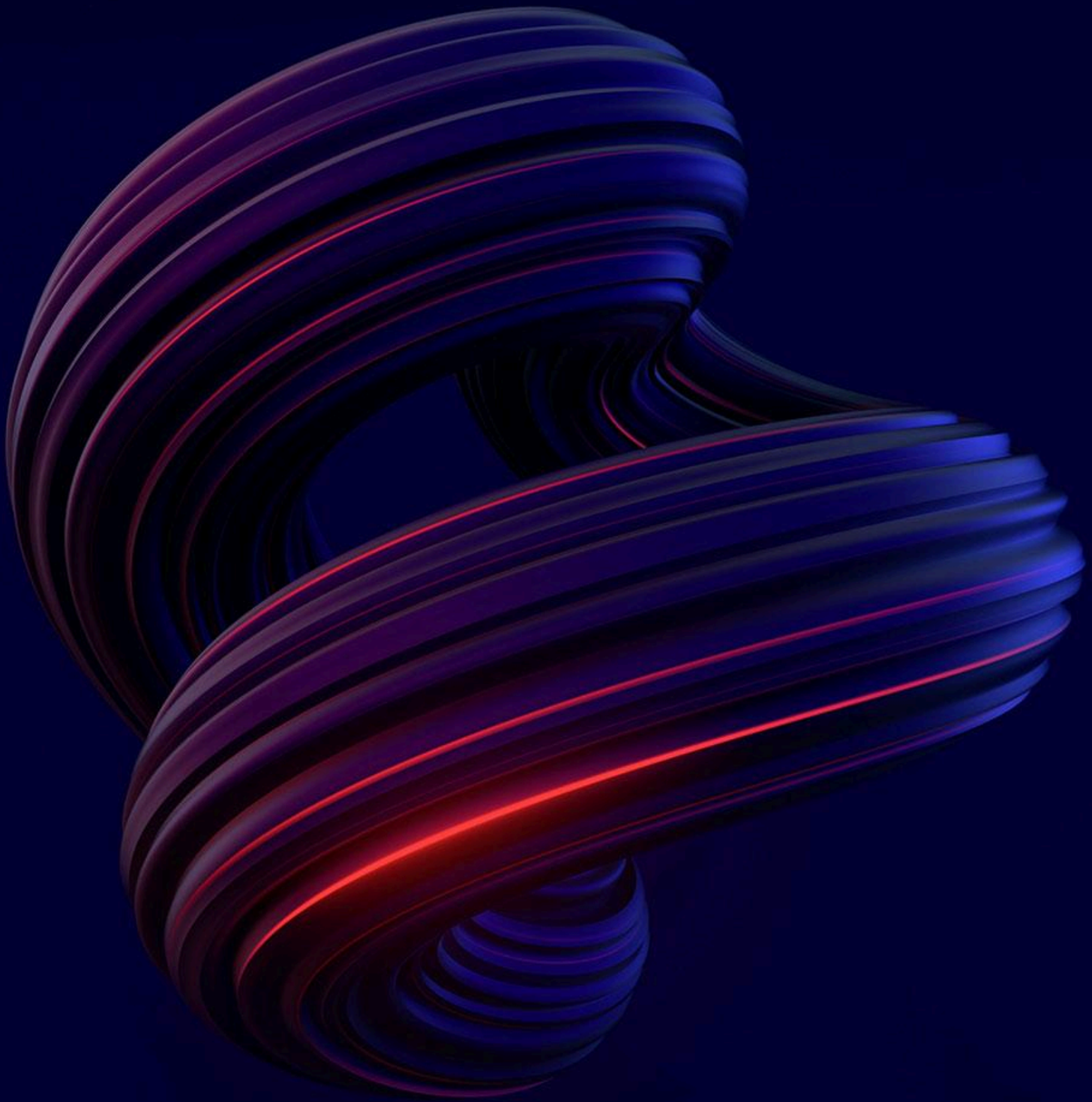
MiniCRM



INTEGRATIONS MANUAL

Last update: December, 2025

BASICS **6**
SystemID 6
How to find the SystemID? 6
Product (module/category) 7
Opportunity card (or project) 8
Primary contact and other contacts 9
Structure of the cards in special products (invoice, order, offer etc.) 10
Project Management specifics 11
Company and contact (or account) 12
Company card 13
Contact card 14
Field name 15
API key generation 16
REST API key generation 16
VOIP API key generation 18
Register your company as a VoIP services provider in MiniCRM 19
VOIP integration specifics 19
Production and test environments 20
How to request a test environment? 20
INTEGRATION OPTIONS **21**
REST API vs. XML synchronization 22
Invoice API 23
Call log endpoint (VOIP integration) 24
Webhooks 25
Webhook filter criteria 26
Type 26
Product 26
Fields 26
Webhook examples 27
How to set webhooks 28
Built-in integrations 29
MiniSync Integration 29
Benefits of MiniSync 29
When to Choose MiniSync 30
Use cases for MiniSync 30
Integration with calendars 31
Google calendar 31
Other calendars 31
Gmail 31
Outlook 32
Facebook ads forms 32
Gravity Forms Integration 32
WooCommerce 33
Shoprenter 33
UNAS 33
OpenApi 34
Cégjelző 34

NAV (Bejövő számlák) 35
BestJobs / eJobs integration 36
SmartBill integration 36
REQUIREMENTS 37
Pagination 37
cURL usage 38
Postman 40
HTTPie 41
Error messages and error handling 42
Error codes and their meaning 42
API limits 43
Encoding 43
Request format 43
Results pagination 44
Other specific limitations 45
Item-based products 45
To-do endpoint 45
Templates 45
Contacts 45
API ENDPOINTS 46
Company / contact endpoint 46
Create a new company / contact card 47
Create a new company 47
Create a new contact 47
Search for companies / contacts 48
Search based on name 48
Search based on the email address 50
Search based on the phone number 51
Search based on the update date 52
Search based on a specific keyword 53
Search based on a specific field 54
Retrieve all the contacts from a company 56
Detailed data of a contact / company 57
Update data for companies / contacts 59
Delete a contact 60
Address endpoint 61
Create a new address 61
Search for address 62
Search based on contact/company ID 62
Search based on address ID 63
Card endpoint 65
Schema of the card 65
Create a new card 67
Create a new a company 67
Create a new contact person 68
Search for cards 69
Detailed data of a card 69
A company's cards from one module 70

Search based on card status 71
Search based on the update time 72
General time and date format 72
Search based on fields 75
Search for cards Extra 75
Search the card status history 76
API request to get the contents of a filter 78
All cards of a company 81
Update a card 82
Opportunity card owner/responsible: 83
Moving an opportunity card to another contact connected to the same business 84
Changing the main contact of the card which is connected to another business 84
File uploading to a card 85
Multiple file upload to a card 85
Multiple file upload 86
Deleting a card 87
Modifying deleted cards 89
Restoring a card 89
Delete a contact 90
Error handling 90
Card connection endpoint **91**
Create a new card connection 91
List card connection 92
Update card connection 93
Delete card connection 93
To-do endpoint **94**
Create a new to-do 94
Search for to-dos 95
Detailed data of a to-do 96
Update a to-do: 98
File upload 99
Single file upload 99
Multiple file upload 99
Custom close date and time 100
Templates endpoint **101**
Get the template list for a specific product 101
Search for templates 102
Template details 102
Order endpoint **103**
Creating a new order 103
Search for orders 105
Detailed data of an order 105
Order list endpoint 107
Update order's data 108
Adding an order item 108
Editing custom order fields 110
Editing the status of an order 111
Finalizing an order 111

Editing an order 111
Completing an order 111
Setting an order to Successful 112
Setting an order to Unsuccessful 112
Deleting an order item 113
Offer endpoint 114
Creating and issuing offers 114
Search for offers 117
Detailed data of an offer 117
Offer list endpoint 119
Updating an Offer 120
Update Offer's opportunity card fields 120
Adding a new item 120
Modifying an item 121
Deleting an item 121
Invoice endpoint 122
Create a new invoice 122
Search for invoices 125
Detailed data of an invoice 125
Invoice list endpoint 127
Update an invoice 128
Mark an invoice as paid 128
Storn an invoice 130
Editing custom fields (invoice card fields) 132
Stock endpoint 133
Additional pagination details for stock-related products 133
List filtering parameters 134
Adding a new product 135
Update a product 136
Delete a product 137
CallLog API Endpoint 138
Create a new call log entry 138
Search for phone calls 141
Listening calls in MiniCRM 141
Integrating Webforms through API with Advanced Webform Addon 142
Creating a Webform 142
Form Submission Via API 143
CONTACT 145

BASICS

SystemID

The SystemID is a basic concept in the MiniCRM API, and it is formed from 5 numbers. It is used to authenticate the API requests. The ID is stable, and it will not be changed at any point in the future.

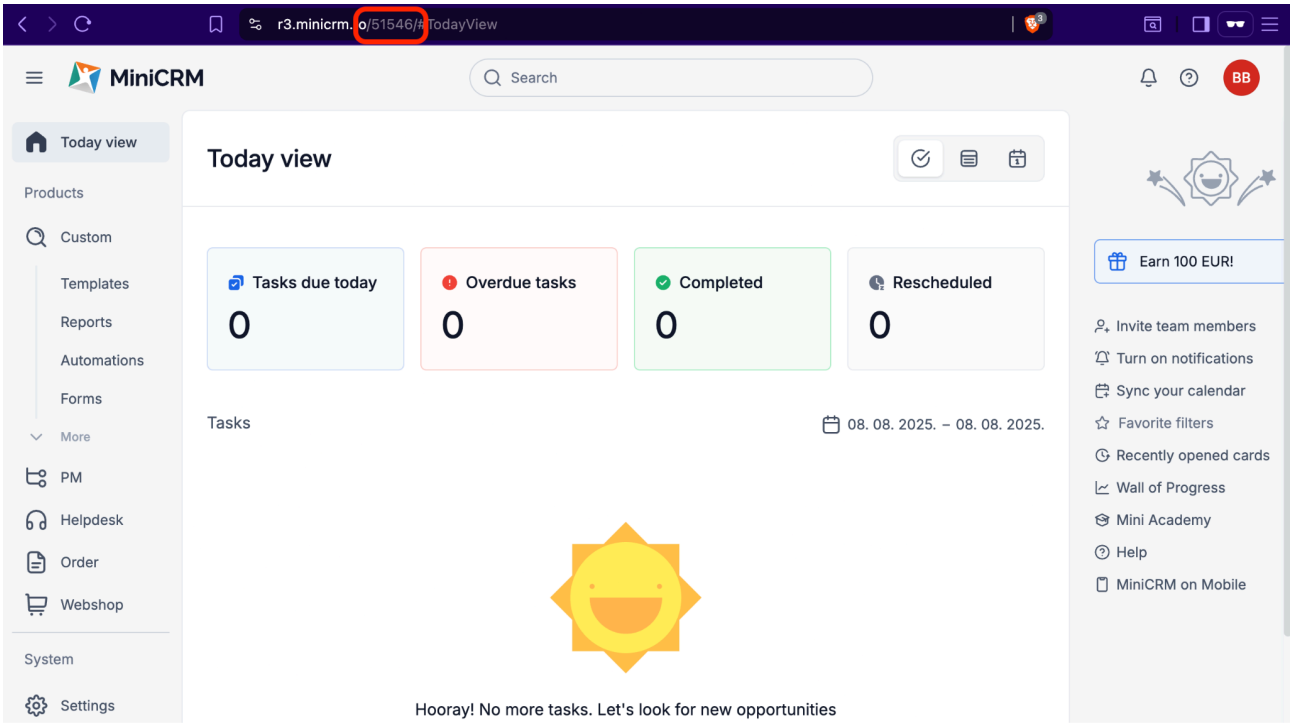
The SystemID can also be used in a cURL connection URL:

```
curl https://r3.minicrm.hu/Api...
```

Also, during this manual, we will refer to this as *SystemID*.

How to find the SystemID?

The SystemID can be found in your browser address bar right after the domain (51546 in this case):

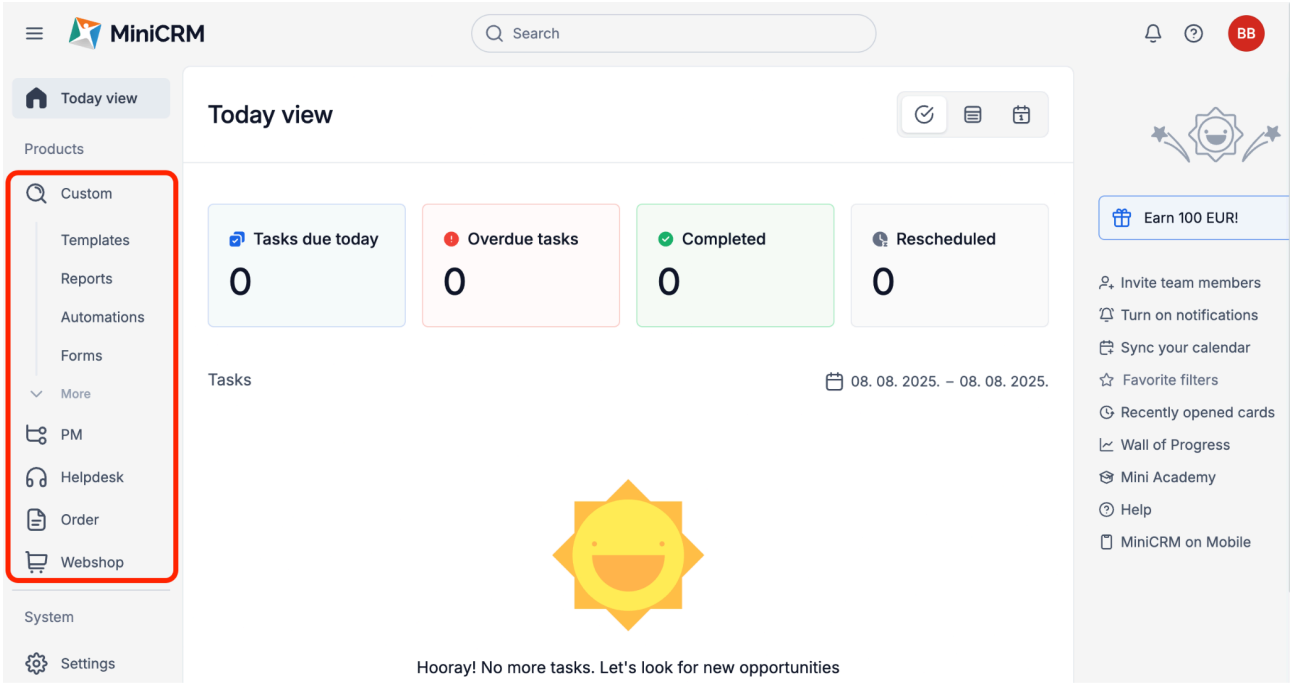


Product (module/category)

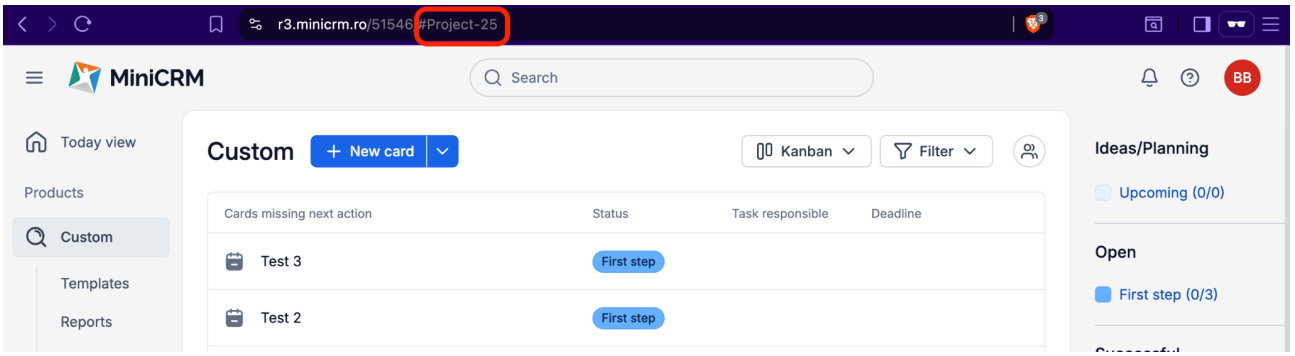
In MiniCRM, the data and the processes are organized into products. We will use, during this manual, also the term "product" or "module" or "category" – *all representing the same thing*.

It is possible to notice that in the API we will refer to a product as a *category* (and the most used term is the CategoryID).

