacts”) |
| tags | Tag array |
| tags/id | Unique tag identifier |
| tags/name | Tag name (it is also the text) |
| custom\_fields/id | Unique identifier of the custom field |
| custom\_fields/name | Custom field name |
| custom\_fields/code | Field code. It is set only for predefined fields |
| custom\_fields/values | Array of values of the current cusom field |
| custom\_fields/values/id | Unique identifier of the value of the current custom field |
| custom\_fields/values/value | Value of the current custom field |
| custom\_fields/values/enum | Value of list |
| custom\_fields/values/last\_modified | Date of the last modification (transmitted in timestamp format) |
| server\_time | Time stamp of the current server time with adjustment to the account’s time zone (transmitted in timestamp format) |

Adding and updating tasks

This method allows to add a new task or to modify an existing task. To set the result for the task, use [POST notes/set](https://developers.amocrm.ru/rest_api/notes_set.php) method (by setting the corresponding [type of note](https://developers.amocrm.ru/rest_api/notes_list.php)

Resource URL

POST /private/api/v2/json/tasks/set

Parameters

| PARAMETER | TYPE | DESCRIPTION |
| --- | --- | --- |
| add | Numerated array | The list of tasks to be added |
| update | Numerated array | Modification of existing tasks  All parameters that are described in “add” also work in “update” |
| add/element\_idrequire | Number | Unique identifier of the contact or lead (contact/lead must be indicated in element\_type) |
| add/element\_typerequire | Number | Type of element to be linked (1 - contact, 2 - lead) |
| add/date\_create | timestamp | Date of creation of this task (optional parameter) |
| add/last\_modified | timestamp | Date of the last modification of this task (optional parameter) |
| add/request\_id | Number | Unique identifier of a record in client-side program (optional parameter) (information about request\_id will not be saved anywhere, it will only be transmitted within a response) |
| add/task\_typerequire | Number | Task type (see task types in [Account information](https://developers.amocrm.ru/rest_api/accounts_current.php)) |
| add/textrequire | String | Text of task |
| add/responsible\_user\_id | Number | Unique identifier of a responsible user (see users in [Account information](https://developers.amocrm.ru/rest_api/accounts_current.php)) |
| add/complete\_tillrequire | timestamp | A date by which the task must be completed. If time has value 23:59, then “All day” will be displayed in system interfaces instead of time. |
| update/idrequire | Number | Unique identifier of the updated task |
| update/last\_modifiedrequire | timestamp | Date of the last modification of the current entity. If the parameter is empty, or it is less than the one in the database, then the update will not take place, and information from amoCRM Database will be sent in response (mandatory parameter) |

List of tasks

This method allows obtaining a list of tasks with possibility of filtration and page-by-page selection. The limit of data returned within a single page (offset) is 500 tasks.

Expired tasks are the tasks that have complete\_till parameter less than the current time, and the “unclosed” status.

Resource URL

/private/api/v2/json/tasks/list

Request type

GET

Parameters

| PARAMETER | DESCRIPTION |
| --- | --- |
| if-modified-since (изменено с) | Mon, 22 Jul 2013 10:35:00 Data must be transmitted in “D, d M Y H:i:s” format through HTTP header |
| type | contact or lead  Obtaining data only for contact or lead Without this parameter, all the data will be obtained including the data unassigned to any objects |
| limit\_rows | Number of selected rows (system limit is 500) |
| limit\_offset | Selection offset (from which row to select) (Works only if limit\_rows parameter is also indicated) |
| id | Select an element with the specified ID (if this parameter is indicated, all other parameters will be ignored) (It is possible to transmit an array containing several IDs) |
| responsible\_user\_id | Additional search filter, by a responsible user  (May be transmitted in the form of array) |

Response Field Guide

| PARAMETER | DESCRIPTION |
| --- | --- |
| id | Unique task identifier |
| element\_id | Unique identifier of linked lead/contact |
| element\_type | Type of linked element: 1 - contact, 2 – lead |
| task\_type | Task type: "LETTER" , "MEETING" , "CALL" , "OTHER" |
| created\_user\_id | ID of the user, who has created the task |
| date\_create | Date of creation of this note in timestamp format |
| last\_modified | Date of the last modification of this note in timestamp format |
| text | Textual representation of the task |
| responsible\_user\_id | Unique identifier of a responsible user |
| complete\_till | A date by which this task must be completed (in timestamp format) |
| status | Task status designation: 0 – uncompleted 1 – completed |
| account\_id | Unique account identifier |
| server\_time | Time stamp of the current server time with adjustment to the account’s time zone (transmitted in timestamp format) |

Adding and updating events

This method allows to add new events or to update the existing events.

Resource URL

POST /private/api/v2/json/notes/set

Parameters

| PARAMETER | TYPE | DESCRIPTION |
| --- | --- | --- |
| add | Numerated array | The list of events to be added |
| update | Numerated array | Modification of existing events All parameters that are described in “add” also work in “update” |
| add/element\_idrequire | Number | Unique identifier of contact or lead (contact/lead must be indicated in element\_type) |
| add/element\_typerequire | Number | Type of element to be linked (contact or lead) |
| add/note\_typerequire | Number | Event type (see [Events list](https://developers.amocrm.ru/rest_api/notes_list.php)) |
| add/date\_create | timestamp | Date of creation of the event (optional parameter) |
| add/request\_id | Number | Unique identifier of a record in client-side program (optional parameter) (information about request\_id will not be saved anywhere) |
| add/last\_modified | timestamp | Date of the last modification of the event (optional parameter) |
| add/textrequire | String | Text of task |
| add/responsible\_user\_id | Number | Unique identifier of a responsible user (see users in [Account information](https://developers.amocrm.ru/rest_api/accounts_current.php)) |
| update/idrequire | Number | Unique event identifier, which is specified to update the event |
| update/last\_modifiedrequire | timestamp | Date of the last modification of the current entity. If the parameter is empty, or it is less than the one in the database, then the update will not take place, and data of the old entity will be sent in response (mandatory parameter) |
| update/id | Number | Unique event identifier, which is specified to update the event |
| update/last\_modified | timestamp | Date of the last modification of the current entity. If the parameter is empty, or it is less than the one in the database, then the update will not take place, and information from amoCRM Database will be sent in response (mandatory parameter) |

List of notes

This method allows obtaining a list of notes with possibility of filtration and page-by-page selection. The limit of data returned within a single page (offset) is 500 notes.

