Management API

In this article

- Access the API
- Authentication
- API Request
- API Response
- Work with files
- Examples: Pure JSON-RPC
- Examples: Python Library
- Methods Reference

The **Management API** allows you to easily integrate 3rd party applications with the JeraSoft Billing platform. It may be accessed using JSON-RPC - a standard protocol for remote procedure calls.

Most programming languages have libraries to work with this protocol. The API uses the same logic as the web interface and works very similar, which makes its usage easier.

Access the API

To access the **JSON-RPC** interface, use:

```
https://<your-system-IP>/jsonrpc/
```

Before accessing the system, make sure that your **IP is allowed** on the firewall. Also, please make sure that the rest of the world is blocked by the firewall. You can get more details about it in the JeraSoft Billing First Steps.

Authentication

To make any request to the API, you need to **authenticate** using login and password.

We strongly recommend having a **separate API account** for each application you make calls from. Besides this, we recommend having a dedicated Role for API users, which allows only needed actions. Authentication is done by sending the following array along with other arguments:

```
{ "auth": { "login": "admin", "password": "password" } }
```

To increase performance, you may not authenticate for each call but do it once and **save Session ID** given back to you in response.
This Session ID may be sent in further requests instead of login and passwords to continue using the same session without re-authentication. Session ID should be sent with other arguments in the following format:

```
{ "SID": "1-dsglnqr4qnsdihr8djj6da7qr4" }
```

It should be mentioned that authentication should be made **within** the first call to the API.

API Request

Each request besides authentication information should include:

Name	Description	Example
Method Name	Name of the module and action to call	`clients.editForm`

Arguments	List of arguments for the method call	`id_clients = 11`
		`auth[login] = admin`
		`auth[password] = password`

API Response

Each response will include the following information:

Name	Description	Example
code	Return code, usually true on success or false on failure	`1`

return	Array with data returned by the method	`[client] => Array`
		`(`
		`[id] => 11`
		`[id_companies] => 3`
		`[type] => 0`
		`[name] => Customer A`
		`[groups] => Customers`
		`[c_dt] => 2013-03-30 16:26:15+03`
		`[status] => active`
		`[credit] => 100`
		`...`
		`)`

session_id	Session ID, which may be used to speed up next calls	`1-dsglnqr4qnsdihr8djj6da7qr4`
messages	List of success/warning messages returned by message	`array()`
errors	List of abnormal errors if they fired during processing	`array()`

Work with files

There is a specific case when your request to the billing should provide file response. For example, it could be an invoice file download, xDRs List download, etc. Using plain JSON-RPC with base64 on top of it is not effective here as the file may contain hundreds of megabytes. This special case is handled with common **HTTP Request** to:

```
https://<your-system-IP>/admin/
```

The request may be either GET or POST and should include either Login and Password or Session ID. In response, the server will send the file according to HTTP protocol.

Examples: Pure JSON-RPC

JSON-RPC Request

```
{
  "method": "clients.editForm",
  "params": {
    "id_clients": 11,
    "SID": "1-bmdgeu6bn22jlmkuffg391t513"
  },
  "id": 1
}
```

JSON-RPC Response

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
    "code": true,
    "session_id": "1-bmdgeu6bn22jlmkuffg391t513",
    "messages": [],
    "return": {
      "client": {
        "id": 11,
        "name": "Customer A",
        "groups": "Customers",
        "c_dt": "2013-03-30 16:26:15+03",
        "status": "active",
        "credit": 100,
        "c_company": "Mancy",
        "c_address": null,
        "c_email": "admin@example.net",
        "c_email_tech": "admin@example.net",
        "c_email_billing": "admin@example.net",
        "c_email_rates": "admin@example.net",
        "id_currencies": 27
      },
    },
    "errors": []
  }
}
```

Examples: Python Library

To download a sample library for Python, please visit **/opt/jerasoft/vcs/core/pycore/tools/vcsapi.py.** It will simplify work with API. You can find an example below:

Python Example

```
import vcsapi

# create an API object
api = vcsapi.Api('vcs-demo.jerasoft.net', 'demo', 'demo', 443)

# make a call to the API
response = api.clients.editForm({'id_clients': 8})

print response.code      # True
print response.messages  # [ ]
print response.data      # {'client': {'id': 8, 'type': 0, 'name': 'Customer 01', ...}}
```

Methods Reference

At the moment we are working hard to bring you a full and detailed list of methods, arguments and expected output. However, as API fully duplicates web methods, it is easy to find their names and arguments yourself. Let's check a quick example, like creating a reseller.

In the web interface, the link to this action is ***https://<your-billing-IP>/admin/companies/add***, with ***companies*** being a module and ***add*** being a method. The resulting method to call via API is ***companies.add***.

To find out arguments for this method, you may look for HTTP request in your browser (using FireBug in Firefox, Developer Tools in Chrome). Another way is to check for the dump in **/opt/jerasoft/vcs-data/log/runtime.log,** which looks like:

```
[20-Jan-2012 17:32:06+0200] [webAdmin/#260808] REQUEST: /companies/add
Array
(
        [type] => 10
        [name] => TESTCOMPANY
        [id_companies] =>
        [prepaid] => 1
        [credit] => 0.00
        ...
)
```

This log entry includes the full list of the arguments used. However, many of them are optional. Try calling the method with the arguments you need, and the system will let you know if you are missing any of the arguments.

Warning

The JeraSoft team may modify the attributes and methods related to the Management API usage from time to time without advance notice.