We call the product, module, or category the menu entries from the sidebar, and some examples may be Sales, Invoice, Helpdesk, etc.:



During this manual, we will use mostly the CategoryID and this ID can be found also in the browser address bar in the page URL (25 in this case):



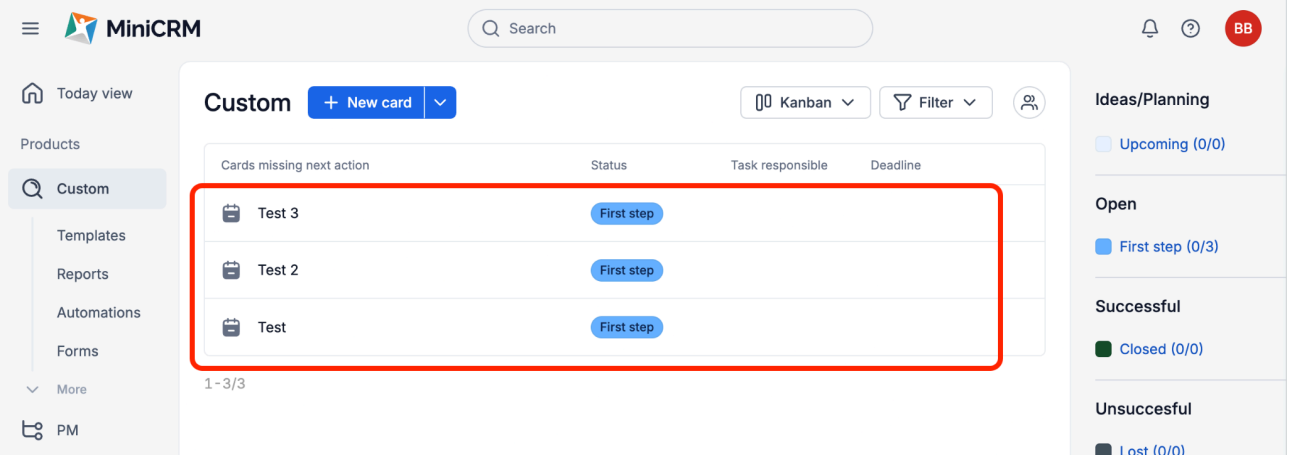
To resume, via API, the products (or modules) are available under the name of *Category* and their IDs are available under the name of *CategoryID*.

Opportunity card (or project)

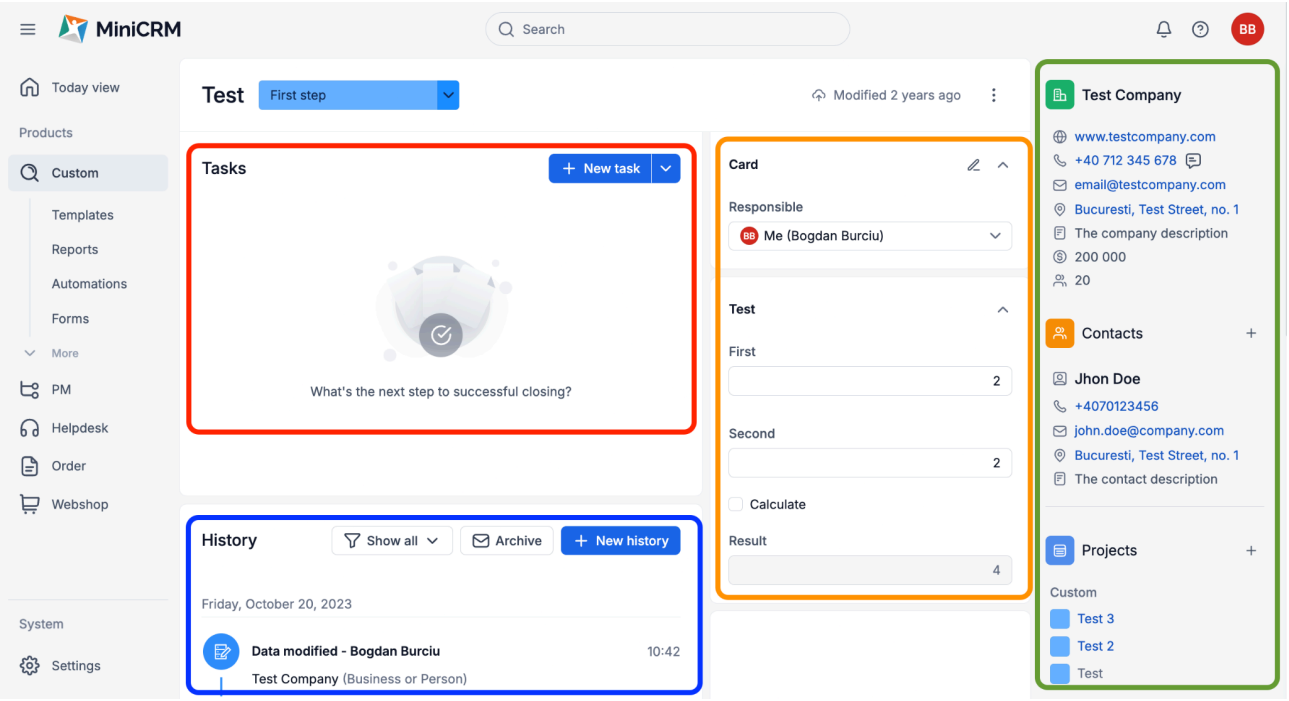
The opportunity cards or *projects* are listed within the products (or modules – please see the previous chapter for details).

The structure of the opportunity cards is the same in every product. The differences lie in the concrete fields (each product may contain other fields – these fields can be added by the administrators) and the data assigned to the cards. The card fields can be customized via the web user interface.

In the image below, each row represents a card within the Custom product:



The card structure is composed by task area, history, card fields and contacts area:



Some basics about each of these areas:

- **Task area** - here the system will display all the open tasks, each of them with details (task content, owner, deadline, comments etc.);
- **History area** - the system keeps track of each change that happens on that specific card and displays them in this area. Also, in this area will be displayed sent emails (or other messages like SMS or WhatsApp). In this section, there are displayed the closed tasks as well;
- **Fields area** - MiniCRM is a customizable system, so you will be able to create your own fields and those fields will be displayed in this area. Each change in these fields will be displayed in the history area;
- **Contacts area** - this area contains some specific areas as described below:
    - ***Business data*** - is the area where the company data will be displayed (check the chapter "Company, contact (or account)" below to understand how data is organized in MiniCRM). Please note that in MiniCRM a card can exist without business data (in this case that card will only display one contact - the primary contact);
    - ***Contacts*** - below this header will be displayed all the contacts that belong to that specific company listed above. Also, there will be only one *Primary contact,* which is the contact that the current card is related with;
    - ***Projects*** - here will be displayed all the other cards related to the company or contact. The cards will be grouped by the product they belong to (Custom in this example).

The account area may contain information related to company, contacts and other cards in other system products for the same company or contact.

Via API, the cards are available under the name of *Project*.

Primary contact and other contacts

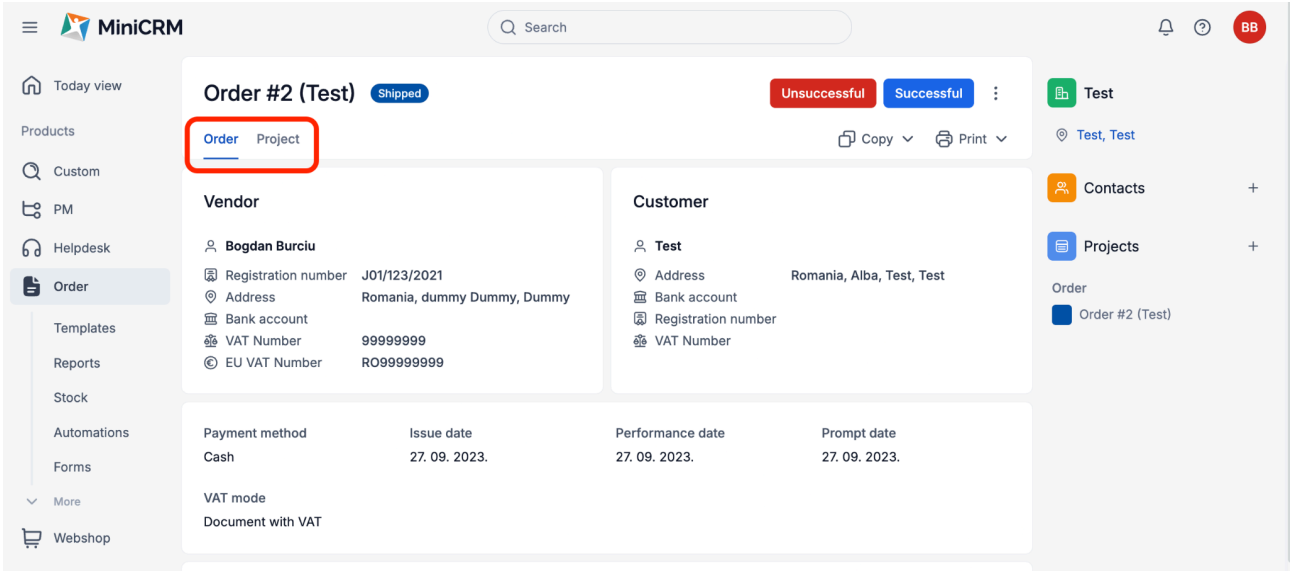
In MiniCRM, an opportunity card will be connected to a contact person (who is called the ***primary contact***) and if that specific card has more than one contact, the rest of the contacts will be displayed below the primary contact under a section called "*Other contacts*".

To sum up, an opportunity card can have, at most, one primary contact at a time and many other contacts listed under the *Other contacts* section.

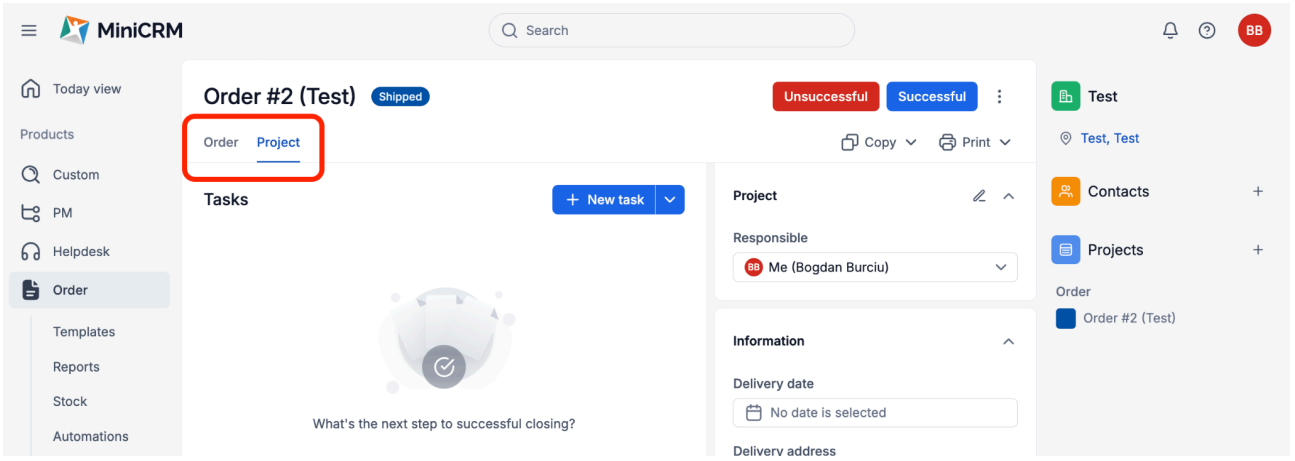
Structure of the cards in special products (invoice, order, offer etc.)

Cards are listed in the Order and Invoice modules as well. Invoices and orders are connected to specific cards so unique fields, tasks and follow-up sequences can be managed on cards, and the order/invoice specific functions operate on a different tab and API endpoint.

In the image below is presented an order where the two tabs are clearly visible, and the *Order* tab is activated:



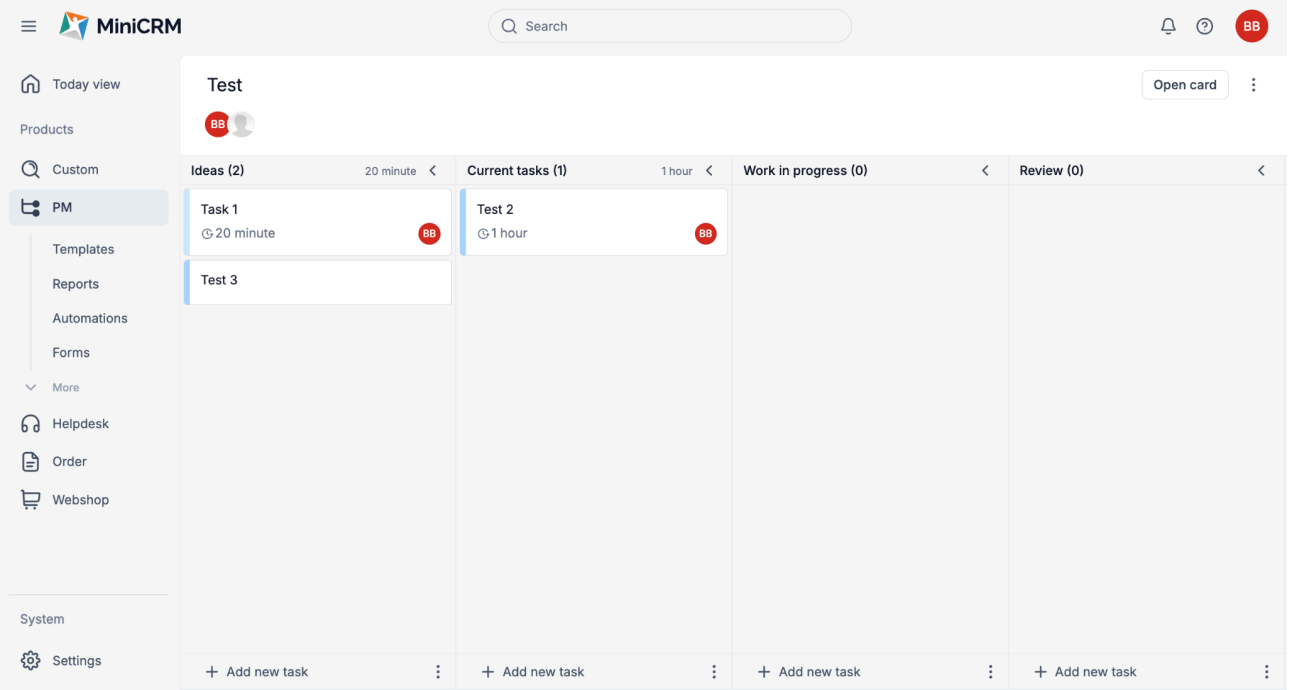
While here, the Project tab (of the order) is active (visible):



The following products have the same structure (two tabs): Invoices, Offer, Order, Delivery Notes, Service Worksheet, Certificate of contract completion.

Project Management specifics

In MiniCRM there is a product called Project Management that has a special view of tasks as described below:

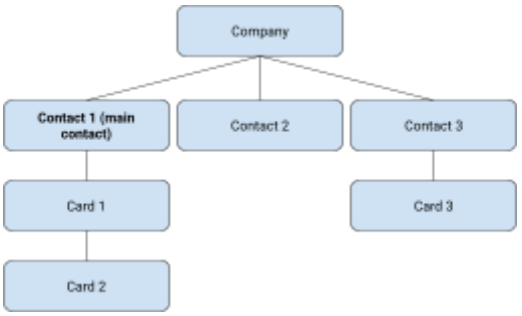


This view is also known as Kanban view and displays the open tasks in specific columns: ideas, current tasks, work in progress, review, pending, done and rejected.

Company and contact (or account)

On the right sidebar of the card, you can see the customer and the contact persons to whom the opportunity card belongs. The customer may be a company or a person, or both – a company with contacts.

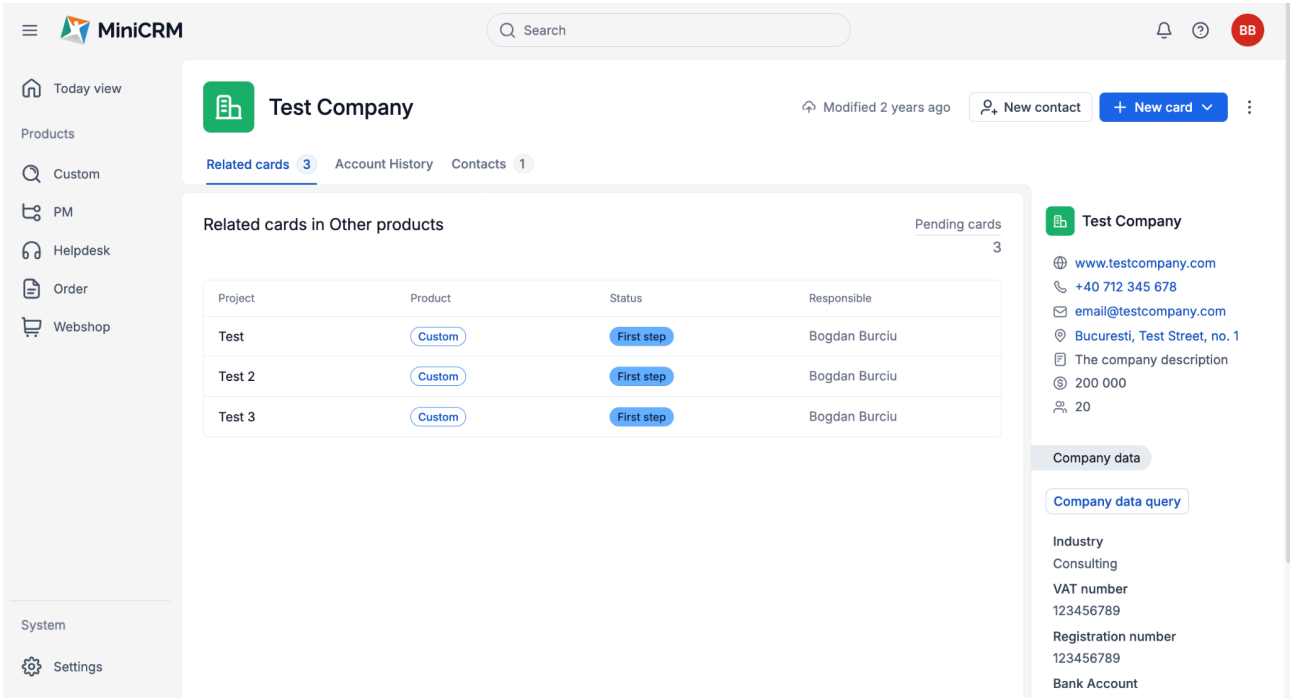
In MiniCRM the cards are assigned to contacts the contacts are assigned to companies. The following image represents these relations:



Another important aspect is that at a moment only one contact can be set as primary contact.