Resource URL

/private/api/v2/json/notes/list

Request type

GET

Parameters

| PARAMETER | DESCRIPTION |
| --- | --- |
| typerequire | contact or lead  Obtaining data only for contact or lead  (mandatory parameter) |
| if-modified-since (изменено с) optional | Mon, 22 Jul 2013 10:35:00 Data must be transmitted in “D, d M Y H:i:s” format through HTTP header |
| limit\_rows | Number of selected rows (system limit is 500) |
| limit\_offset | Selection offset (from which row to select) (Works only if limit\_rows parameter is also indicated) |
| id | Select an element with the specified ID (if this parameter is indicated, all other parameters will be ignored) (It is possible to transmit an array containing several IDs) |
| element\_id | Unique identifier of contact or lead |

Response Field Guide

| PARAMETER | DESCRIPTION |
| --- | --- |
| id | Unique note identifier |
| element\_id | Unique identifier of the linked lead/contact |
| element\_type | Type of linked element: 1 - contact, 2 - lead |
| note\_type | Note type (see the table of note types [here](https://developers.amocrm.ru/rest_api/notes_list.php#notetypes)) |
| created\_user\_id | ID of the user, who has created the note |
| date\_create | Date of creation of the current note in timestamp format |
| last\_modified | Date of the last modification of this note in timestamp format |
| text | Textual representation of the task (outputs different information depending on types, see [this page](https://developers.amocrm.ru/rest_api/notes_list.php#notetext)) |
| responsible\_user\_id | Unique identifier of a responsible user |
| account\_id | Unique account identifier |
| editable | Indicates possibility of editing this note: "Y" or "N" |
| server\_time | Time stamp of the current server time with adjustment to the account’s time zone (transmitted in timestamp format) |

Types of notes

| NOTE TYPE | DESCRIPTION |
| --- | --- |
| 1 | Lead created |
| 2 | Contact created |
| 3 | Lead status modified |
| 4 | Standard note |
| 5 | File |
| 6 | Call made from iPhone applications |
| 7 | Letter |
| 8 | Letter with file |
| 9 | Not used |
| 10 | Inbound call |
| 11 | Outbound call |
| 12 | Company created |
| 13 | Task result |
| 102 | Inbound SMS |
| 103 | Outbound SMS |

Different information is formed in “text” depending on note\_type

| NOTE TYPE | TEXT | DESCRIPTION |
| --- | --- | --- |
| 4 | Talked with: Bob 03.06.2011 19:57 | Hi there. Need any help? John 03.06.2011 19:57 | Yes, please help me with billing." | Standard text written to a note |
| 5 | iCWp7a\_otKA.jpg | Name of file with extension |
| 7 | "a:5:{"s": 3:"mid";i:171634;s:4: "from";s:17:"whoisit@qsoft.ru ";s:7:"subject";s:52:" One more message to the contact";s:14:" attach\_limited";b:0;s:10:"attach\_cnt" ;i:5;} | “Text” contains a serialized array, which also contains “subject” and “mid” |
| 10-11 | "\"UNIQ\": \"234234.2374\", \"LINK\": \"/private/acceptors/asterisk\_new/?GETFILE=1124355.2374\", \"PHONE\": "11122333", \"DURATION\": \"175\", \"SRC\": \"asterisk\"" | Text contains an escaped JSON. More details on using this type are available at [IP telephony page](https://developers.amocrm.ru/examples/ip_ats.php) |
| 102-103 | "text": "{"TEXT":"FromWeb"}", | Text contains JSON, containing one field – “TEXT” |

Creation and deletion of custom fields

This method allows creating or deleting custom fields one by one or in packages. The user will not be able to change their values from the interface, but can browse them and use them for filtering.

Creation or deletion of a field is also possible from the interface.

Resource URL

POST /private/api/v2/json/fields/set

Parameters

| PARAMETER | TYPE | DESCRIPTION |
| --- | --- | --- |
| add | Numerated array | The list of fields to be added |
| delete | Numerated array | The list of fields to be deleted |
| add/namerequire | String | Field name |
| add/request\_id | Number | Unique identifier of a record in client-side program (optional parameter) (information about request\_id will not be saved anywhere) |
| add/typerequire | Number | Field type (see [Account information](https://developers.amocrm.ru/rest_api/accounts_current.php#type_id)) |
| add/element\_typerequire | Number | Entity |
| add/originrequire | String | Unique identifier of the service, which will allow deletion or modification of the field |
| delete/idrequire | Number | Unique field identifier, which is indicated for deleting the field |
| delete/originrequire | String | Unique identifier of the service, which will allow deletion or modification of the field |

Entities

| PARAMETER | DESCRIPTION |
| --- | --- |
| 1 | Contact |
| 2 | Lead |
| 3 | Company |

Response Field Guide

| PARAMETER | DESCRIPTION |
| --- | --- |
| id | Unique identifier of the new/deleted field |
|  |  |
| request\_id | Unique identifier of the entity in client-side program. If request\_id was not transmitted within the request, it will be generated automatically. |
| server\_time | Time stamp of the current server time with adjustment to the account’s time zone (transmitted in timestamp format) |

WebHooks

WebHooks is a notification of external applications by means of sending notifications about events that took place in amoCRM. You can configure HTTP-addresses of your applications and their associated operating rules in your account’s settings (API section).

Examples of scripts

After successful completion of a lead, you can send information about the occurred transaction to your money accounting application and automatically generate a bill.

You can add e-mails of new contacts in CRM system to a mailing list (e.g. UniSender).

You can configure SMS notifications about the changes made in your account.

List of supported entities

Lead

Contact

Company

List of possible events

Adding

Modification

Deletion

Description of parameters in settings

| FIELD NAME | DESCRIPTION |
| --- | --- |
| Name | Sets the name for the WebHook. |
| URL | Sets http-address of the external application |
| EVENT | Sets the list of events that will initiate WebHook sending to the indicated URL. |

WebHook installation

WebHook installation includes three steps as follows:

Adding a WebHook

Associating the WebHook with work processes.

Testing integration.

To create a WebHook

Go to “Settings -> API” menu.

Click “Add WebHook” in “WebHooks” section.

Enter URL for this WebHook.

Choose events, which will initiate notification sending.

Click “Save”.

To test integration

Make an action chosen during WebHook’s creation.

Check data received from amoCRM in your application.

If the no data was received, check correctness of the entered URL and move to item 1.

In which format the data are sent?

WebHook sends all information about the entity to the external application in format described in GET requests of the current section. POST variable contains an array of the following form: {“entity”:{“action”:{array of fields of entity}}} in case of creating and updating the entity, and also {“entity”:{“action”:”id ”}} in case of deleting the entity.

Expected response

At request sending, the information is deemed accepted if code 200 is returned in the http header of the response (according to the table of status codes [w3.org](http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html)).

First attempt to send a request takes place immediately after making the selected action. In case the attempt is unsuccessful, one more sending attempt will take place in accordance with the rules shown in the table below.

If the seventh attempt will be unsuccessful, sending failure notification will be sent to the e-mail indicated in settings.

Getting started with widgets

Widget is an archive containing a set of files, which will connect to all accounts in amoCRM that have turned this widget on. Widget allows obtaining additional functionality, if necessary:

Displaying additional data in amoCRM interfaces. Special [areas](https://developers.amocrm.ru/widgets/areas.php), where you can output information, are provided for widgets. For example, you can output contact accessing statistics from the internal system;

Interacting with a user, or with data entered by a user. You can enable JS-scripts in almost any system interface. For example, it is possible to show a pop-up card during incoming call;

Enabling amoCRM account administrator to enter individual settings for your service (for example, authorization key for your API).

In all other cases, you can simply use the opened amoCRM API.

First, we will try to create a fully operable widget example step-by-step and upload it to our account.

[Account registration](https://developers.amocrm.ru/widgets/index.php#registration)

[Widget key generation](https://developers.amocrm.ru/widgets/index.php#generation)

[Downloading widget archive with PHP-library and widget example](https://developers.amocrm.ru/widgets/index.php#download)

[Preparing structure](https://developers.amocrm.ru/widgets/structure.php)

[Working with manifest.json](https://developers.amocrm.ru/widgets/structure.php)

[Preparing localization files](https://developers.amocrm.ru/widgets/structure.php)

[JS-script development](https://developers.amocrm.ru/widgets/script_js.php)

[Packing and uploading our archive](https://developers.amocrm.ru/widgets/upload.php)

We will explore an option without the use of PHP-library, using JavaScript only. You can read about working with PHP in [PHP-library](https://developers.amocrm.ru/widgets/php_lib.php) section.

1. Account registration

You will have to create an account. If you already have an account, please, use caution during widget testing – don’t damage your data due to debugging. Perhaps you should create a separate account for beta-testing your widget.

After widget uploading, it will only be available in your account before our employees moderate it.

For my first widget, I will create an account with the following parameters:

| PARAMETER | VALUE |
| --- | --- |
| User email: | email@amocrm.com |
| Account address: | demoaccount.amocrm.com |

2. Widget key generation

When you’re in the account, go to /settings/dev/ page. This is a developer’s page, and for me, its URL will look like https://demoaccount.amocrm.com/settings/dev/. Here, you can generate your first key, download widgets, and see the list of downloaded widgets.

Enter widget code (only lower case letters!). Widget code will not be visible to end-users (only inside the code as an identifier). In response, you will get a unique key for your widget, which will be used in future.

Here’s an example of what you will get:

| PARAMETER | VALUE |
| --- | --- |
| Code: | new\_widget |
| Key: | 57009cb5048a72191f25b01355c17d10dc349df20d4fe2ad0c69930223e13955 |

3. Downloading widget example.

You can always download the recent version at the developer’s page, or you can click on the link below.

Widget structure

After unpacking the archive, you will see files, PHP-libraries and “widget\_example” folder in the root. We will take a look at the structure of “widget\_example” folder:

| FILE | DESCRIPTION |
| --- | --- |
| manifest.jsonrequire | File in json format containing widget description, widget settings, widget parameters shown to a user, and widget localizations. |
| script.js | JS-file, which will be connected at the user’s side in the areas indicated in manifest.json |
| widget.php | Optional file containing business logic in PHP, if necessary. Hosting of php-scripts on the side of amoCRM is available only for public widgets (widgets available to all amoCRM users) |
| images/ | Folder for placement of image files used in the widget. It must contain 3 files (png, jpeg, jpg or gif format), which will be used as widget logo in different scopes. The size of each file must not exceed 300 kilobytes.  logo.png is used on the widget settings page, logo\_min.png and logo\_medium.png are used in all lists and contact/lead cards in minimized and maximized state correspondingly.   |  |  |  | | --- | --- | --- | | https://developers.amocrm.ru/images/widget_logos/logo.png logo.png 130px x 100px | https://developers.amocrm.ru/images/widget_logos/logo_medium.png  logo\_medium.png 120px x 42px | https://developers.amocrm.ru/images/widget_logos/logo_min.png  logo\_min.png 42px x 42px | |
| i18n/ | Folder containing localization files in key:value format. Currently, only two localizations can be used: Russian (ru) or English (en). All translations will be available in both JS and PHP parts. |

As you can see, the only mandatory file is manifest.json

4. Starting development

As an example, we will use a widget that that is based on JS only. It outputs a button in the contact card. When it is clicked, data from the card are sent to the external system, where they are processed by php-script (only as an example), a stub that may be written in any language.

This is a common task, because transmitting data from amoCRM by employee’s action to the internal system of the company (or vice versa, outputting additional data from the external system in amoCRM cards) is often required according to business process.

Besides, amoCRM API supports the event model for calling external scripts through WebHooks mechanism. It allows calling URL-script (indicated in settings) and transmitting the latest data to it, when certain events take place (for example, contact modification, or change of lead status). For detailed information about WebHooks, please, refer to [the corresponding section.](https://developers.amocrm.ru/rest_api/webhooks.php)

OK. First, we need to copy the folder with widget example and name it as “widget”. This is a basis for our future widget.

We will not use widget.php for now, so you can easily delete it. Now, let’s start analyzing the remaining files, one after another.