Company card

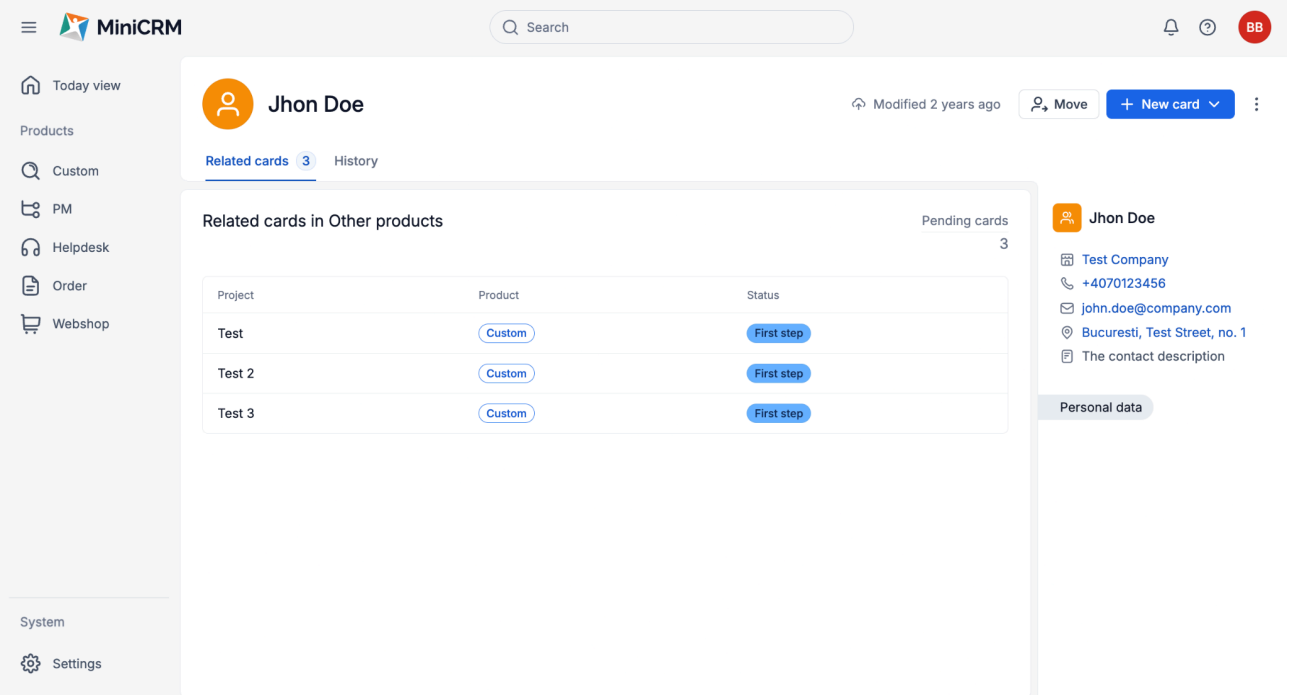
Besides the opportunity card in MiniCRM we also have a company card (or company page) where all the data related to that specific company are registered.



The company card contains information related to the account history (which is the history from all cards related to that specific company), a list of contacts and the company data like website, description, email address, phone number, VAT number, EU VAT number, registration number and others. Please note that you may create custom fields on the company card too, so you may register your own specific information here.

Contact card

Similar to the company card in MiniCRM we also have a contact card. On this card, you will be able to register information related to the contact like first and last name, position of that contact, the contact phone number, email address or physical (postal) address. There is the possibility to add here as well, some custom fields if needed.



To sum up, in API, if you will need to create an entirely new card (with company, contact, and card data) you will need, for this purpose, 3 different API calls:

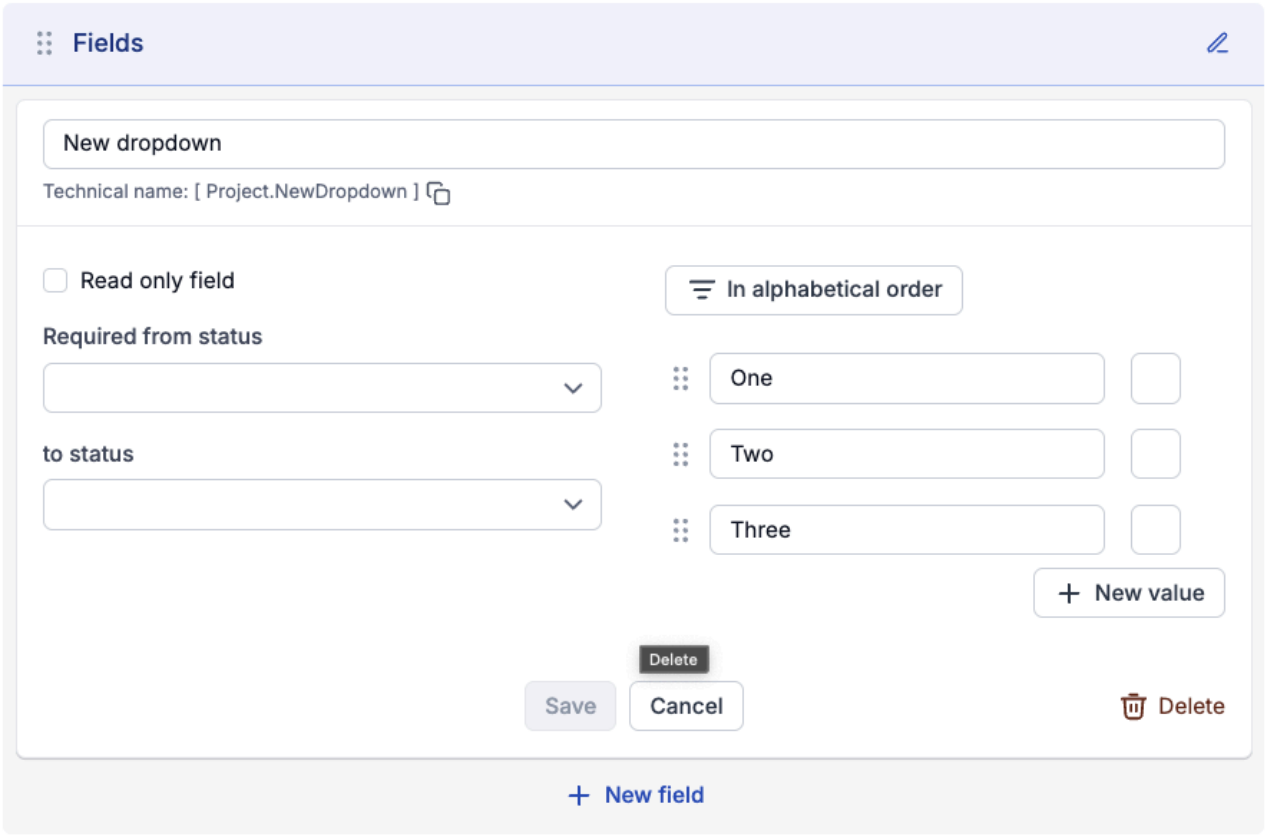
1. To create the company;
2. To create the contact and assign it to the previously created company;
3. To create the opportunity card and assign it to a previously created contact.

Field name

In MiniCRM it is possible for the user to create and set his own fields on the cards. While setting those fields is not the subject of this manual, it is possible that you will need to use those fields for integrations, and you will need to know where to find the field names that are registered in the database.

To get the field's name, you will have to have administrator rights in MiniCRM to be able to reach the fields' configuration page.

From that page please click on the icon to edit the fields and there you will be able to find the registered name for that particular field:



In this case, the name of the field is "***NewDropdown***".

Depending on the card where the field is registered, you may see field names like Contact.*FieldName* or Project.*FieldName*. In integrations, we will use only the name after the dot (we will ignore *Contact.* or *Project.*).

API key generation

REST API key generation

To use our API, you will need to generate an API key. The key can be generated only by an admin (we, as MiniCRM, cannot generate that key for a specific system).

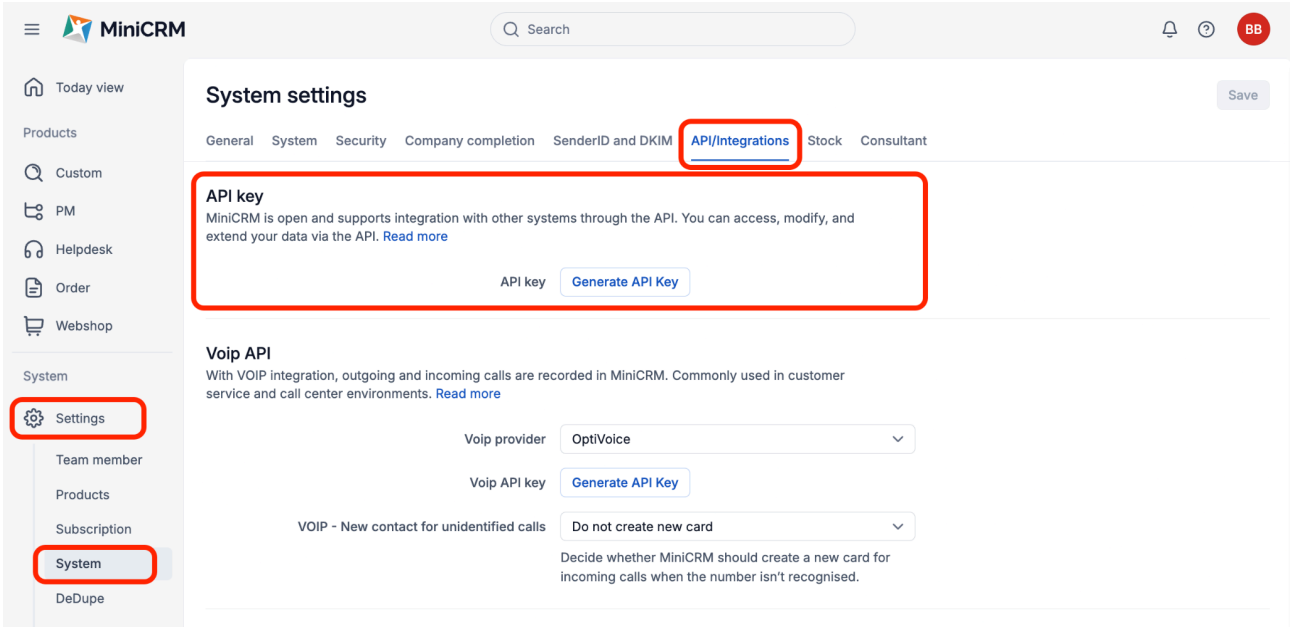
Please save the generated key in a safe place (maybe it's a good idea to use a password generation application like Bitwarden, LastPass etc., they offer the option to have secured notes.

On the key generation page, you will be able to see the API key only once, after that you will not be able to see it or access it (in case you forgot the key). This is why you should save it in a safe place right after the generation.

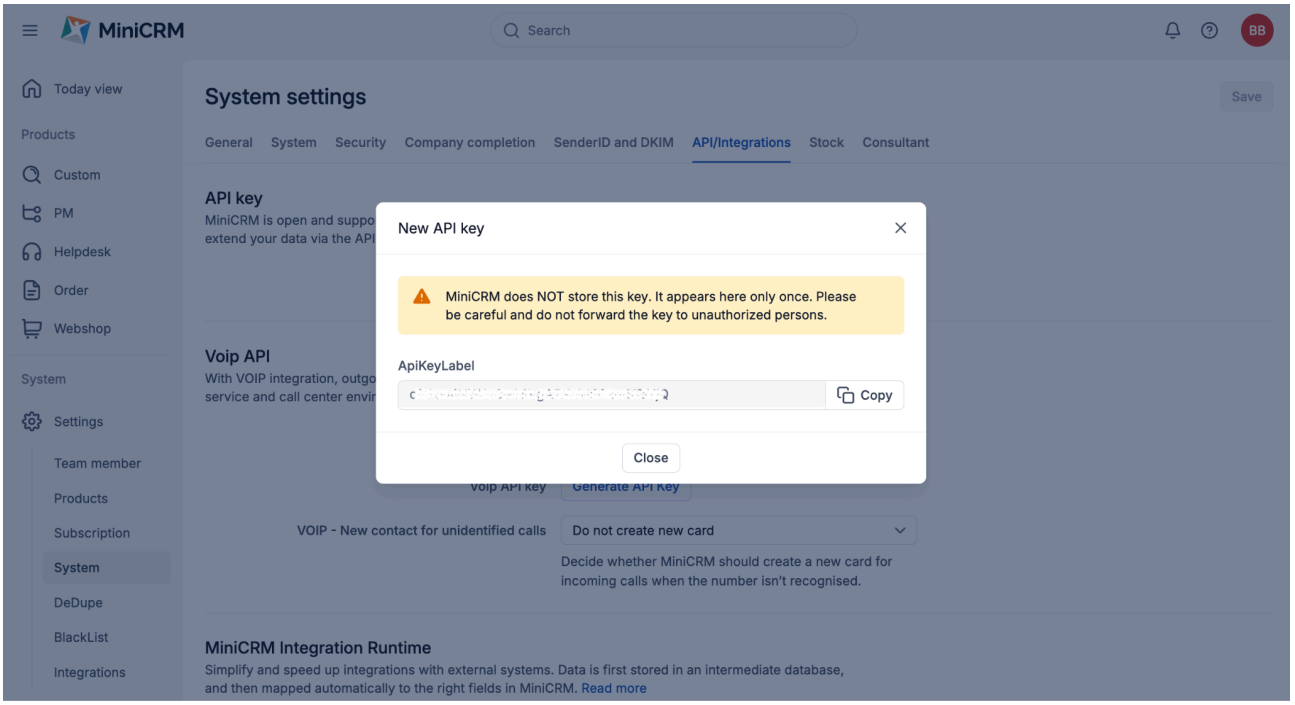
Please **don't share** your key with anyone who is not authorized to have it (even with us as MiniCRM). That key is your "password" to access the API, so it should be protected since anyone who has the key can access your data.

If you lost your API key, and you need it, the only option is to generate a new one instead. If you have some active integrations with MiniCRM, this will mean that you will need to update the existing integrations to use the new key.

The API key can be generated directly from the MiniCRM and you should access the settings and then the system page:



Then the generated key will be displayed (*the key will be displayed only once*):

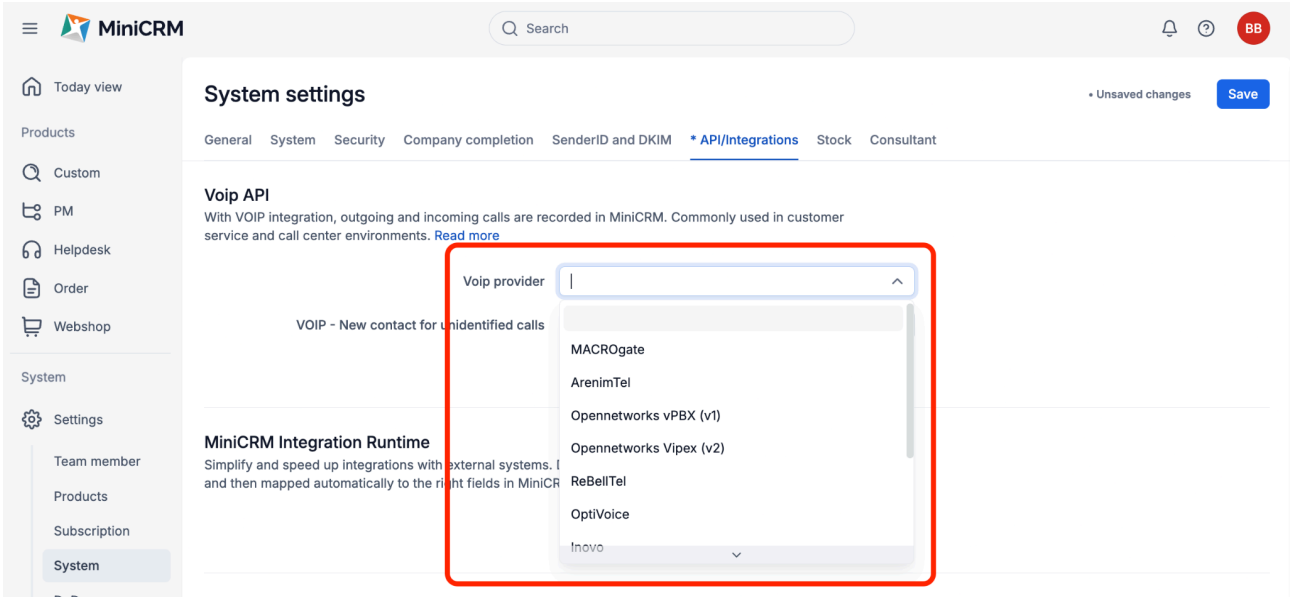


After that, you will be able to only update (generate a new key) or delete the existing key.