5. manifest.json

Let’s start editing the file in accordance with the table of descriptions of its parameters. You may use a link to language messages as the value, if necessary.

| PARAMETER | DESCRIPTION |
| --- | --- |
| widget | The unit contains all basic settings of the widget |
| widget/name | Widget name, including for the purpose of displaying in the widget list. In the example, widget.name value is assigned, which means that the value will be taken from the corresponding file from “i18n” folder depending on localization. If you use only one language, you may enter the name at once. |
| widget/description | Full widget description, it is shown in widget settings window. You can use html-tags as well as special short-tags in order to form maximally personified description. For example, you need to show the description by giving user a hint with the subdomain of his amoCRM account. Tag list:  #HOST# - the current host;  #SUBDOMAIN# - account’s subdomain;  #LOGIN# - login of the current user, authorized user;  #API\_HASH# - hash-key of the user;  #ACCOUNT\_ID# - ID of the current account in the system;  #USER\_ID# - ID of the current user in the system. |
| widget/short\_description | Short description of output in the list of widgets |
| widget/code | Your widget code, which you have entered in Latin symbols on the developer’s page during key generation |
| widget/secret\_key | Widget secret key, which you have generated on the developer’s page |
| widget/version | Widget version. It is of informative nature only. |
| widget/interface\_version | (int) Interface version (1,2). This indicates, for which system interface the widget is downloaded. It is necessary to leave “2” by default. Interface “1” is the previous interface version of the whole amoCRM system, which doesn’t use AJAX. Registration for the old version is now closed. |
| widget/init\_once | (true/false) Indicates the necessity to collect js-object widget for 1 time during the work session with amoCRM interface. At widget initialization, render(), init() and bind\_actions() functions are called (for more information about these functions, see [script.js](https://developers.amocrm.ru/widgets/script_js.php) part). “true” or “false” indication regulates the ability to call init() and bind\_actions() functions during each transition from one area to another ([more information on connection areas](https://developers.amocrm.ru/widgets/areas.php)), or to call them only once. For example, telephony widgets are always holding WebSocket connection, and it must not be interrupted, therefore init\_once must have “true” value. In case there is no context common to all pages, it is better to select “false” value. |
| widget/locale | An array of localizations, in which the widget is available. For widget publication in English, the developer must provide support in English. |
| Widget/installation | (true/false) If “false”, widget will only be displayed in the widget list without requesting configuration or installation in the account. This may be of use, when all configurations take place in another system interacting with amoCRM through API. |
| locations | Interfaces that must display the widget. It is necessary to fill the array in case it is necessary to use JS part of the widget. More information on [areas](https://developers.amocrm.ru/widgets/areas.php). Possible locations:  “llist” — lead list, “clist” — contact list, “tlist” — task list, “lcard” — lead card, “ccard” — contact card.  Also, it is necessary to inform out system if the widget will use the right column for displaying. It can be done by writing 0 or 1 after specifying the zone  I.e. if you specify "clist-0", the widget will be initialized in this zone, but will not use the right column.  For example, WEB-background panel is located not in the right column of widgets, but at the bottom (in any interface). Therefore, “0” must be written in widget settings for all connection locations, at that, it is initialized on each page. |
| settings | An array of widget settings available only to the user, i.e. fields that will be displayed in widget settings window and filled by the user. This section is required only if installation = true; if installation = false, this section is not required, because only widget description will be displayed in settings window. Field code “FIELD\_CODE” will be a key in the array. |
| settings/FIELD\_CODE/name | Field name (only a link to an element in lang file) |
| settings/FIELD\_CODE/type | Field type. Available options:  text  pass  list  users (the list of system users with 1 text field for each; it is required in case it is necessary to input some information on each employee, for example internal phone number for IP telephony)  users\_lp (the list of system users with 2 fields (login,password) for each user).  These types of fields are described in more details in the section [Field types in “settings manifest.json](https://developers.amocrm.ru/widgets/fields.php)” |
| settings/FIELD\_CODE/required | true/false; the necessity of filling the field by the user. |
| settings/FIELD\_CODE/enums | The list of possible options for types “list” in Name-Value format. |

Example of manifest.json

6. i18n Localization files

As you may have noticed, widget.name constructions are used in manifest.json example. They are required if your widget is supposed to work in more than one language.

In this case, you have to create to localization files in “i18n” folder, one for Russian (ru.json) and one for English (en.json). After that, depending on account language, messages in the corresponding language will be available in both JS and PHP parts.

IMPORTANT: If 2 localizations are used, it is necessary for files to have similar structure.

i18n/ru.json

In order to call language messages from widget’s JS code, use self.i18n('obj\_name') method, where obj\_name is one of the objects of localization file.

Examples of manifest.json file are presented in the section [Field types of “settings manifest.json](https://developers.amocrm.ru/widgets/fields.php)”

Common mistakes

Since the most part of files (and manifest.json in particular) have json format, it is better to ensure correctness of their syntax before uploading by means of online testing json files. One of the most common mistakes is uploading a file with incorrect syntax.

Encoding — all files must be in UTF-8 encoding without BOM

Even at first uploading of widget, it is necessary to change the code and key from the provided example to those unique that you’ve generated in the manifest

Oftentimes, there’s “widget” folder in the root of packed archive as the first level, but in fact, there must be files already

If you’ve downloaded incorrect manifest initially, then you will have to generate a new code and key, because the previous key will be discredited

7. script.js development

Let’s look at the overall structure of script.js:

This part of widget consists of the necessary basic parts, which we are going to study. Also, script.js may contain additional functions. Let’s look at the initial body of this file.

The whole widget is represented as an object. When the system starts downloading widgets, it expands the existing system object “Widget“ with functionality describer in script.js. Therefore, object CustomWidget inherits the properties and methods that will be useful for our work; we will look into them later. The object has callback functions, which are called upon certain conditions. These functions are listed in the table below the script.js example.