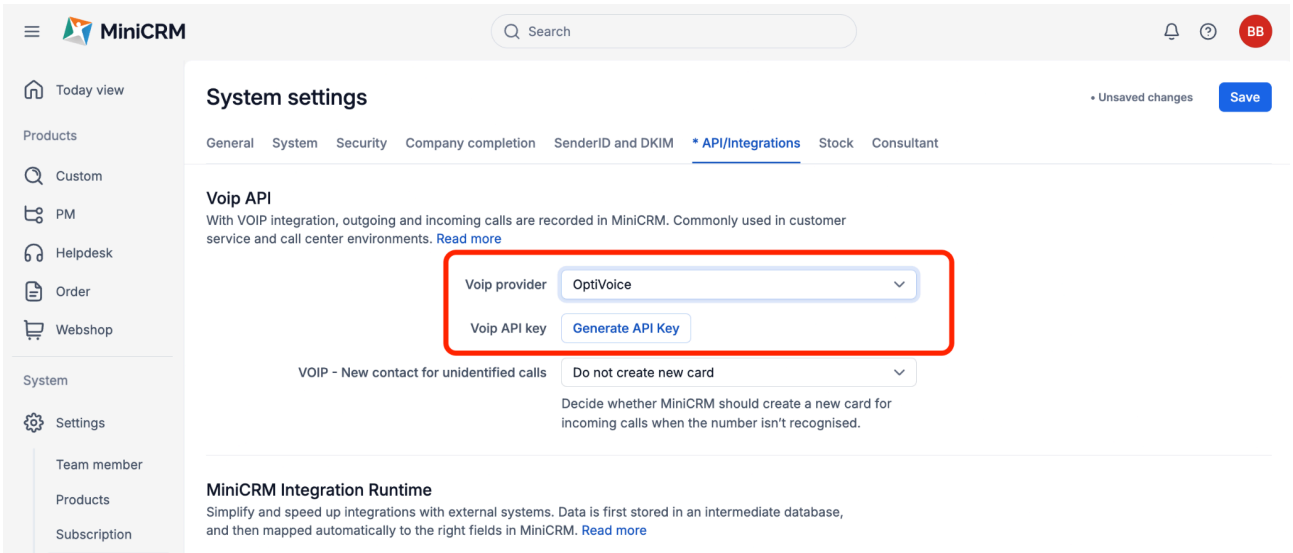
VOIP API key generation

It is possible to generate a VoIP API key for an already registered provider. If you want to register as a VoIP services provider in MiniCRM please check the below chapter for details (here).

To use our VoIP API, you will need to generate the VoIP API key. Firstly, you will need to select the VoIP service provider:



Then you will be able to generate the VoIP API key as usual. The same logic is applied here also (you should save and store the key in a safe place and share it only with the authorized persons only).



Register your company as a VoIP services provider in MiniCRM

To be registered as a VoIP services provider in MiniCRM you will need to provide us the following information about your services:

1. The name of the service or the name of the company;
2. The IP address (or addresses) that you are using (to whitelist them).
3. If you offer the call recording option, you will need to provide us a recording URL to enable the customers to listen to the calls directly from MiniCRM. The URL can be something like https://your.domain.com/recordings.php?Rec={%RefereceId%} where the ReferenceId is the phone call recording's unique identifier in the system.

VOIP integration specifics

Please check the chapter that is related to VOIP integration (here).

Production and test environments

At MiniCRM it is possible to have a test environment (or test system) completely separate from the production environment that you are using every day.

The test environment is completely independent of the production environment, it is located on another server, and it can be accessed using a specific URL (you will receive all the details when you will request the test environment to be created).

There are some specifics that you should consider when it is about to request a test environment:

- The test environment can be created on demand;
- The test environment and production environment are completely separated, and they cannot communicate with each other;
- The test environment is loaded based on the production environment (never vice versa);
- If you make some changes in the production environment, and you need those changes to be applied also into the test environment, we can reload the test environment to reflect the changes;
- We recommend that you access the test environment using an Admin user from the production environment (you will be able to access it using the same email address and password as in the production environment). If you need, you can change the credentials for the production environment *after* the test environment is created. The changes will affect only the production environment;
- You will not be able to send emails from the test environment, and the most important mention is that you will not be able to request a password reset from the test environment;
- The data is encrypted into the test environment by default (for GDPR purposes) but on demand we can load clear data on the test environment too.

How to request a test environment?

To request a test environment, you will need to email us to one of the following email address:

- For Romanian customer support: suport@minicrm.ro
- For Hungarian customer support: help@minicrm.hu
- For customers from other countries (please write in English): help-en@minicrm.io

We will provide you all the necessary information you may need as a reply to your email.

INTEGRATION OPTIONS

MiniCRM can be integrated with any application that can support REST API communications with the specification that in a certain integration, MiniCRM can be only the “passive” part that expects requests. MiniCRM will not initialize requests to external REST APIs.

The system can be integrated with other applications using some different methods and for various purposes.

We support integrations via REST API and XML synchronization. We also have some specific REST API endpoints for invoices and for call log purposes.

In this manual, we will cover all the integration possibilities for MiniCRM.

REST API vs. XML synchronization

In the following table, we will compare the two main methods of integration that are available into the system.

REST API	XML synchronization
It is used to transfer a small amount of data. There is a limitation of 60 requests per minute in place (please check the API limits chapter)	It is used when a large amount of data needs to be transferred into the system (for example, 100+ cards)
The data are synced almost instantly (depending on your internet speed).	Depending on the amount of data that is synced, the transfer may not be visible right away.
It works as a communication between MiniCRM and a third-party app, where that app can create requests for MiniCRM and the system will respond with the required data.	It works only as a one-way data transfer (from a third-party app to MiniCRM).
Security measures can be implemented by the third-party app. The system uses HTTPS protocols.	The XML file should be publicly available to be accessed and processed by the system[1].
To create an entirely new card, it will need 3 different requests (one for the company card, one for the contact card and one for the opportunity card).	To create an entirely new card (with company, contact, and opportunity card), one single request is needed.

The above table should offer you the information you need to be able to choose the correct way of integration with our system.

For example, if you plan to integrate a webshop with MiniCRM, using the above table you probably will find that the most appropriate solution will be to use XML synchronization since, in most common cases, you will want to synchronize a large amount of data at once.

However, if you cannot decide the proper method for what you are planning to do, please feel free to contact us:

- For Romanian customers: suport@minicrm.ro
- For Hungarian customers: help@minicrm.hu
- For customers from other countries (please write in English): help-en@minicrm.io

We will gladly help you to decide the most appropriate method based on your need.

[1] There can be set some IP restrictions for security reasons. We can provide our servers IPs if needed.

Invoice API

This endpoint enables the customer to interact with invoices in MiniCRM. You will be able to manage your invoices and perform operations like creating new invoices, searching for invoices, and marking them as paid or storn them.

Also, by using the endpoint, the customer can update invoice card fields. You may find detailed information by consulting the dedicated chapter (here).

Call log endpoint (VOIP integration)

The endpoint is useful in situations where you need to use a centralized phone service like VoIP (Voice Over IP) solutions.

This endpoint will enable you to store phone call details like the phone number that was called, by whom, when and the phone call duration.

Moreover, you will be able to record and listen to phone calls if you will need this feature and if your VoIP solution allows you to do that[2].

Please note that, based on your country's legislation and/or European legislation, you may need to inform and get the consent of your customers about the phone call recording policy.

After the call is finished, the details about that specific call will be recorded in the most recently updated card history section.

We will record phone call details like:

- Who called/received the phone call (MiniCRM user);
- The phone call status (answered or missed)
- The contact name;
- The phone number that was involved in the call;
- The phone call duration;
- The phone call registration (this feature depends on the VOIP solution - some of them may not include this possibility).

You can find more information on the topic by reading the dedicated chapter below (here).

[2] This option depends on the country's legislation where your headquarters is located, and also it may depend on the EU legislation on the GDPR topic. We only provide this feature, but the customer is that who is responsible to consider those legal things and who needs to be sure that he informs his clients that the phone calls are recorded.

Webhooks

There can be some situations when an external system or application needs to react to a change that happens in MiniCRM. In those cases, we recommend the webhooks.

Some examples of when webhooks may be helpful:

- Complex CRM automation - an external system is informed about a new lead, ticket, project, invoice, offer, order etc. via webhook. That system can react to that information in various ways.
- Sync with an external system - when a card, project, ticket, invoice, etc., is updated in MiniCRM, the external system is informed too about the change.

The changes that happen on the cards, companies, or contacts can be sent using a POST request to an API endpoint given by the customer.

The endpoint should respond with an HTTP response (200 OK) to consider that the data was successfully processed.

In case that we don't receive an HTTP response 200, we will consider that an error occurred, and we will make another 6 attempts later with the following delay:

1. 10 seconds;
2. 1 minute;
3. 5 minutes;
4. 15 minutes;
5. 30 minutes;
6. 1 hour.

The delay is calculated from the first attempt, therefore, after the card was modified, the receiving party has 1 hour 51 minutes and 10 seconds to process the data. If it is not processed over this time, we will not make another attempt, and we will consider the changes obsolete.

Webhook filter criteria

To avoid overload, it can be filtered by many parameters that determine when is a specific endpoint is called.

IMPORTANT: If the program that is called makes modifications via the MiniCRM REST API based on the data received, it can receive repeated notifications. This way an infinite loop can occur. In this case, be cautious when selecting the call criteria. The endpoint should only pay attention to the changes of those fields that won't be changed by it anymore.

Type

- Contact (person and company as well)
- Project (card)

Product

It can only be interpreted in the case of cards (project). Notification is sent if a modification was made in a selected module.

If you would like to receive notifications about more than one product but not about all of them, you can set more webhooks, one for every product separately.

Fields

Filter the change of specific fields. The endpoint is called if one of the listed fields has been changed. If no field is added to the endpoint, the endpoint is called if any of the fields has changed (please find here details about how to get field names).

Products and field filters can be combined.

Webhook examples

The Sales product has the ID = 3, and I would like that only the webpage that displays content to my subscribers would be notified if a subscription was made and/or canceled.

```
Type: Project
CategoryId: 3
Fields: ["StatusId"]
ApiUrl: https://$User:$Pass@example.com/StatusChanged/
```

The Helpdesk product with ID = 4, and I would like that new tickets would be preprocessed automatically by a script so we could create complex follow-up sequences.

```
Type: Project
CategoryId: 4
Fields: ["CreatedAt"]
ApiUrl: https://$User:$Pass@example.com/TicketAdded/
```

Example:

```
{
  "Id": "10",
  "Type": "Project",
  "Data": {
    "CategoryId": "8",
    "ContactId": "16",
    "MainContactId": "15",
    "StatusId": "11684",
    "UserId": "2",
    "Name": "The card name goes here",
    "CreatedBy": "2",
    "CreatedAt": "2018-03-02 23:35:41",
    "UpdatedBy": "2",
    "UpdatedAt": "2018-03-02 23:55:42",
    "InternalUrl": "https://r3.minicrm.hu/123456/#Project-8/10"
  },
  "Changed": [
    "StatusId",
    "UserId",
    "Name",
    "UpdatedAt"
  ]
}
```

How to set webhooks

Currently, webhooks are running in a live-test mode. Our internal systems are also connected in a live environment by using this.

However, webhook settings are not available on the web interface. Contact our helpdesk (you may find our contact details here) to request additional information and the setting of the webhook.

As more and more real-life experiences are collected, the webhook settings options and the content of the messages could be changed.

Built-in integrations

Most of these built-in integrations are add-ons, and they may not be activated for your system. To use them, you will need to activate them on the Subscription page from settings directly from your MiniCRM system.

MiniSync Integration

MiniSync is exclusively available for the Professional and Enterprise platforms, offered only through annual or multi-year subscriptions or loyalty agreements.

It's important to note that we can schedule/undertake integrations more quickly if they are compatible with our existing infrastructure.

MiniSync is a platform designed to simplify and expedite the integration processes. It utilizes an intermediary database to store data from third-party companies. These data are then mapped to the corresponding fields within MiniCRM.

Benefits of MiniSync

You don't need to navigate the technical complexities of MiniCRM when setting up SyncFeed. You can submit data in the format/structure received from a third party without adhering strictly to MiniCRM specifications.

There's no need to fully comprehend our 100+ pages of API documentation, MiniCRM's data structure, or deal with identification/duplication, API limitations, etc. If you have a "data feed" in your own structure, it's sufficient for us. We take care of the mapping logic and submit it to your MiniCRM system.

When to Choose MiniSync

It's essential to understand that integration is fundamentally a collaborative process involving your participation and that of the third party, preferably a technical person from the software you're using. While our tool can assist you in fetching data from another software, processing and retrieving data from the other software are not solely our responsibility. This requires cooperation and support from the third party.

Use cases for MiniSync

Continuous data synchronization between two systems (e.g., MiniCRM and an ERP).

Limited resources: Ideal for clients lacking internal software design or development capacity and lacking detailed technical knowledge about MiniCRM.

Your tasks in the integration process:

- Imagine and document your needs: Note down your requirements – data fields, workflow ideas, and how you want things to synchronize.
- Consult with the third party about integration. Discuss technical details, fees, and ensure that the data connection will always remain active and supported in the future.

Responsibilities of the Third Party:

While we offer various connectors for data retrieval from the third party, it's crucial that they provide continuous access to up-to-date information.

These connection forms include:

- **LocalFile Connector** - provides a unique endpoint for the third party to upload fresh data.
- **HttpDownload Connector** - requires an endpoint from the third party from which we can download fresh data.
- **MySQL Connector** - requires access to the database of the respective system to extract the necessary data. This is a more time-consuming connection form, as we need to understand the database structure.

Integration with calendars

Google calendar

The integration allows you to synchronize the tasks from MiniCRM with your Google calendar account and display them into the calendar. The only thing to do is to set the integration, then the tasks will be automatically synchronized with it.

Details about this addon can be found on our Help pages:

- Romanian: https://www.minicrm.ro/help/sincronizarea-cu-calendarul/
- Hungarian: https://www.minicrm.hu/help/google-naptar-szinkronizalas/

Other calendars

MiniCRM can synchronize tasks in some other calendars (like Outlook and/or iPhone calendar). The list of available calendars may change over time, so please contact us if you need specific information at the time you read this information.

You may also check our Help pages and check the Integration section there to see what calendars are listed:

- Romanian: https://www.minicrm.ro/help/
- Hungarian: https://www.minicrm.hu/help/

Gmail

This addon enables you to directly send emails from your inbox to MiniCRM history for a specific customer (you have to open an email from the customer first).

Also, you may add new cards, tasks or to open MiniCRM directly from Gmail interface.

Details about this addon can be found on our Help pages:

- Romanian: https://www.minicrm.ro/help/integrare-cu-gmail/
- Hungarian: https://www.minicrm.hu/help/minicrm-integracioja-a-gmail-webes-feluletevel/

Outlook

This integration is very similar with Gmail one, and you may find the setting details on our Help pages.

Details about this add-on can be found on our Help pages:

- Romanian: https://www.minicrm.ro/help/integrare-outlook/
- Hungarian: https://www.minicrm.hu/help/minicrm-kontaktok-szinkronizalasa-outlook-nevjegyekkel/

Facebook ads forms

Using this built-in integration, you will be able to collect lead data from Facebook paid ads forms directly into MiniCRM.

Details about this add-on can be found on our Help pages:

- Romanian: https://www.minicrm.ro/help/integrarea-cu-facebook-lead-ads/
- Hungarian: https://www.minicrm.hu/help/facebook-lead-ads-connector/

Gravity Forms Integration

If you want to gather all your leads through your WordPress-based website into your MiniCRM system with custom web form design, this integration is the best way up-to-date way.

Gravity Forms gives you the ability to create styled web forms, and with this integration connect it to your system and after somebody submits the form they will appear in your system and the cooperation, automated processes, sales, etc., can begin.

All you have to do is to download our Gravity Forms plugin, create two identical web forms in your system and in Gravity and connect them together with the short code.

For more information, please read our help pages:

- Romanian: https://www.minicrm.ro/help/integrare-gravity-forms/
- Hungarian: https://www.minicrm.hu/help/gravity-forms-integracio/

WooCommerce

This integration may be used in case you have a WordPress and WooCommerce webshop to synchronize your orders directly into MiniCRM.

The integration requires two products in MiniCRM: webshop and orders.

For more and technical details, please read our dedicated help pages,

- Romanian: https://www.minicrm.ro/help/integrare-woocommerce/
- Hungarian: https://www.minicrm.hu/help/woocommerce-integracio/

Shoprenter

If you want to connect your MiniCRM system with one of the most popular webshops in Hungary, Shoprenter you can with our boxed integration.

With this integration, you can synchronize all your orders and customers from your Shoprenter webshop into two different products in your system: Webshop (customer database) and Orders.

For more details, please check our help page in Hungarian:

- Hungarian: https://www.minicrm.hu/help/shoprenter-integracio/

UNAS

This webshop integration is available in multiple countries and works similarly to other webshop integrations.

Your customers and orders will be synchronized via MiniCRM API connection into the Webshop and Orders modules in your MiniCRM system.

For technical details, please read our dedicated help pages,

- Romanian: https://www.minicrm.ro/help/sincronizareunaswebshop/
- Hungarian: https://www.minicrm.hu/help/unas-webshop-integracio/

OpenApi

OpenApi is similar to ListaFirme from Romania and this service allows you to have updated data about the companies.

The data came from the https://openapi.ro/ database.

After activation, the system will scan the (in any way) created companies details and synchronizes the closest company record with up-to-date details from the Cégjelző database.

This will sync the following details of the companies:

- Tax identification number
- Company name
- Address
- Company register
- Number of employees
- Initial capital
- Annual revenue
- Year of establishment
- Scope of activities

Later on, the system will periodically scan the companies that are in your database and compare their details with the up-to-date details on Cégjelző and update the information in your system if needed.

For more details, check our help page:

- Romanian: https://www.minicrm.ro/help/openapi/

Cégjelző

This is our boxed solution on how the MiniCRM system can keep your database, and the partner companies' details, up to date regarding companies data.

This solution works via API and gets data from the https://cegjelzo.com/ database.

After activation, the system will scan the (in any way) created companies details and synchronizes the closest company record with up-to-date details from the Cégjelző database.

This will sync the following details of the companies:

- Tax identification number
- Company name
- Address
- Company register
- Number of employees
- Initial capital
- Annual revenue
- Year of establishment
- Scope of activities

Later on, the system will periodically scan the companies that are in your database and compare their details with the up-to-date details on Cégjelző and update the information in your system if needed.

For more details, check our help page:

- Hungarian: https://www.minicrm.hu/help/ceginfo-szolgaltatas-integracio-cegjelzo-creditinfo/

NAV (Bejövő számlák)

With this custom integration, we provide you the opportunity to sync in all the invoices that were issued for you and also the partners who issued them.

With this integration, you have the possibility to create a cash flow report that automatically updates itself in Google Sheets if you have the required add-ons. For more help and information regarding the possibilities, please feel free to contact our Helpdesk.

For more information about the integration, check our help page:

- Hungarian: https://www.minicrm.hu/help/nav-bejovo-szamlak-integracio/

BestJobs / eJobs integration

These integrations may help you gather information about the candidates from that platforms directly into your MiniCRM system. This way you will have all the information centralized and may continue your recruitment process from MiniCRM.

For more information about the integration, check our help page:

- BestJobs: https://www.minicrm.ro/help/integrare-bestjobs/
- eJobs: https://www.minicrm.ro/help/integrare-ejobs/

SmartBill integration

You may use your MiniCRM and SmartBill accounts together creating offers or orders, for example, from MiniCRM and then automatically generating the invoice based on those documents in SmartBill.

You can also use the stock item's module from SmartBill.

For more information about the integration, check our help page:

- Romanian: https://www.minicrm.ro/help/integrare-smart-bill/

REQUIREMENTS

To use our API, you will need to authenticate. In the authentication process, you will need the following two important elements:

1. The SystemID (please check the Basics chapter for more details - details here)
2. The API key (also explained in the Basic chapter - details here)

Important: He highly recommends that before you start your integration, you run some of our examples that are presented in this manual because this will help you to better understand how our implementation works.

Pagination

Those endpoints through which it is possible to filter/search in the system give back the results according to a unified framework.

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Project?CategoryId=3"
{
	"Count": 42,
	"Results": {
		"2009": {
			"Id": 2009,
			"Name": "Test card",
			"Url": "https://r3.minicrm.hu/Api/R3/Project/2009",
			"ContactId": 7461,
			"StatusId": 2540,
			"UserId": 22404,
			"Deleted": 0,
			"BusinessId": 7453
		},
	...
	}
}
```

- Count: In the response, the results get in an array and the number of results can be found on its Count key.
- Results: Under the Results key, the results can be found in separate arrays. Only the most
- Fundamental data is listed.
- URL: Use the address got in the URL field to access every piece of information that is available about an item via API.

In the case of an API search, 100 results are shown on one page. It is possible to page by using the Page parameter. In the case of the API, paging starts from 0, so the second page can be accessed by using the Page=1 parameter.

Example:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Project?CategoryId=5&Page=1"
{
	"Count": 181,
	"Results": {
		...
	}
}
```

cURL usage

> If you have version 1803 (April 2018 update) or later of Windows 10, curl *is installed by default.*

Anyway, if you need help with cURL installation, please check the following link: https://everything.curl.dev/.

In our examples, we suppose that the SystemID and the API key have been set in the environment variables named SystemId and ApiKey.

Example:

```
$ export SystemId=12345
```

then

```
$ export ApiKey=YourAPIKeyHere...
```

then

```
$ curl --user $SystemId:$ApiKey -I -X GET "https://r3.minicrm.hu/Api/R3/Category"
```

In the example above, $ represents the command line. The commands can be run by copying the characters that come after $ into the terminal program, and they produce a result/outcome that is equal to the example.

Curl is a useful tool to get to know the MiniCRM API better and discover the endpoints that are significant for you. You can learn quickly what kind of responses are given by

MiniCRM to different requests, so after examining the examples, it will be easier for you to implement them into your own system.

The MiniCRM REST API delivers a JSON response without any formatting. Computers can interpret these responses easily, however, they are not easily understandable for humans:

```
$ curl --user $SystemId:$ApiKey https://r3.minicrm.hu/Api/R3/Category
{"1":"Sales","2":"Invoice","3":"Helpdesk","4":"Order","5":"Webshop","6":"Offer"}
```

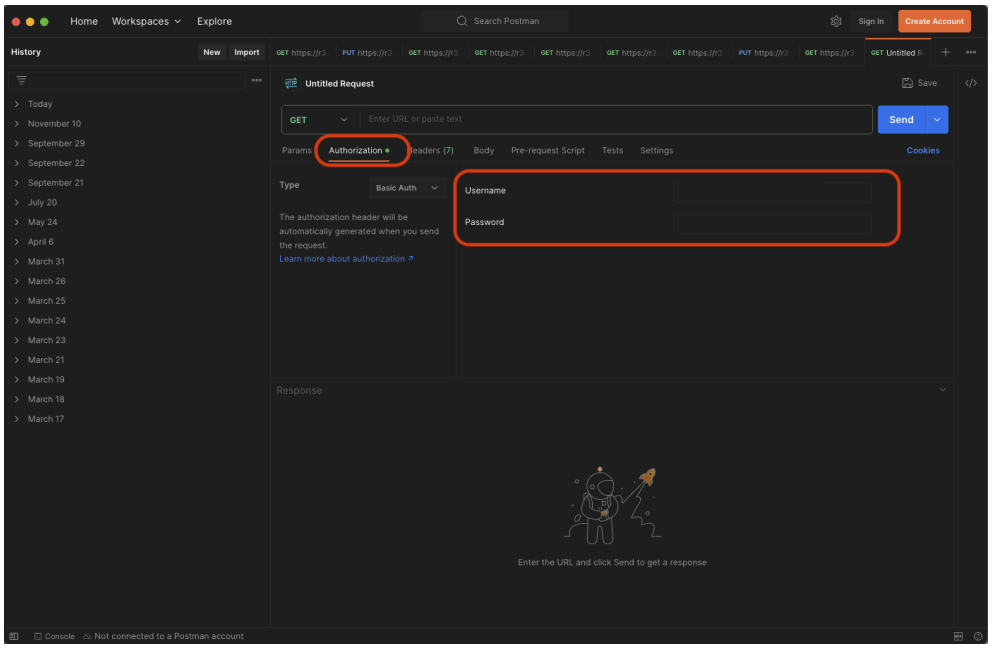
You can use jq (https://jqlang.github.io/jq/) to understand API responses easier:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Category" | jq
{
      "1": "Sales",
      "2": "Invoice",
      "3": "Helpdesk",
      "4": "Order",
      "5": "Webshop",
      "6": "Offer",
}
```

Reminder: MiniCRM API uses a Unicode code page with the UTF-8 representation method. Every incoming parameter's encoding is UTF-8, and every response is given as UTF-8.

Postman

The MiniCRM API can be used with Postman (https://www.postman.com/) also. Postman is a graphical tool to work with REST API systems, and the configuration for the MiniCRM API is pretty simple.



You will need to set the authentication *Type* to **Basic** and then to complete the *Username* and the *Password*.

For the username, you will use your MiniCRM SystemID and for the password, you will use your MiniCRM API key.

After that, you need to use our API endpoints in the Postman URL field without using the full URL. So, instead of the complete cURL URL:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Contact?Name=Name"
```

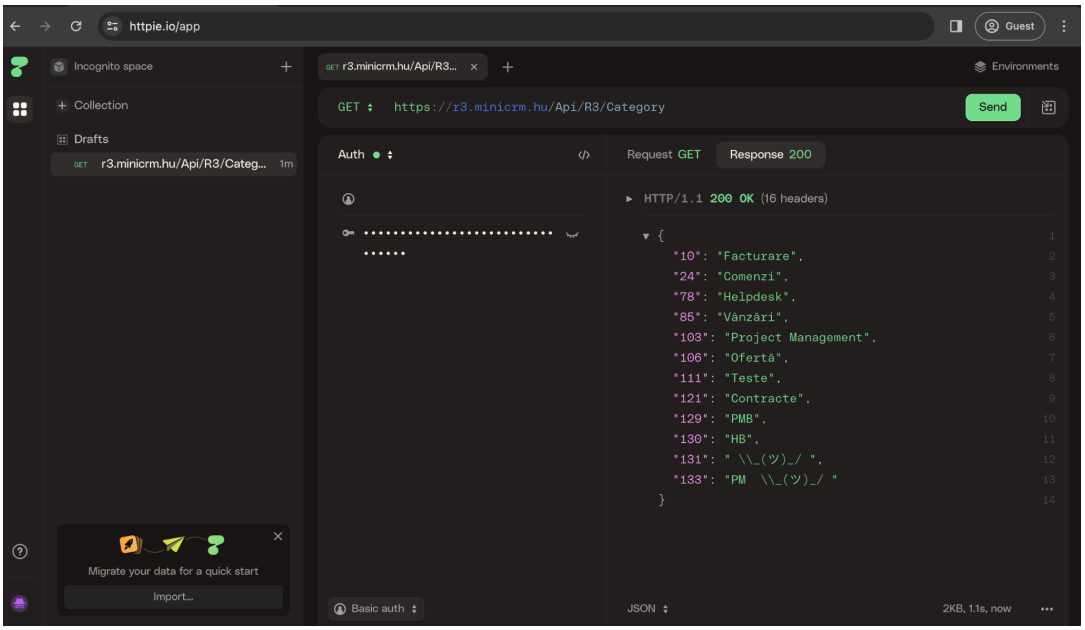
In Postman, you will use:

```
https://r3.minicrm.hu/Api/R3/Contact?Name=Name
```

HTTPie

HTTPie (https://httpie.io/) is an online REST API tool, very similar to Postman, with the advantage of using it online (without having to install it on your computer).

The interface is similar to Postman, and you'll have to begin with the credential's configuration, then you can use the application to run API requests.



Error messages and error handling

The MiniCRM API is using the standard HTML response error codes, so it is easy to understand and debug the integrations.

Error codes and their meaning

Make sure the values sent in checkboxes or dropdowns already exists in the system as possible values, since they are fields with predefined values (it is **not** possible to create or modify their values via API).

200 - OK – It is improper to say that this is an error code, since 200 OK means that the request was correct and the system responded accordingly.

400 - Bad Request – the request may contain parameters that are not recognized by the system (check the typos). Also, this error code is used when the system expects a parameter and that specific parameter it is not exist in the request. Additionally, be advised that our API is case-sensitive (for example, *FieldName* **is different** from *fieldName*).

401 - Unauthorized: Access denied – Is the SystemId and the API key you gave correct? Did you change your subscription plan? If you downgrade to another plan that doesn't include MiniCRM REST API, you will also receive this error message.

404 - Not Found – the specified URL is not found by the system. In most cases, there is a typo.

405 - Method not allowed – the system returns this error when you send a request using the wrong method (POST instead of PUT, for example).

429 - Too many requests – you exceeded the API limits (60 requests/minute). Please check the chapter about limits for more specifications.

500 - Internal Server Error – this is an internal processing error. This may be generated when bad requests are made (for example, you are trying to set a non-existing value for a field).

API limits

You should be aware that the MiniCRM API has some limitations:

1. **60 requests / minute** for all the standard API endpoints (URLs that start with *https://r3.minicrm.hu/Api/R3*);
2. **180 requests / hour** for XML synchronization requests;
3. **No limit** for the invoice's endpoint (https://r3.minicrm.hu/Api/Invoice);

If the number of requests exceeds our API limits, the server will respond with the *HTTP 429 "Too Many Requests"* error code.

> API limits can be increased to 120 or 180 requests/minute based on a request. You can get more details on this topic by contacting us.

Encoding

The MiniCRM API uses a Unicode code page with the UTF-8 representation method. Every incoming parameter's encoding is UTF-8, and every response is given as UTF-8.

Request format

All the requests made should be formatted as a JSON serialized array.

Results pagination

Those endpoints through which it is possible to filter/search in the system give back the results according to a unified framework:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Project?CategoryId=3"
```

```
{
  "Count": 42,
  "Results": {
    "1234": {
      "Id": 1234,
      "Name": "Test card",
      "Url": "https://r3.minicrm.hu/Api/R3/Project/2009",
      "ContactId": 123,
      "StatusId": 1234,
      "UserId": 12345,
      "Deleted": 0,
      "BusinessId": 1234
    }
  }
}
```

- **Count**: In the response, the results get in an array and the number of results can be found on its Count key.
- **Results**: Under the Results key, the results can be found in separate arrays. Only the most fundamental data is listed.
- **URL:** Use the address got in the URL field to access every piece of information that is available about an item via API.

In the case of an API search, 100 results are shown on one page. It is possible to page by using the Page parameter. In the case of the API, paging starts from 0 (zero), so the second page can be accessed by using the Page=1 parameter.

Example:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Project?CategoryId=5&Page=1"
```

```
{
      "Count": 181,
      "Results": {
            ...
      }
}
```

Other specific limitations

Item-based products

1. We only offer endpoints for invoices, orders, and offers.

To-do endpoint

1. You can create requests only for the opened or closed tasks (you will not be able to reopen a task via API).
2. You will not be able to create recurring tasks using the API.
3. You can't add comments on the tasks via API.

Templates

1. You will not be able to create a new template via API.

Contacts

1. You will not be able to move contacts between companies via API.

Addresses (postal)

1. You cannot delete addresses via API.

Companies

1. You cannot delete companies using API.

API ENDPOINTS

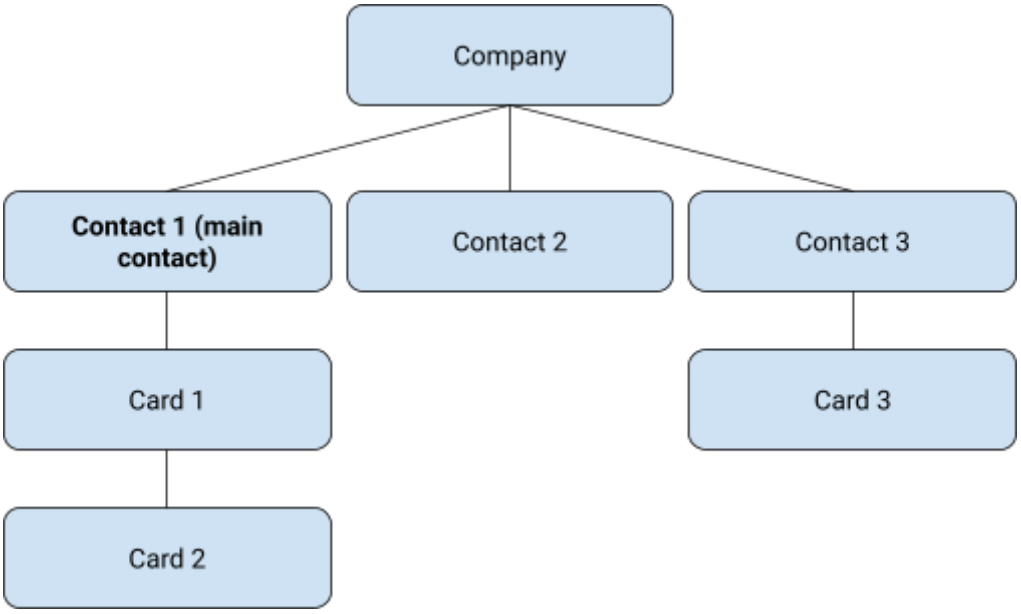
In the following pages, we will describe the API endpoints that we provide currently and the specifics for each of the endpoints. Please don't hesitate to contact us if you have any questions regarding this topic.

You may find our contact information at the end of this document.

Company / contact endpoint

First, we should differentiate the *contacts* from the *companies*. One company can have multiple contacts assigned, while one contact can be assigned only to one company at a time.
On the other hand, the cards can be assigned to a contact or to a company (but without a contact). If a company has at least one contact, the card you will create should be assigned to that contact.

In MiniCRM, the hierarchy between companies, contacts, and cards is as in the image below:



So, the cards are assigned to a contact, and one contact is assigned to a company. It can exist in some special cases when the card may be assigned to a company directly, but this is not the most correct scenario, so, in integrations, you should always try to assign the cards to contacts and contacts to companies.

Create a new company / contact card

To be able to create a new company or a new contact, it will need:

- To PUT a JSON encoded request for setting data
- API endpoint: **https://r3.minicrm.hu/Api/R3/Contact**
- HTTP method: **PUT**
- Data type: **JSON**

When creating a new company or contact, the type of the card should be specified:

- **Business** - for companies
- **Person** - for contacts

To assign a contact to an existing company, the *BusinessId* parameter should be used.