Overall layout of script.js

Callback functions, “callbacks” object

| FUNCTION NAME | DESCRIPTION |
| --- | --- |
| render: | During widget construction, callbacks.render is the first to be called. Usually, this method describes actions for widget displaying. Widget will be displayed independently only in “settings” menu. In order to display widget in other areas, e.g. in the right column, you will have to use special methods in this function, such as methods of render() object and/or render\_template() object, which are provided below. It is necessary that callbacks.render returns “true”. It is important, because without it, callbacks.init and callbacks.bind\_actions methods will not be activated. |
| init: | This function is initiated immediately after callbacks.render, simultaneously with callbacks.bind\_actions. init() method is usually applied to collect the necessary information and for other actions, e.g. communication with an external server and API authorization if the widget is used for transmitting or requesting information from/to the external server. In the simplest case, it may define the current location, where the user is. callbacks.init must return true in order to continue. |
| bind\_actions: | callbacks.bind\_actions method is used for assigning events to actions taken by the user, e.g. pressing a button. callbacks.bind\_actions must return true. |
| settings: | callbacks.settings method is called upon click on the widget’s icon in the settings area. It may be used for adding a modal window to the page (it is shown in more details below). |
| onSave: | callbacks.onSave is called upon user’s click on the “Save” button in widget settings. It can be used for sending data that were entered in the form, or changing widget status. |
| leads:selected | This function is called in case of selecting lead list elements with the use of checkbox and further click on the widget’s name in additional menu. It is used, when we need to perform certain actions with the selected objects. Examples are described below. |
| contacts:selected | This function is called in case of selecting contact list elements with the use of checkbox and further click on the widget’s name in additional menu. It is used, when we need to perform certain actions with the selected objects. Examples are described below. |
| destroy: | This function is called upon widget’s deactivation through its settings menu. For example, you need to delete all widget elements from DOM if it was deactivated, or perform some other actions. |

Example of the widget’s JS-code:

The example shown below demonstrates use cases of callback object with additional functions, and also the use of some functions of widget object. All these functions are shown in the examples below. We recommend just to look through this code, and to see the description of widget object functions for details.

This widget will select the marked contacts from the contact list and transmit phone numbers and e-mail addresses to the external server.

Functions used in this example are described in more details further. First of all, we need to turn our attention to callbacks object.

Methods of widget object

render() method

render() method is used for working with templates of twig.js template engine, which is easy to use. You can see its documentation [here](https://github.com/justjohn/twig.js/wiki).

This method is a wrapping method for twig.js. It accepts the template information (data) and data for the template rendering (params) as parameters. render(data, params). The method returns a rendered template. result = twig(data).render(params).

Let’s look at one simple example:

The result will be the following markup:

It is possible to transmit one of our system’s templates to this function. For this purpose, it is needed to specify a link to the template “ref: '/tmpl/controls/#TEMPLATE\_NAME#.twig” in the transmitted “data” object. For example, we can use the existing template to create a dropdown list:

To look at positioning in “data”, we need to add “data” into DOM. The markup of the dropdown list is styled after our system.

The full list of templates is given below. In order to use other system templates, it is necessary to modify “ref” parameter. General form: ref: '/tmpl/controls/#TEMPLATE\_NAME#.twig'

render() method accepts not only links to the existing system templates, but to our own templates as well. In order to do so, we need to transmit “data” object with a set of parameters. It is necessary to create “templates” folder in the root folder of our widget, and place template.twig in it. Let’s take a look at the example:

If the template exists at the link’s address, then the transmitted “callback” function is called, and a template object (containing “render” method) is transmitted to it. We assign parameters for rendering to “render” object. In this example, calling of callback function will take place if the template exists in the folder.

Example of function for template loading from “templates” folder

Let’s create a function for convenience in calling. We will transmit parameters into it: “template” – name of template that is located in the widget’s “template” folder, “params” – template parameters object, “callbacks” – callback function, which will be called after template loading (in this case we will add the template to a modal window). More information about modal window object is available in section [JS methods and objects for working with amoCRM](https://developers.amocrm.ru/widgets/js_sdk.php).

render\_template() method

render\_template() method wraps the transmitted markup or template in standard shell for widgets (markup) and places the resulting markup into the right column of widgets

It is possible to transmit html-markup to this function, or a template with rendering data, as in the case with render() method.

The function complements the transmitted markup with its own markup that is stored in “template\_element” variable of “widget” object.

We have shown the simplest example of using the template, but render\_template() method can also accept templates and template data as parameters. Also, a link to the template can be transmitted to it as with render() method.

Now, we get a widget in the right column that was created from the template.

“widget” method has another set of useful functions, which can be used to solve various tasks.

Description and examples are given below.

set\_lang() function

set\_lang() function allows changing parameters, which are set by default from the files of i18n folder.

The current object “lang” is stored in “langs” variable of “widget” object

set\_settings() function

set\_settings() function allows adding properties to the widget.

widgetsOverlay() function

widgetsOverlay() function (true/false) enables and disables an overlay, which appears from the list of contacts or leads, when widget is called.

add\_action() function

During the user’s work, it is possible to provide call of some function in the contact/lead list area by clicking on a contact’s phone number or e-mail address.

add\_action() function uses (type,action) parameters, where “type” has ”e-mail” or “phone” value, and “action” is a function that will be called by clicking on a phone number or an e-mail address.

set\_status() function

Widgets may have one of three statuses. Widget’s status is displayed in “settings” area on the widget’s icon. In case your widget uses data entered by a user for an API of an external service, and these data were entered incorrectly, then it is possible to use this function for displaying “error” status.

The available statuses are “install” (widget is not active) and “installed” (widget is active), error (widget is in error state).

crm\_post(url, data, callback, type, error) method

This method is used for request sending to your external server through amoCRM proxy. It is necessary to use this method, because users work with amoCRM through a secure SSL protocol and the browser may block cross-domain requests. The best solution are availability of a signed SSL certificate on the side of internal system and working via HTTPS. The function is similar to jQuery post() function, but it has an ability to capture errors (5th argument “error”), see documentation (<http://docs.jquery.com/Post>)

Method description

self.get\_settings method

This method is necessary in order to obtain data that the user has entered during widget connection. The data are returned in the form of “javascript” object

This method allows obtaining an object from language files containing messages in language locales used by the user.   
The name of object to be derived is transmitted in “objname”

For example, let’s call self.i18n('userLang') function

Therefore, having this simple tool for interaction with DOM and performance of cross-domain requests allows you (besides creating simple text widgets) to easily change the design of page elements, create your own information units based on external data, or vice versa, forward data to external services, at that, all of this will immediately work for all users of your account.