Create a new company

To create a new company, the name of that company is mandatory:

```
$ curl -XPUT --user $SystemId:$ApiKey https://r3.minicrm.hu/Api/R3/Contact -d '{
  "Name": "Company Name Here",
  "Type": "Business",
  ...
}'
```

In case of success, the system will respond with the newly created company ID:

```
{
  "Id":12345
}
```

Create a new contact

For the contact, at least the first name or last name is mandatory:

```
$ curl -XPUT --user $SystemId:$APIKey https://r3.minicrm.hu/Api/R3/Contact -d
'{
      "FirstName": "ContactFirstName",
      "LastName": "ContactLastName",
      "BusinessId": 1234, - OPTIONAL (please check the details above)
      "Type": "Person",
      ...
}'
```

In case of success, the system will respond with the newly created contact ID:

```
{
  "Id":12345
}
```

Search for companies / contacts

Search based on name

This endpoint will search for a specified company / contact name, and it will return all the companies and/or contacts that match the search parameter.
You may differentiate the companies from the contacts by following the *type* parameter.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Contact**
- Parameter: **Name** (company/contact name): *You can specify a partial name as well*
- HTTP method: **GET**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Contact?Name=Name"
```

The system will respond with the company/contact details (company in this example):

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Contact?Name=John
Doe Ltd"
```

```
{
    "Count": 1,
    "Results": {
        "37146": {
            "Id": 36721,
            "Name": "John Doe Ltd",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/36721",
            "Type": "Business",
            "Email": "contact@example.com",
            "Phone": "+40000000000"
        }
    }
}
```

An example for the contact (person) details:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Contact?Name=John
Doe"
{
    "Count": 1,
    "Results": {
        "37147": {
            "Id": 37147,
            "Name": "John Doe",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/37147",
            "Type": "Person",
            "Email": "john.doe@example.com",
            "Phone": "",
            "BusinessId": 37146
        }
    }
}
```

Search based on the email address

This endpoint will search for a specified company / contact email address, and it will return all the companies and/or contacts that match the search parameter.
You may differentiate the companies from the contacts by following the *type* parameter.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Contact**
- Parameter: **Email** (company/contact email address)
- HTTP method: **GET**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Contact?Email=Email"
```

The system will respond with the company/contact details:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Contact?Email=john.doe@test.com"
```

```
{
    "Count": 1,
    "Results": {
        "37147": {
            "Id": 37147,
            "Name": "John Doe",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/37147",
            "Type": "Person",
            "Email": "john.doe@test.com",
            "Phone": "",
            "BusinessId": 37146
        }
    }
}
```

Search based on the phone number

This endpoint will search for a specified company / contact phone number, and it will return all the companies and/or contacts that match the search parameter.

You may differentiate the companies from the contacts by following the *type* parameter.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Contact**
- Parameter: **Phone** (company/contact phone number).
- HTTP method: **GET**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Contact?Phone=Phone"
```

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Contact?Phone=+40000000000"
```

The system will respond with the company/contact details:

```
{
    "Count": 2,
    "Results": {
        "37146": {
            "Id": 37146,
            "Name": "John Doe Inc.",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/37146",
            "Type": "Business",
            "Email": "",
            "Phone": "+40000000000"
        },
        "37147": {
            "Id": 37147,
            "Name": "John Doe",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/37147",
            "Type": "Person",
            "Email": "john.doe@example.com",
            "Phone": "+40000000000",
            "BusinessId": 37146
        }
    }
}
```

Search based on the update date

This endpoint will search for a specified company / contact that are updated on a specified date, and it will return all the companies and/or contacts that match the search parameter.
You may differentiate the companies from the contacts by following the *type* parameter.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Contact**
- Parameter: **UpdatedSince** (company/contact phone number). *You may use the format: 2023-03-24+10:30:00 (please use the Hungarian timezone).*
- HTTP method: **GET**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Contact?UpdatedSince=UpdatedSince"
```

The system will respond with the company/contact details:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Contact?UpdatedSince=2023-03-24+10:30:00"
{
    "Count": 2,
    "Results": {
        "37146": {
            "Id": 37146,
            "Name": "John Doe Inc.",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/37146",
            "Type": "Business",
            "Email": "",
            "Phone": "+40000000000"
        },
        "37147": {
            "Id": 37147,
            "Name": "John Doe",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/37147",
            "Type": "Person",
            "Email": "john.doe@example.com",
            "Phone": "+40000000000",
            "BusinessId": 37146
        }
    }
}
```

Search based on a specific keyword

This endpoint will search for a specified keyword in every text field from the company / contact card. The system will return only those results that match the whole keyword, and it doesn't matter if you type the keyword with lower, upper or accented letters.

You may differentiate the companies from the contacts by following the *type* parameter.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Contact**
- Parameter: **Query**
- HTTP method: **GET**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Contact?Query=Query"
```

Please check the red color text from the output below.

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Contact?Query=mc"
```

The system will respond with the company/contact details:


```
{
    "Count": 2,
    "Results": {
        "36590": {
            "Id": 36590,
            "Name": "Test contact",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/36590",
            "Type": "Person",
            "Email": "john.doe+mc@example.com",
            "Phone": "+40000000000",
            "BusinessId": 275
        },
        "37147": {
            "Id": 37147,
            "Name": "John (MC) Doe (MC)",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/37147",
            "Type": "Person",
            "Email": "john.doe@example.com",
            "Phone": "+40000000000",
            "BusinessId": 37146
        }
    }
}
```

Search based on a specific field

This endpoint will search for a specified value in the specified field from the company / contact card. By specific field, we refer here to the fields that were added and set by users on the cards.
You may differentiate the companies from the contacts by following the *type* parameter.

You may read again our chapter where we explained how to get the custom field names here.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Contact**
- Parameter: **FieldName**
- HTTP method: **GET**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Contact?FieldName=FieldName"
```

For this example, we will assume that we have a field named "TestCardContact" created in my MiniCRM system on the contact card that contains random text and I want to search for all contacts that have a specific text in that field.

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Contact?TestCardContact=test"
```

The system will respond with the company/contact details:


```
{
    "Count": 3,
    "Results": {
        "44": {
            "Id": 44,
            "Name": "Test John Doe",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/44",
            "Type": "Person",
            "Email": "john.doe@example.com",
            "Phone": "+40000000001",
            "BusinessId": 275
        },
        "37542": {
            "Id": 37542,
            "Name": "Test Jane Smith",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/37542",
            "Type": "Person",
```

```
            "Email": "",
            "Phone": ""
        },
        "37147": {
            "Id": 37147,
            "Name": "Test Mike Johnson",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/37147",
            "Type": "Person",
            "Email": "mike.johnson@example.com",
            "Phone": "+40000000002",
            "BusinessId": 37146
        }
    }
}
```

Retrieve all the contacts from a company

This endpoint will compile the list of all the contacts that are associated with a specified company.
The system will also return the company's data into the results.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Contact**
- Parameter: **MainContactId**
- HTTP method: **GET**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Contact?MainContactId=MainContactId"
```

Example:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Contact?MainContactId=37146"
```

The system will respond with the company/contact details:


```
{
    "Count": 3,
    "Results": {
        "44": {
            "Id": 44,
            "Name": "John Doe",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/44",
            "Type": "Person",
            "Email": "john.doe@example.com",
            "Phone": "+40000000001",
            "BusinessId": 275
        },
        "37147": {
            "Id": 37147,
            "Name": "Mike Johnson",
            "Url": "https://r3.minicrm.hu/Api/R3/Contact/37147",
            "Type": "Person",
            "Email": "mike.johnson@example.com",
            "Phone": "+40000000002",
            "BusinessId": 37146
        }
    }
}
```

Detailed data of a contact / company

The endpoint will return the details of a contact or a company based on the ID of that contact/company.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Contact**
- Parameter: **Id**
- HTTP method: **GET**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Contact/$Id"
```

Example:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Contact/37146"
```

System response in case of a company:


```
{
    "Id": 37146,
    "Type": "Business",
    "Name": "ABC Corporation",
    "Email": "",
    "Phone": "+40 0000 0000",
    "Description": "",
    "Deleted": 0,
    "CreatedBy": "John Doe",
    "CreatedAt": "2022-08-25 12:18:49",
    "UpdatedBy": "John Doe",
    "UpdatedAt": "2023-03-30 09:29:55",
    "Url": "https://www.example.com",
    "BankAccount": "",
    "Swift": "",
    "RegistrationNumber": "",
    "VatNumber": "",
    "Industry": "",
    "Region": "",
    "Employees": 0,
    "YearlyRevenue": 0,
    "EUVatNumber": "",
    "FoundingYear": 0,
    "Capital": 0,
    "MainActivity": "",
    "BisnodeTrafficLight": "",
    "NonGovernmentalOrganization": "No",
    "Tags": []
}
```

System response in case of a person:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Contact/37147"
```


```
{
    "Id": 37147,
    "Type": "Person",
    "FirstName": "John",
    "LastName": "Doe",
    "Email": "john.doe@example.com",
    "Phone": "+40 0000 0000",
    "Description": "",
    "Deleted": 0,
    "CreatedBy": "John Doe",
    "CreatedAt": "2022-08-25 12:18:49",
    "UpdatedBy": "John Doe",
    "UpdatedAt": "2023-04-05 09:56:10",
    "Url": "",
    "BankAccount": "",
    "Swift": "",
    "VatNumber": "",
    "Position": "",
    "Industry": "",
    "Region": "",
    "YearlyRevenue": 0,
    "DataManagement_Consent": "",
    "TestCardContact": "Another test sample text",
    "BusinessId": 37146,
    "Tags": []
}
```

Update data for companies / contacts

This endpoint allows you to update the data for a specified company or contact.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Contact/Id**
- Parameter: **Id**
- HTTP method: **PUT**
- Data type: **JSON**

The request URL should be something like (company update):

```
$ curl -s --user $SystemId:$ApiKey -XPUT "https://r3.minicrm.hu/Api/R3/Contact/37147"
-d '{
    "Email": "suport@example.com"
}'
```

The request URL should be something like (person update):

```
$ curl -s --user $SystemId:$ApiKey -XPUT https://r3.minicrm.hu/Api/R3/Contact/37147
-d '{
 "FirstName":"John",
 "LastName":"Doe"
}'
```

In case of a successful update, the system will reply with the ID of that company / contact:

```
{
    "Id": 37146
}
```

Delete a contact

A contact can only be deleted if it is not set as a primary contact on any card.

- API endpoint: **https://r3.minicrm.hu/Api/R3/PurgePerson/ContactId**
- Parameter: **ContactId**
- HTTP method: **PUT**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey -XPUT
"https://r3.minicrm.hu/Api/R3/PurgePerson/37148"
```

In case of success, the system will reply with a successful message:

```
{
    "Message": "Successfully deleted"
}
```

In case of error, the system will reply with a helpful message:

```
{
    "Message": "The person is marked as the main contact on one or more cards."
}
```

Companies cannot be deleted using this method.

Address endpoint

Using this endpoint, you will be able to manage physical addresses (postal addresses). In MiniCRM both, contacts and companies can have a physical address.

When creating an address, the country, county, and address types are **fixed values** that you can choose from the user interface within the system, so because of this in the JSON, you have to save **the exact value** based on your system's language.

Create a new address

To be able to create a new address, it will need:

- API endpoint: **https://r3.minicrm.hu/Api/R3/Address**
- HTTP method: **PUT**
- Data type: **JSON**

To create a new address, you need to specify a contact/company ID (in MiniCRM the address needs to be linked to a contact or to a company). The ContactId parameter below can contain the contact or the company ID (it doesn't change in case of a company):

```
$ curl -s --user $SystemId:$ApiKey -XPUT https://r3.minicrm.hu/Api/R3/Address -d '{
  "ContactId": 12345,
  "Type": "Headquarter",
  "Name": "Address name",
  "CountryId": "Romania",
  "County": "John Doe",
  "City": "John Doe",
  "PostalCode": 000000,
  "Address": "John Doe, nr. 00",
  "Default": 1
  }
```

In case of success, the system will respond with the newly created address ID:

```
{
  "Id":12345
}
```

When creating an address for the Country parameter, you may send either the country code or the country name: *"CountryId":36* or *"CountryId":"Hungary"*.

Search for address

Search based on contact/company ID

- API endpoint: **https://r3.minicrm.hu/Api/R3/AddressList/ContactId**
- Parameter: **ContactId** (an existing contact or company ID)
- HTTP method: **GET**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/AddressList/ID"
```

The ID parameter can contain a contact ID or a company ID.

The system will respond with the address details:

```
$ curl -s --user $SystemId:$ApiKey https://r3.minicrm.hu/Api/R3/AddressList/12
```

```
{
    "Results": {
        "307": "NoName ( John Doe Address)",
        "310": "Office Address (John Doe Address)"
    },
    "Count": 2
}
```

The address details can also be retrieved in a structured response if needed:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/AddressList/ID?Structured=1"
```

The system will respond with the address details in a structured way:

```
{
    "Results": {
        "307": {
            "Id": 307,
            "Address": "No Name (John Doe)",
            "Url": "https://r3.minicrm.hu/Api/R3/Address/307"
        },
        "310": {
            "Id": 310,
            "Address": "Test Address (John Doe Test Street, no. 123)",
            "Url": "https://r3.minicrm.hu/Api/R3/Address/310"
```

```
        }
    },
    "Count": 2
}
```

Search based on address ID

- API endpoint: **https://r3.minicrm.hu/Api/R3/Address/AddressId**
- Parameter: **AddressId** (an existing address ID)
- HTTP method: **GET**
- Data type: **JSON**

The request URL should be something like:

```
$ curl https://r3.minicrm.hu/Api/R3/Address/AddressId
```

The system will respond with the address details:

```
curl -s --user $SystemId:$ApiKey https://r3.minicrm.hu/Api/R3/Address/307
{
    "Id": 307,
    "ContactId": 37979,
    "Type": "",
    "Name": "",
    "CountryId": "Romania",
    "PostalCode": "",
    "City": "Test",
    "County": "Test",
    "Address": "John Doe Street, no. 1",
    "Default": 1,
    "CreatedBy": "John Doe",
    "CreatedAt": "2023-09-29 08:35:46",
    "UpdatedBy": "John Doe",
    "UpdatedAt": "2023-09-29 08:35:46"
}
```

Update an address

- API endpoint: **https://r3.minicrm.hu/Api/R3/Address/AddressId**
- Parameter: **AddressId** (an existing address identifier)
- HTTP method: **PUT**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -XPUT "https://r3.minicrm.hu/Api/R3/Address/AddressId"
```

This will update the address:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Address/307"
{
  "Address": "Test Street, no. 1"
}
```

In case of success, the system will respond with the updated address ID:

```
{
  "Id":12345
}
```

Card endpoint

Using this endpoint you will be able to get cards details (from opportunity cards, contact and/or companies cards) and also you will be able to update information on those cards.

Schema of the card

The available fields and their possible values in case of a given system can be queried via API via the Schema endpoint.

Example:

- API endpoint: **https://r3.minicrm.hu/Api/R3/Schema/Type**
- Parameter: In place of *Type* come Business or Person or Project/$CategoryId
- HTTP method: **GET**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Schema/Project/20"
```

```
{
    "AutoSalesV3_Qualification": {
        "421": "Unqualified",
        "422": "Qualified",
        "423": "VIP"
    },
    "AutoSalesV3_SalesStatus": {
        "425": "Discovery",
        "426": "In progress",
        "427": "Won opportunity",
        "428": "Lost opportunity"
    },
    "Name": "Text(512)",
    "StatusUpdatedAt": "DateTime",
    "IsPrivate": "Boolean",
    "Invited": "Text(65535)",
    "Deleted": "Int",
    "UpdatedAt": "DateTime",
    "AutoSalesV3_LeadStep": {
        "2730": "Offer",
        "2731": "Follow-up offer",
        "2733": "Contracting"
    },
    "AutoSalesV3_CustomerStep": {
        "2737": "Implementation",
        "2738": "Closed succesfully",
```

```
        "2739": "Feedback"
    },
    "Enum1221": {
        "2673": "Facebook",
        "2675": "SEO",
        "2677": "Google Ads",
        "2774": "LinkedIn"
    }
}
```

In the example above, we put only a few values, but the request will result in all values of the specific module.

Create a new card

If you need to create a card with full new data, which does not exist into your system (data for company, contact, and opportunity card), you will have to perform 3 API requests:

- 1st, for creating the company card;
- 2nd, for creating the contact card and associate it with the previously created company Id;
- 3rd, for creating the opportunity card and associate it with the previously created contact.

Create a new a company

If the company search doesn't produce results, and you would like to add a new company to MiniCRM, you can do it this way.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Contact/**
- HTTP method: **PUT**
- Data input: **JSON**

You should fill in the *Type* (with the value ***Business***) and the *Name* parameters.