Widget uploading

8. Archive packing and uploading

You will need to pack only “widget” folder into a zip archive, which must necessarily have “widget.zip” name.

Now, we must enter our developer’s page in amoCRM, where we have received our widget code, and upload our archive.

In case the uploading is successful, you will receive a corresponding message. Widget’s version and the number of enabled widgets are shown in the same line as the widget code.

Initial widget status is “private”. It means that the widget will be visible only to the users of your account.

Widget may have one of 4 statuses:

“private”: widget is visible only to its creator. It is the general status during widget’s development.

“moderate”: widget obtains this status only in case it is necessary to publish the widget for all amoCRM users

“prepublic”: widget endpoints are enabled at this status. Widgets obtain this status for final testing before publication, and your widget will not remain in this status for a long time. (our employees give this status)

“public”: widget having this status is available to all amoCRM accounts for installation and use. Widget obtains this status after successful moderation. (our employees give this status). In order to roll out an update for the already published widget, you must upload an archive with different code, and indicate (during moderation) that it is necessary to replace the already published widget.

Immediately after the widget is uploaded, you will see it on the page of your account’s widget list, just like other users will see it, when you publish this widget.

9. Widget testing

For widget testing, you may use a simple php-script.

10. Widget moderation

Only those widgets that must be published to all amoCRM users will undergo the moderation. These are the general issues considered during moderation:

Ease of use. A non-technical user must understand how to turn the widget on, how to configure it, how to work with it, and where to get support. The clearer the widget operation is, the more users will use it.

Development quality. There must be no errors known by developers. If an emergency arises, the widget must give messages to the user about failure with contact details

Security.

How to submit a widget for moderation:

Assign “moderate” status to widget on the developer’s page

Write a message to support@amocrm.ru indicating:

The code of widget to be published

The description of widget’s work

If needed, the test data for connection to an external system, because our employees are going to check the widget’s operability from the user point of view

In case of problems in widget’s operation, a phone number or e-mail must be available, where an end-user or a member of our technical support can get consultation, help or error correction

At the moment of passing to moderation, widget must be ready for publication, i.e. there must be no debugs or testing data in the code, widget’s description must be prepared, and languages for publication must be defined.

PHP-library

Basics

First of all, you need to download a library for interaction with amoCRM. It is located in php-widget example that can be downloaded in the bottom of this page. The library must be placed into the root of your web-server, and the server must have a second-level domain. After that, you can create “index” file, which will contain basic settings and connection of this library. Also, it is necessary for the widget to have “public” status, since the endpoints, which will be created further, are available only to public widgets.

Now, we have to create /widget/ folder, where the widget development will take place. There, we shall create the whole widget structure and additional files.

/widget/widget.php file

/widget/widget.php file must contain “Widget” class, which inherits \Helpers\Widgets system class.

“Widget” class must contain methods playing a role of access points to the widget. These methods must have more strict access level than “public” (i.e. “protected” and/or “private”). Access point must have a name, which begins with endpoint\_ prefix, for example: endpoint\_get().

Access point call is made via URL: /#ACCOUNT#/#CONTROLLER#[/#METHOD#[/#ENDPOINT#]], where #ACCOUNT# is your account in the system, #CONTROLLER# must have “loader” value for calling widget access point, #METHOD# is the widget code (code must contain only lower-case letters!), #ENDPOINT# is an access point itself.   
For example: /test/loader/addcontact/get/

CONTROLLER may also have “builder” value. In this case, it is not needed to specify METHOD and ENDPOINT. Call of “builder” controller builds our widget for working with “loader” controller, and creates a zip-archive for uploading to amoCRM.

At the first call of “loader” controller and after each modification of widget files, it is necessary to call “builder” controller; or you can set AUTO\_BUILD as “true” (described above)

Also, it is important to remember that “amouser” and “amohash” must be transmitted during access point call in order to work with amoCRM server. These fields can be seen in user’s profile settings in the system (/private/account/settings/profile.php). “amohash” field is the user’s e-mail, “amohash” is the API authorization key.

The library for working with widgets enables direct interaction with the system by means of access points (see above). For example, let’s create a widget that adds contacts to amoCRM.

Peculiarity and limitations of the library

All requests must be forwarded to your index.php file in the root directory. If you have Apache web-server, or you have unpacked an archive with the library to its root, then it already has .htaccess file with instructions for mod\_rewrite (make sure it is enabled)

By default, links to objects are already present in your widget’s object   
- contacts: $this->contacts  
- companies: $this->companies  
- leads: $this->leads  
- notes: $this->notes  
- tasks: $this->tasks  
- account information: $this->account->current()  
First five objects have two methods – “get” and “set”. Their parameters are described in the example file widget.php. “contacts” and “companies” objects have one more method - links(). This method corresponds to the similar method in API.

For sending cURL-requests to an external service, you can use a built-in class  
\Helpers\Curl::init($url,[$post=FALSE],[$cookie=FALSE]);   
where:   
$url is a link to where requests will be sent,   
$post is an array to be transmitted (if it is full, the request will be sent via POST method),   
$cookie (TRUE/FALSE) indicates whether to use cookie-files or not

Any parameters coming from GET or POST must be obtained through   
\Helpers\Route::param(#ELEMENT\_KEY#)

\Helpers\Route::param(#ELEMENT\_KEY#) method is available as much as $this->param(#ELEMENT\_KEY#)

Obtaining settings of the current widget in amoCRM may be performed by means of calling   
$this->account->current('widget');

The following built-in class may be used for working with language messages  
\Helpers\I18n::get('settings.enums.yes')  
All language messages must be described in /widget/i18n/#lang#.json directory

Creating your own web-page

In the example below, we will demonstrate creating a simple html-form for contact adding.

Let’s create a file with an html-form and call it form.php

Manifest creation

Widget manifest is a file containing widget description and settings in JSON format. It is recommended to carry name, description and other static information over to the widget’s localization files (see below).

Creating localization files

Localization file is a file in JSON format, containing transfer of static information used during widget’s development. These files are edited during widget’s logic development depending on necessity to enter a certain type of new information.

Let’s create two localization files for our example: one in English and one in Russian.

Widget programming

Let’s create an empty class “Widget”, which inherits \Helpers\Widgets system class, and then make an access point in it, which will be named “add”

Let’s create get\_data() inner method (marked with “private” modifier), which will receive data from the form and write them to the inner property $data. After that, we will call it within the access point.

Now, we’ll create get\_custom\_fields\_info() inner method to obtain information about certain fields in amoCRM. After that, we must save its result in $custom\_fields variable within the access point.

Now, we have to find out if the user has a contact with the specified e-mail. For this purpose, we will create is\_contact\_exists() inner method and perform the necessary check within the access point.

Finally, we can create a contact in amoCRM. For this purpose, we will write add\_new\_contact ($custom\_fields) inner method, which accepts an array with information we gathered in get\_custom\_fields\_info() as a parameter. Now, we must call it in the access point.

Now, PHP-logic of our widget is ready!

Debugging

We should notice that \Helpers\Debug::vars($var\_for\_debug[, $name\_of\_debug\_block]) can be used for debugging on local hosts. This method outputs a made-up page with debugging information. As the first parameter, we must transmit a variable that needs to be debugged. As the second (optional) parameter, we can transmit the name of debugging block. When uploading the widget, don’t forget to delete debug outputs.

During development of your widget for integration with amoCRM, you may meet with the numerical codes of states or errors, which our API returns along with the response. In order to understand what one or another code means, you may use our  [API reference guide](https://developers.amocrm.ru/rest_api/response_digest.php) or use \Helpers\Curl::get\_error\_code($code) method, which returns an error message by its numerical code.

Packing and uploading

If during development you have specified AUTO\_BUILD constant as “true”, then the folder, where you have created your widget, will contain the automatically created /widgets/code/ structure (if there is no such folder, you have to call “builder” controller manually), where “code” is the widget code. It also contains widget.zip archive, which must be uploaded to amoCRM in /private/account/settings/development.php section

Download the PHP-library for widget development

You can always download the latest version of php-library by using the link below:

The example provided on this page is available here:

Advanced widget settings

amoCRM widgets support adding a proprietary program logic (an unstructured field and appearance) to the widget settings page

An unstructured field consists of “hidden input” (a field, through which reading and saving are performed), div-element, where we can output DOM-elements for interaction with a user, or some javascript code that provides the necessary logic.

In order to use an unstructured field, we need to make two easy steps:

Add a field to manifest.json and permit the widget to work on the settings page

Implement data reading and writing

Describing an unstructured field in mainfest.json

It is a standard field with special type “custom”. Perhaps no more than one such field is needed in one widget. Don’t forget to add “settings” location to “locations” array in mainfest.json!

Now, build the widget and upload it to your account. You will obtain “div” with an ID <widget code >\_custom\_content and “hidden input” with ID <widget code >\_custom.

Implementation of data reading and writing

For this purpose, you can simply implement “callback settings” in this.callbacks. See the example below.

In this example, we have implemented the possibility to write a random number from 0 to 1 in the unstructured field.

JS methods and objects for working with amoCRM

In this section, we have described functions and objects, which serve for the simpler calling of environment (information about widget, authorized user, etc.), and also for calling certain interface elements.

Pop-up notification

The system adopts ability to output a notification window in the lower right corner. As an example, we can take notification called by the telephony about the incoming call.

In order to implement this function, the provided object may be used. In the example, there is a function created to work with such object.

This example not only calls a notification window, but also makes a link.

Error notification

We recommend using such notifications if JS performs some background actions (called hidden from the user, not by his call). In this cases, you may notify the user that something went wrong, and what should the user do.

For example, you’re developing a telephony widget. You connect to server in background, and await events about incoming calls. At a certain point, you find out that you can’t perform a connection with your server, and the expected functionality won’t work. It is better to notify the user about this fact with an error message (describing the cause of error) containing the phone number of technical support.

The amount of inquiries received by the technical support about problems that the user can’t understand depends heavily on how thoroughly such messages are processed. If you will output a message to the user telling that he has entered an incorrect password, then the probability that you will receive a call about error is less.

This object is very similar to the above-described notification, but it outputs an error message. At that, the function for window closing behaves differently; it remembers closing event in the user’s COOKIE and doesn’t show it anymore.

Let’s analyze the parameters in more details:

header: (string) a widget name that will be shown in the header

text: (string) an error message

date: date

callbacks: callback functions object. AJAX request is sent to the server when adding a new message or error. It returns the number of such message in case of successful data saving. Depending on the request’s success, one of the transmitted object functions is activated.

“Modal” object for working with modal window.

In order to work with this object, it must be added to script.js file

In this example, we have shown the use of modal window object “Modal”

The example is provided below.

In order to work with modal window object, we need to enable it with “require” (“define” in the beginning of script.js) and transmit parameters: class\_name, init() , destroy(). The data to be displayed in the modal window and “trigger” events are transmitted to “init” in order to enable methods of “Modal” object” and output the modal window to DOM.

Examples. Form for adding

For your special benefit, we have prepared training lessons on working with amoCRM mechanisms. These examples will help you to introduce yourself to the system in the shortest time possible and to use it to the maximum.

We have already [described](https://developers.amocrm.ru/widgets/php_lib.php) the process of developing a widget that adds a contact to amoCRM using a standard html-form. Now, we need to do the same using [REST API](https://developers.amocrm.ru/rest_api/)

Creating your own web-page

The example below demonstrates creation of a simple html-form for contact adding.

Let’s create a file with html-form and call it index.php

We will add styles to the form, so it would look or less decent. Let’s create main.css file containing the following:

And enable styles in html-file described above. In order to do so, we must insert the following string into <head> tag, right after </title> tag:

Programming a logic

For illustration purpose, we will derive separate logical blocks into different files and connect files sequentially. You can develop your own architecture for your application interacting with our API.

Controller

Let’s create handler.php file. It will connect a number of files, each performing a certain task.

Preparation

We will prepare.php file, where preparative actions will be performed: declaring of functions that we will need further and parsing of incoming data.