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Contact/"-d
{
    "Type": "Business",
    "Name": "MiniCRM Ltd."
}
```

```
{
    "Id": 256
}
```

Create a new contact person

If a new company is added to the system, you can add a new contact person to it on this endpoint as well.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Contact/**
- HTTP method: **PUT**
- Data input: **JSON**

You should fill in at least the following data:

- **Type** (with the values ***Person***);
- **FirstName** (at least the first name is required in MiniCRM);
- BusinessId (optional) - you should fill this information if you want to connect the contact with an existing company.

Request

```
$ curl -s --user "https://r3.minicrm.hu/Api/R3/Contact/" -d
{
    "Type": "Person",
    "FirstName": "John",
    "LastName": "Doe",
    "Email": "john.doe@example.com",
    "BusinessId": 256
}
```

In case of success, the system will reply with the newly created contact Id:

```
{
    "Id": 256
}
```

Search for cards

Detailed data of a card

If you have the card's identifier – or API URL—here you can query detailed data. In the examples above, I got the "Id": 160 response during card search, therefore, I am going to use this.

In the ProjectEmail field, we store the unique e-mail of the Project. After you send a mail to this e-mail address, it will be archived to that card.

A ProjectHash field stores the Project's unique, hashed ID.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Project/CardId**
- Parameter: **CardId** (opportunity card Id)
- HTTP method: **GET**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Project/160"
```


```
{
  Id: 160,
  CategoryId: 20,
  ContactId: 155,
  UserId: "John Doe",
  Name: "Doe Industries LTD (John Doe)",
  StatusUpdatedAt: "2023-03-23 12:56:51",
  IsPrivate: 0,
  Deleted: 0,
  CreatedBy: "John Doe",
  CreatedAt: "2023-03-17 10:40:08",
  UpdatedBy: "John Doe",
  UpdatedAt: "2023-03-23 13:11:31",
  AutoSalesV3_Qualification: "Qualified",
  AutoSalesV3_IsHot: "Yes",
  AutoSalesV3_SalesStatus: "Won opportunity",
  AutoSalesV3_LeadStep: "Follow-up offer",
  AutoSalesV3_CustomerStep: "Closed successfully",
  ClosingDate: "2023-03-22 23:59:59",
  AttachedOffer: "https://r3.minicrm.io/64421/Download/S3/?t=project&inline=",
  OfferDate: "2023-03-21 23:59:59",
  SaleValue: 1500,
  WhyLost: "",
  DetailsWhyLost: "",
  BusinessId: 154,
  ProjectHash: "1bth0t9s2l2bclk3d01q0c7ci4c5ol",
  ProjectEmail: "projectemail@domain.com",
}
```

A company's cards from one module

Many times it is necessary to have a company's cards that are in a specific module. You can find the cards by using the BusinessId field's value of the contact response. In my system used as an example, I got the "BusinessId": 154 response for myself, therefore, I am going to use this.

The module identifier's – CategoryId – value can be figured out by using the methods mentioned in the beginning of the document.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Project?MainContactId=MainContactId&CategoryId=CategoryId**
- Parameter: **MainContactId** (opportunity card main contact Id - please read more here), **CategoryId** (product Id for the product within you want to search the cards)
- HTTP method: **GET**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Project?MainContactId=154&
CategoryId=20"
```


```
{
    "Count": 1,
    "Results": {
        "160": {
            "Id": 160,
            "Name": "Doe Industries LTD (John Doe)",
            "Url": "https://r3.minicrm.hu/Api/R3/Project/160",
            "ContactId": 155,
            "StatusId": 2813,
            "UserId": 115950,
            "Deleted": 0,
            "BusinessId": 154
        }
    }
}
```


If you test it with curl, put the URL between " signs, otherwise, the part that comes after & gets 'lost'.

Search based on card status

It is possible to combine the criteria used for narrowing the search, for example, a specific client's cards from a specific status:

- API endpoint: **https://r3.minicrm.hu/Api/R3/Project?MainContactId=$MainContactId&StatusId=$StatusId**
- Parameter: **MainContactId, StatusId**
- HTTP method: **GET**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Project?MainContactId=154&StatusId=2810"
```

Response:


```
{
    "Count": 1,
    "Results": {
        "160": {
            "Id": 160,
            "Name": "Doe Industries LTD (John Doe)",
            "Url": "https://r3.minicrm.hu/Api/R3/Project/160",
            "ContactId": 155,
            "StatusId": 2810,
            "UserId": 115950,
            "Deleted": 0,
            "BusinessId": 154
        }
    }
}
```

Search based on the update time

If you want to list in the system the opportunity card based on the date and time when it was updated, you can do it by products.

If you would like to list the opportunity card which are located in specific product and were updated at a specific time, you can use the following request:

For this, you can use the UpdatedSince and UpdatedAt parameters.

> When you give the date and time, you can increase or decrease the strictness of the specific condition.

General time and date format

Exact date (year, month, date) and time (hour, minutes, seconds)

```
UpdatedAt/Since=yyyy-mm-dd hh:mm:ss
```

Exact date (year, month, date) without time:

```
UpdatedAt/Since=yyyy-mm-dd
```

> The minimum information needed for the request are: ***year***, ***month*** and ***day***

1. UpdatedSince parameter

- API endpoint: **https://r3.minicrm.hu/Api/R3/Project?CategoryId=$CategoryId&UpdatedSince=$UpdatedSince**
- Parameter: **CategoryId** (product Id), **UpdatedSince** (searched date in format yyyy-mm-dd hh:mm:ss)
- HTTP method: **GET**
- Data type: **JSON**

With this method, you can list the cards that were updated after a specific date and time.

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Project?CategoryId=$Id&UpdatedSince=yyyy-mm-dd
hh:mm:ss"
```

Response:

```
{
    "Count": 1,
    "Results": {
        "XXXX": {
            "Id": XXXX,
            "Name": "Project Name 1",
            "Url": "https://r3.minicrm.hu/Api/R3/Project/XXXX",
            "ContactId": XXXX,
            "StatusId": XXXX,
            "UserId": XXXX,
            "Deleted": 0
        }
    }
}
```

2. UpdatedAt parameter

With this method, you can list the cards that were updated at a specific date and time.

- API endpoint: **https:/r3.minicrm.hu/Api/R3/Project?CategoryId=$CategoryId&UpdatedAt=$UpdatedAt**
- Parameter: **CategoryId** (product Id), **UpdatedAt** (searched date in format yyyy-mm-dd hh:mm:ss)
- HTTP method: **GET**
- Data type: **JSON**

Example:

```
$ curl -s --user $SystemId:$ApiKey
"https:/r3.minicrm.hu/Api/R3/Project?CategoryId=$Id&UpdatedAt=yyyy-mm-dd hh:mm:ss"
```

Response:


```
{
    "Count": 1,
    "Results": {
        "XXXX": {
            "Id": XXXX,
            "Name": "Project Name 2",
            "Url": "https://r3.minicrm.hu/Api/R3/Project/XXXX",
            "ContactId": XXXX,
            "StatusId": XXXX,
            "UserId": XXXX,
            "Deleted": 0
        }
    }
}
```

Search based on fields

Also, it is possible to search based on a specific field value or to combine different values, as presented earlier.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Project?FieldName=SomeValue**
- Parameter: **FieldName, SomeValue** (searched value)
- HTTP method: **GET**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Project?Enum1221=Facebook"
```

```
{
  "Count": 1,
  "Results": {
    "260": {
      "Id": 260,
      "Name": "John Doe",
      "Url": "https://r3.minicrm.hu/Api/R3/Project/260",
      "ContactId": 253,
      "StatusId": 2806,
      "UserId": 115950,
      "Deleted": 0,
    },
  },
}
```

Search for cards Extra

You can search for the list of cards that have specific data saved in them.

For example if you want to list the cards which are in the Sales product, the expected revenue is 20000 EUR

Endpoint:
https://r3.minicrm.hu/Api/R3/Project?CategoryId=$CategoryId&$Customfield=$Value

Endpoint: **https://r3.minicrm.hu/Api/R3/Project?CategoryId=154&RevenueEur=20000**

Search the card status history

It is possible to query a card's status change history via API.

- API endpoint: **https://r3.minicrm.hu/Api/R3/ProjectHistory/$ProjectId/?Type=StatusHistory**
- Parameter: **ProjectId** (card Id)
- HTTP method: **GET**
- Data type: **JSON**

A card's current status is the Status_New_Id variable's value and the old one is Status_Old_Id. Status_Old_Name and Status_New_Name are the status names in the system that belongs to the identifiers above.

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/ProjectHistory/29592?Type=StatusHistory"
```

Response:


```
{
    "Count": "6",
    "Results": {
        "567": {
            "Id": "567",
            "ProjectId": "160",
            "Type": "StatusHistory",
            "UserId": "0",
            "ClosedAt": "2023-03-23 13:31:20",
            "Status_Old_Id": "2810",
            "Status_New_Id": "2811",
            "Status_Old_Name": "Qualified",
            "Status_New_Name": "VIP"
        },
        "353": {
            "Id": "353",
            "ProjectId": "160",
            "Type": "StatusHistory",
            "UserId": "115950",
            "ClosedAt": "2023-03-17 10:40:22",
            "Status_Old_Id": "2801",
            "Status_New_Id": "2806",
            "Status_Old_Name": "Qualify",
            "Status_New_Name": "Follow-up"
        },
        "352": {
            "Id": "352",
            "ProjectId": "160",
            "Type": "StatusHistory",
            "UserId": "115950",
```

```
            "ClosedAt": "2023-03-17 10:40:08",
            "Status_Old_Id": "",
            "Status_New_Id": "2801",
            "Status_Old_Name": "",
            "Status_New_Name": "Qualify"
        }
    }
}
```

API request to get the contents of a filter

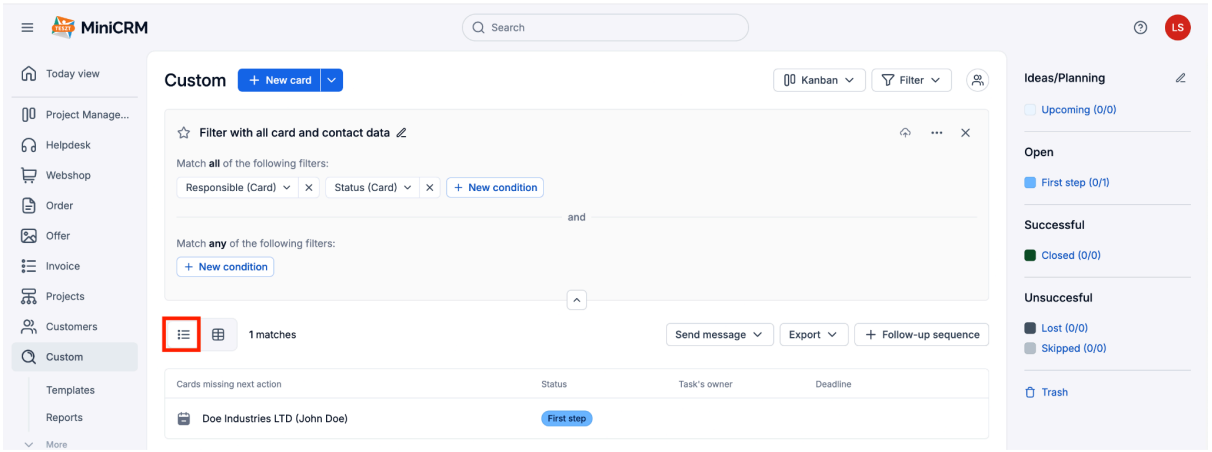
When creating a filter in the system, you have the possibility to save in Excel/synchronize the results.

You have the possibility to synchronize the results with either Google sheet or create a general HTTP request to get the same data.

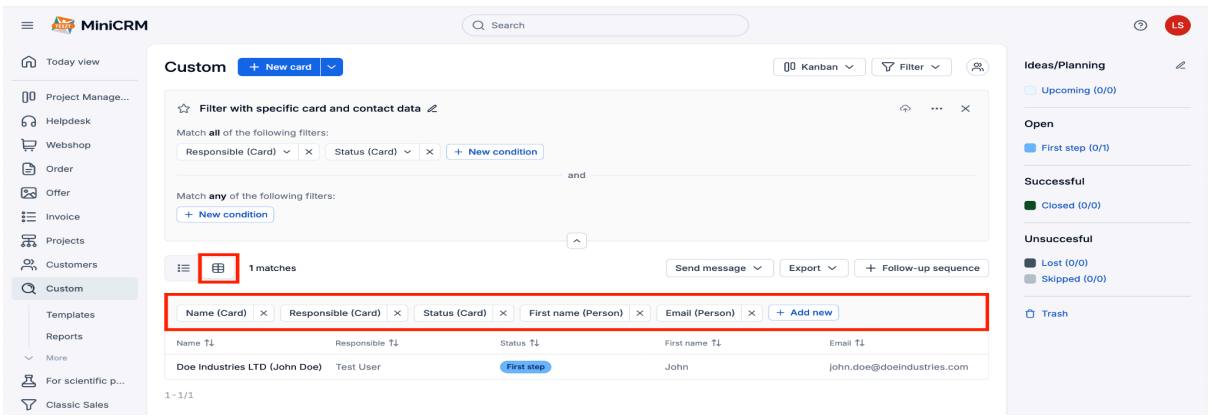
To use this method, you need the Google Sheet sync paid add-on, which is available in Settings → Subscription page.

Two ways to synchronize the data:

- When creating the filter in the system, you can synchronize **all the data** from the opportunity card and the contact card by creating the filter in **list view**:



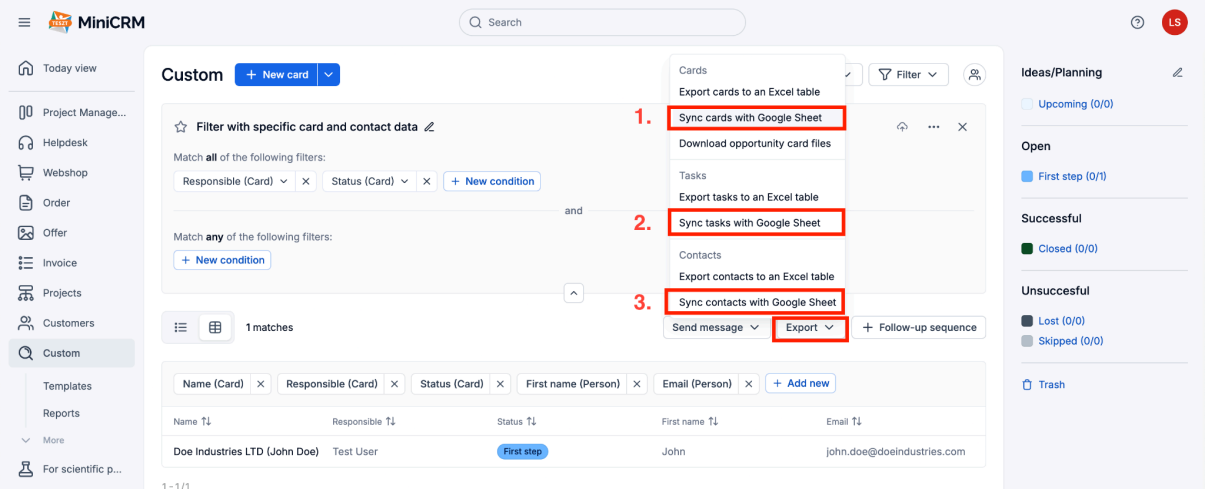
- The other option is to see and synchronize specific data from the opportunity and contact cards by **switching** to **table view** and **adding the specific field:**



Creating the request:

For this method, you don't need to use your system ID or API key.

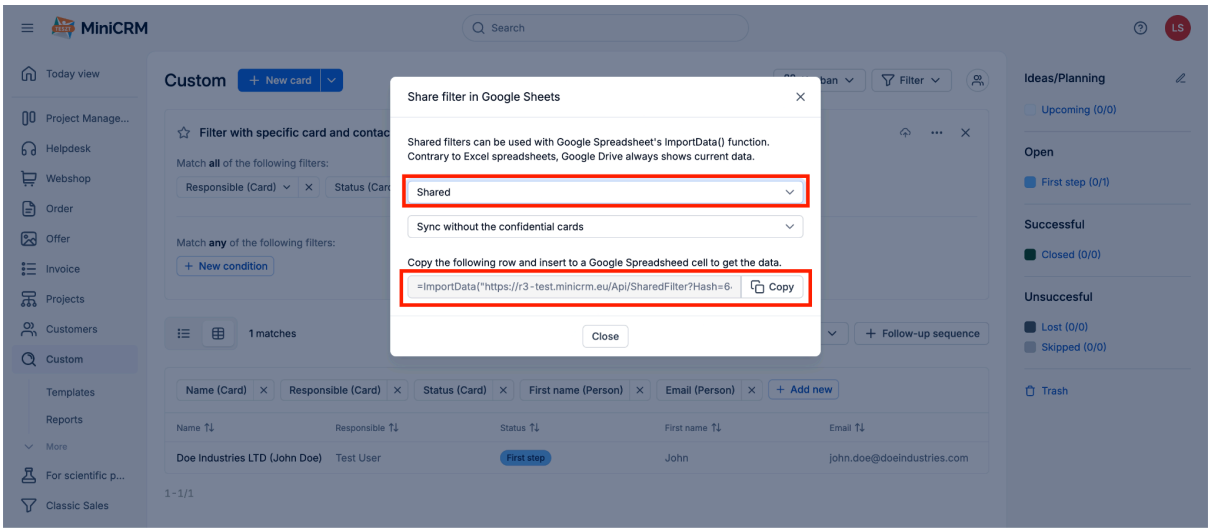
- Method: GET
- Endpoint: You can get it by selecting the synchronization with Google Sheet option in the Export option.



As demonstrated by the picture above, you can synchronize data from :

- the opportunity card
- Contact person/company card
- tasks/to-dos

Here you need to copy the sharing link, and use the main endpoint part from it:



You only need to use the red part from the created function:

=ImportData("https://r3-test.minicrm.eu/Api/SharedFilter?Hash=64421-0e01938fc48a2cfb5f2217fbfb00722d-EO9tXmhLWydgUmz_GTsTxQ&Type=Project")

Example response:

"Card: Id","Card: Name","Card: Responsible","Card: Status","Person1: First name","Person1: Email"
0qb2403d0s0ejgpmds630tlse4wyg5,"Doe Industries LTD (John Doe)","Test User","First step",John,john.doe@doeindustries.com

This method has one big advantage: you can get more and detailed data in 1 API request, compared to creating multiple queries to get the same result.

All cards of a company

Many times it is necessary to have all cards that belong to a company. You can find the cards by using the "BusinessId" field's value of the contact response.

In my system used as an example, I got the "'BusinessId": 154 response for myself, therefore, I am going to use this.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Project?MainContactId=$MainContactId**
- Parameter: **MainContactId**
- HTTP method: **GET**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Project?MainContactId=154"
```


```
{
   "Count": 3,
   "Results": {
       "160": {
           "Id": 160,
           "Name": "Doe Industries LTD (John Doe)",
           "Url": "https://r3.minicrm.hu/Api/R3/Project/160",
           "ContactId": 155,
           "StatusId": 2807,
           "UserId": 115950,
           "Deleted": 0,
           "BusinessId": 154
       },
       "162": {
           "Id": 162,
           "Name": "Mass email sending",
           "Url": "https://r3.minicrm.hu/Api/R3/Project/162",
           "ContactId": 155,
           "StatusId": 2573,
           "UserId": 115950,
           "Deleted": 0,
           "BusinessId": 154
       }
   }
}
```

Update a card

> It is recommended to send only those fields that you would like to modify.