Authorization procedure

Authorization procedure is described in details in [documentation](https://developers.amocrm.ru/rest_api/auth.php). All you have to do is create auth.php file and copy the code there after replacing authorization data your credentials. The result will be something like this:

Obtaining information on the current account

The procedure of obtaining information on the current account is described in details in [documentation](https://developers.amocrm.ru/rest_api/accounts_current.php). All you have to do is create account\_current.php file and copy the code there. The result will be something like this:

Obtaining a filtered contact list

The procedure of obtaining a filtered contact list is described in details in [documenta](https://developers.amocrm.ru/rest_api/contacts_list.php)tion. All you have to do is create contacts\_list.php file and copy the code there. The result will be something like this:

Obtaining field information

Let’s create fields\_info.php file and process the account information (obtained above) in order to obtain field information.

Adding a contact

The procedure of adding a contact/contacts is described in details in [documentation](https://developers.amocrm.ru/rest_api/contacts_set.php). Let’s create contact\_add.php file. It will perform forming of the array with collected data as well as the very process of adding contact to amoCRM.

Integration with virtual ATS

Integration of amoCRM and virtual ATS (telephony) is about the following: data exchange takes place upon certain events and during a certain period of time. There are several cases for this, and we will look into each of them in more details further.

Click To Call (C2C)

This functionality allows outputting an icon of phone handset by the side of each phone number in lists or in a card. This icon, when clicked, sends a request to the virtual ATS for call commutation. When the request is received, the ATS calls the internal number of an employee (indicated in widget settings) at the first place, and then the phone number, which needs to be call (or vice versa in some ATS).

For the purpose of simpler implementation of this function in your widget, you may use a ready-made JS-object [C2C](https://developers.amocrm.ru/widgets/js_sdk.php).

This function is aimed to make life easier to the users that make outbound calls. It works fine with both landline IP-telephones and SoftPhones.

Call recognition

Immediately at the moment of receiving a call to the employees’ phone, virtual ATS has ability to require information about the caller through amoCRM API and transmit it to the employees phone. The employee’s phone is ringing, and at that, name (company) of the contact found in amoCRM is displayed on the phone screen. The manager will know, whom he is speaking to, after picking up the phone.

In order to perform a search, we need to use [contacts/list](https://developers.amocrm.ru/rest_api/contacts_list.php) method by transmitting the phone number to “query” field. At that, this method (just like all API methods) is called from an authorized user. It means that user privileges for accessing contacts are considered. I.e. in order to recognize the phone number, the contact must be in amoCRM database, and the corresponding user must have privileges for accessing the needed contact card.

It is necessary to set minimal “timeout” method for request processing. Otherwise, there may be problems with calls in case of degradation of communication between ATS and amoCRM.

It is worth noting that it is necessary to output the phone number of the caller in the call recognition display when using the above-described functionality of C2C.

Smart diversion

If the above described function of call recognition is used during an incoming call, we can also obtain ID (e-mail) of amoCRM user, which is responsible for the contact card of the caller. We can obtain the internal phone of the responsible employee from settings, and divert the call exactly to this employee.

Incoming card

During an incoming call from a contact to a user of amoCRM, we can output a notification about the incoming call in the right lower corner of the interface. In case this contact is already in amoCRM database, then we will output its Name and Company by giving a link to the card. In case there is no such contact in CRM yet, then the notification appears indicating the caller’s phone number and offering to create a contact.

For convenience in outputting notification, this [JS-](https://developers.amocrm.ru/rest_api/contacts_list.php)object is provided.

The most difficult thing is to deliver information about the incoming call to the JS-script on the client’s side. This is usually performed by means of WebSocket technology (for example, in onlinePBX or Oktell telephonies), when a permanent connection and event subscription are set up between client and server. We may use the technology of periodical calls to an external server via JS (this is how Asterisk widget is working). For this purpose, a JS-file from the targeted server is loaded on the client’s side every few seconds, where an array with channel status is defined (there is or there is no call to a certain employee’s internal phone).

The selection of method depends on technical capability on the side of virtual ATS to support a WebSocket connection. At that, it is necessary to consider the internal phone numbers of employees and their correspondence with the users authorized in amoCRM and browsing the interface.

Call logging

A user must have ability to view all of his calls made to a certain contact. Calls are logged to events of the corresponding contact, to the corresponding types of events for outbound or inbound calls. If the ATS supports call recording, then the user may obtain a link and a media player to listen to the recorded conversation.

In order to add records to the events, we need to use [calls/add](https://developers.amocrm.ru/rest_api/calls_set.php) method. At that, the events must be added only to the already existing contact found by the phone number. This is the reason why we mark call records in the database as sent or unsent (for which a corresponding card was not created yet), and try to add the unsent call records again within 18 hours.

The example of calling method for event adding:

Description of the general parameters can be seen in the corresponding method.

Error processing

During appropriate telephony operation, a continuous data exchange takes place in the background of each user in order to define the status of his line. The problem is that background processes (not called by a user) provide no place, where we could output an error. Oftentimes, a user can’t figure out why his telephony is not working even if he has entered a wrong login/password pair to the ATS.

Exactly for the purpose of notifying a user about problems that take place in the background processes, it is necessary to use a separate JS-object, which will output an error notification to the user during the call (e.g. about failure of connection to server). For more information, read documentation on [error notifications](https://developers.amocrm.ru/widgets/js_sdk.php).

Asterisk testing script

If you as a developer have a task to simply get acquainted with the functionality of a telephony system, or you have to connect an external ATS exactly for your account, then you may download a special stub-script in PHP imitating the work of Asterisk.

[Download](https://developers.amocrm.ru/download/astgw.zip)

After downloading, you will have to connect Asterisk widget in your account, and specify an address where you have placed a testing script. Then, you can replace the stubs of each PHP-method with the actual data obtaining from your ATS, and the integration will work.

API console

In order to test any method (except for authorization), you need to click on the dropdown menu, which is at the top of “Authentification”. Then, you need to select “Custom token”. In the next menu, you need to input data in the following form: USER\_LOGIN and USER\_HASH. It is important to notice that this step is not needed, if you’re testing the authorization method.

In order to select a method for testing, you need to click on the slider on the left, fill all mandatory fields and click “send”. The response with all headers will appear in the “Response” window.

In order to send POST request, select a method that contains POST. Then, see the manual for data that we need to send. After that, go to BODY tab and input JSON array (which we have formed using the manual) in the text field. After that, click “send”.