If you have the card's identifier – or API Url – you can also modify data through this.

In the examples above, I got the "Id": 160 response during card search, therefore, I am going to use this here as well.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Project/$CardId**
- Parameter: **CardId**
- HTTP method: **PUT**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey -XPUT "https://r3.minicrm.hu/Api/R3/Project/160"
-d
{
    "AutoSalesV3_Qualification": "VIP"
}
```

```
{
    "Id": 256
}
```

In case of dropdown fields such as SatusId and UserId, you can send the value by saving as you can see on the user interface, or you can use its unique identifier as well that is stored on the system (which one is easier for you).

Following a successful saving, the response is a JSON object that contains the opportunity card's identifier in Id property. If you would like to add a new card, you also have to call the endpoint above by HTTP PUT method and omit the Id.

Among the data that is sent in, the required fields have to be filled out. It depends on the system, module, and status which field is required. You can check it on the MiniCRM user interface if you add a new card manually to the selected module's selected status.

Opportunity card owner/responsible:

When creating or modifying an opportunity card with a JSON containing the responsible user`s name it will work the following way:

You can use either the character-accurate name or id of the user who will be the responsible/owner of a specific opportunity card when creating or updating it.

Example JSON:

Username version:

```
{
    "Id": $ProjectId,
    "CategoryId": $CategoryId,
    "ContactId": $ContactId,
    "StatusId": "Processing",
    "UserId": "Specific User Name",
    "Name": "Test Opportunity card",
....
```

UserId version:

```
{
    "Id": $ProjectId,
    "CategoryId": $CategoryId,
    "ContactId": $ContactId,
    "StatusId": "Processing",
    "UserId": 108637",
    "Name": "Test Opportunity card",
....
```

If the user field`s value contains a typo and you don`t have an active user in your system with that name or id, the request will be sent in successfully and you will get back the following response “Ok” with code 200. In this case the system will randomly assign that opportunity card to a specific user.

In case of users who don`t have access to the specific product (Category) for example Sales, but the sent in array contains a valid username or Id, the system will automatically set the responsible to a user who has access to that product.

Operating with this API logic will result in less errors regarding to requests and more efficient data processing since the number of missing opportunity cards will be minimal, instead of giving back an error message and not creating a card because of typos.

Moving an opportunity card to another contact connected to the same business

- Endpoint: **https://r3.minicrm.hu/Api/R3/Project/$ProjectId**
- Method: **PUT**
- Example **JSON**

Original

```
{
"ContactId": 6505
}
```

Changed

```
{
"ContactId": 6623
}
```

Changing the main contact of the card which is connected to another business

For this change you have to send in the new main contact of the card specific card you want to move and also you have to send in the id of the business to which the contact is connected.

- Endpoint: https://r3.minicrm.hu/Api/R3/Project/$ProjectId
- Method: PUT

Example JSON:

Original

```
{
"ContactId": 6505,
"BusinessId": 6504
}
```

Changed

```
{
"ContactId": 1622,
"BusinessId": 1235
}
```

Limitation:
In the systems current version you cannot change the contact person to another contact person which is not connected to a business entity.

File uploading to a card

With the card Id or with the API URL, you are able to upload a file to a card, which will show in the history, and you can download it with a single click. To achieve this, you need to store the file on a public server (you can use HTTP authentication), where MiniCRM can reach and download that file.

You will also need to place a File column on your card.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Project/$CardId**
- Parameter: **CardId**
- HTTP method: **PUT**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey -XPUT "https://r3.minicrm.hu/Api/R3/Project/160"-d
{
    "$FileColumnName": "$FileUrl"
}
```

```
{
    "Id": 160
}
```

After the successful saving, the response will be the “Id” data name, and the Id value of the card.

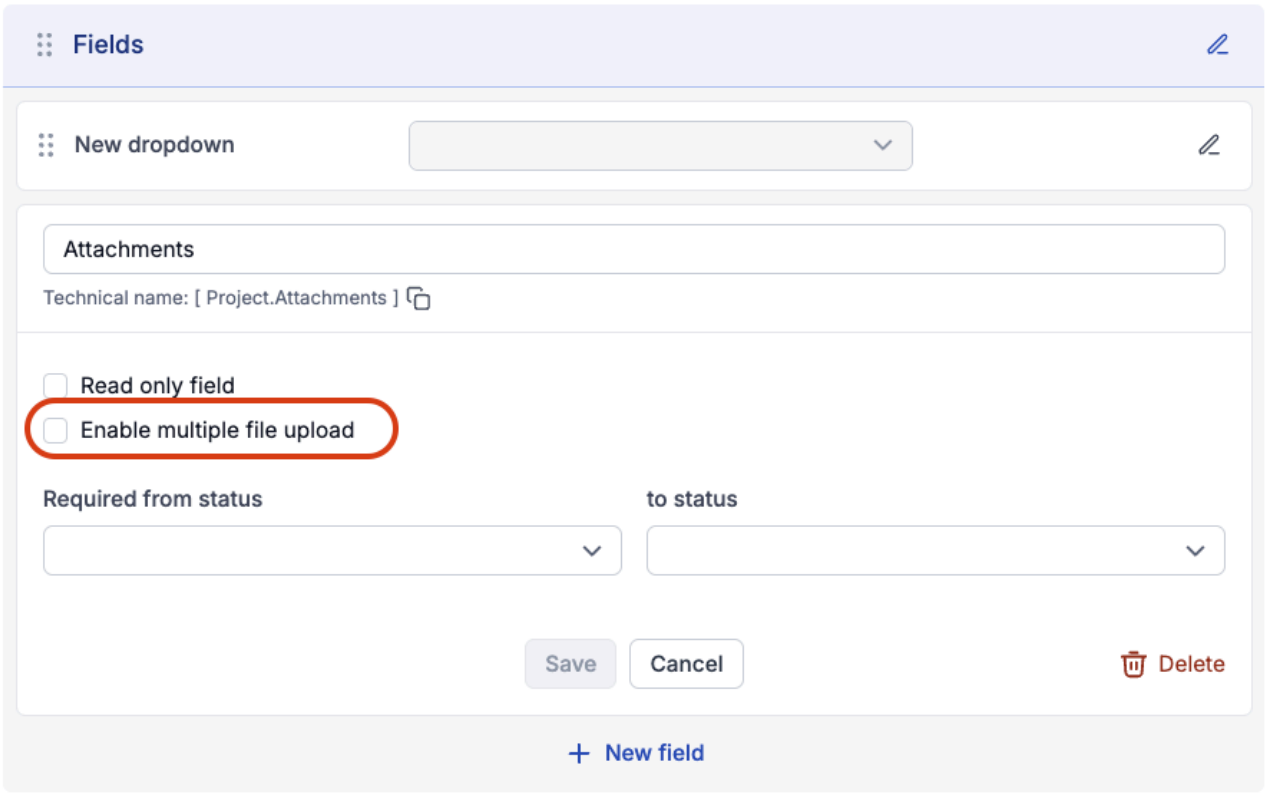
Multiple file upload to a card

When it comes to file uploading, you can do it two ways from the system and with API as well.

On the opportunity cards, you have to add a file filed to the existing fields in your product. There, you have the ability to change the settings of how file fields work.

First, you can use the fields to hold only 1 file, and when another file is uploaded the system will overwrite the old one.

Second, you can use the file fields to hold multiple files at once. If you want to do it, please check the box within the field settings:



Multiple file upload

- API endpoint: **https://r3.minicrm.hu/Api/R3/Project/$CardId**
- Parameter: **CardId**
- HTTP method: **PUT**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey -XPUT "https://r3.minicrm.hu/Api/R3/Project/160"-d
{
    "$FileFieldName": "$FileUrl1,$FileUrl2"
}
```

Example request:

```
$ curl -s --user $SystemId:$ApiKey -XPUT "https://r3.minicrm.hu/Api/R3/Project/160"-d
{
"FileField":"https://file-examples.com/wp-content/storage/2017/02/file_example_XML_24
kb.xml,https://file-examples.com/wp-content/storage/2017/02/zip_2MB.zip"
}
```

Example answer:

```
{
   "Id": 6356
}
```

Deleting a card

You have the ability to put the opportunity cards into Trash via API using the following method:

- Endpoint: **https://r3.minicrm.hu/Api/R3/Project/$ProjectId**
- Method: **PUT**

Example JSON:

```
{
    "Deleted": 1
}
```

Important: With this method you cannot permanently delete data from your system, in order to do that you have to log into the system and do one of the following actions:

- Open the trash in one of your products
- Open the opportunity card you want to delete
- And in the upper right corner click on the red “Delete permanently” button and here you can permanently delete data with the following methods:
    - you can delete that specific opportunity card and nothing else
    - you can delete the contact and all the cards connected to it

Side note: In item based products like Offer, you cannot permanently delete any card data.

Creating a query to deleted cards

You have the ability to get the list of the cards which have been deleted before in the specific product you want.

You cannot create a request to get all the deleted cards from your whole system at once, you need more than one request to do that.

- Endpoint: **https://r3.minicrm.hu/Api/R3/Project?CategoryId=$CategoryId&Deleted=1**
- Method: **GET**

Example response JSON:


```
{
    "Count": 2,
    "Results": {
        "8185": {
            "Id": 8185,
            "Name": "Test Contact Project 1",
            "Url": "https://r3.minicrm.hu/Api/R3/Project/8185",
            "ContactId": 6495,
            "StatusId": 5227,
            "UserId": 108637,
            "Deleted": 1
        },
        "8186": {
            "Id": 8186,
            "Name": "Test Contact 2 Project 2",
            "Url": "https://r3.minicrm.hu/Api/R3/Project/8186",
            "ContactId": 6496,
            "StatusId": 5227,
            "UserId": 108637,
            "Deleted": 1
        }
    }
}
```


Here the “Count” indicates the number of deleted cards in the specific product which ID was used in the endpoint.

You have the ability to get the details of the specific opportunity card and the todo list on the Project endpoint using the ID which you got in the response for your first query.

You cannot check the status history of the deleted card, you can only do this with active cards.

Modifying deleted cards

When it comes to opportunity cards, even if they are in the Trash, you have the ability to change the card's status, fields or even create or modify a to-do on it.

Regarding to-dos, it's important information that if you create a to-do on a deleted card, it won't show up in the Today's view screen of that specific user.

If you want:
For modifying card data

- Endpoint: **https://r3.minicrm.hu/Api/R3/Project/$ProjectId**
- Method: **PUT**

For creating or modifying to-dos on the card

- Endpoint: **https://r3.minicrm.hu/Api/R3/Todo/**
  OR
- Endpoint: **https://r3.minicrm.hu/Api/R3/Todo/$ToDoId**
- Method: **PUT**

Restoring a card

For this you can use the general Project endpoint and to restore a specific card you just have to change the “Deleted” row's value:

- Endpoint: **https://r3.minicrm.hu/Api/R3/Project/$ProjectId**
- Method: **PUT**

Example JSON:

```
{
    "Deleted": 0
}
```

Delete a contact

A Contact can be deleted when it is not the Primary Contact of any Card.

- API endpoint: **https://r3.minicrm.hu/Api/R3/PurgePerson/$ContactId**
- Parameter: **ContactId**
- HTTP method: **GET**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/PurgePerson/257"
```

Response:

```
{
    "Message": "Successfully deleted"
}
```

If a contact is the primary contact on one (or more) card(s), you need to set a new primary contact on that card by sending a new *ContactId* to that *Business*.

Error handling

If the data that should be sent in is not correct, the server will return the *"HTTP 400 Bad Request"* response.

When you are trying out API calls in Terminal, it is not shown by default what code (200 or something else) was returned by MiniCRM. By using the -w parameter, it is possible to affix the return code to any command.

In case of an error, the response is not encoded in JSON, but it is a plain text error message.

If a similar case occurs that you can see below, you can figure out the possible values via the Schema endpoint mentioned above or on the MiniCRM user interface.

```
$ curl -s --user $SystemId:$ApiKey -w "\nHTTP CODE: %{http_code}\n" -XPUT
"https://r3.minicrm.hu/Api/R3/Project/2056" -d
{
    "AutoSalesV3_Qualification": "Such value cannot be found in the CRM"
}
```

```
Project (PUT): Invalid value 'Such value cannot be found in CRM' for the Lead
qualification field. HTTP CODE: 400
```

Card connection endpoint

Using this endpoint, the user will be able to create new card connections and to manage the existing ones in MiniCRM. Also, this can be used to list card connections.

Create a new card connection

To be able to create a new card connection, it will need:
- API endpoint: **https://r3.minicrm.hu/Api/R3/CreateProjectConnection**
- Parameter: **FromProjectId** (card's identifier) and **ToProjectId** (card's identifier)
- HTTP method: **PUT**
- Data type: **JSON**

To create a new card connection, the authentication will be required as in the following example:

```
$ curl -s --user $SystemId:$ApiKey -X PUT
"https://r3.minicrm.hu/Api/R3/CreateProjectConnection/" -d
'{"FromProjectId":$FromProjectId,"ToProjectId":$ToProjectId}
```

The system will respond with the connection details:

```
{
    "ProjectId": 22222,
    "ProjectName": ProjectName,
    "Description": ,
    "CategoryId": 1,
    "StatusId": 33333,
    "MainContactId": 1
}
```

List card connection

To be able to list existing card connection, it will need:
- API endpoint: **https://r3.minicrm.hu/Api/R3/ListProjectConnection**
- Parameter: **FromProjectId** (card's identifier)
- HTTP method: **GET**
- Data type: **JSON**

To list an existing card connection, the authentication will be required as in the following example:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/ListProjectConnection?FromProjectId=$FromProjectId"
```

The system will respond with the connection details:

```
{
    "TotalCount": 1,
    "Page": 0,
    "ItemPerPage": 100,
    "Connections": {
            "0": {
                    "ProjectId": 22222,
                    "ProjectName": ProjectName,
                    "Description": ,
                    "CategoryId": 1,
                    "StatusId": 33333,
                    "MainContactId": 1
                }
        }
}
```

Update card connection

To be able to update existing card connection, it will need:
- API endpoint: **https://r3.minicrm.hu/Api/R3/UpdateProjectConnection**
- Parameter: **FromProjectId** (card's identifier), **ToProjectId** (card's identifier) and **Description**
- HTTP method: **PUT**
- Data type: **JSON**

To update an existing card connection, the authentication will be required, as in the following example:

```
$ curl -s --user $SystemId:$ApiKey -X PUT
"https://r3.minicrm.hu/Api/R3/UpdateProjectConnection/" -d
'{"FromProjectId":$FromProjectId,"ToProjectId":$ToProjectId,"Description":$Descriptio
n}'
```

Delete card connection

To be able to delete existing card connection, it will need:
- API endpoint: **https://r3.minicrm.hu/Api/R3/DeleteProjectConnection**
- Parameter: **FromProjectId** (card's identifier) and **ToProjectId** (card's identifier)
- HTTP method: **PUT**
- Data type: **JSON**

To delete an existing card connection, the authentication will be required, as in the following example:

```
$ curl -s --user $SystemId:$ApiKey -X PUT
"https://r3.minicrm.hu/Api/R3/DeleteProjectConnection/" -d
'{"FromProjectId":$FromProjectId,"ToProjectId":$ToProjectId}'
```

To-do endpoint

Using this endpoint, you will be able to manage your tasks (to-dos) - create new tasks, search, update or close existing tasks.

Create a new to-do

The *ProjectId* field is mandatory to be able to create a new to-do!

- API endpoint: **https://r3.minicrm.hu/Api/R3/ToDo/**
- HTTP method: **POST**
- Data type: **JSON**

Example Url:

```
$ curl -s --user $SystemId:$ApiKey -XPUT "https://r3.minicrm.hu/Api/R3/ToDo/"
```

```
{
  "UserId":"MiniCRM Helpdesk",
  "ProjectId":"1331",
  "Comment":"This To-Do was created via API.",
  "Deadline":"2023-02-23 15:35:40",
  "Duration": 60,
  "Reminder": 60,
  "AddressId": 4558,
  "Status":"Open"
}
```

After the successful sending, you will get an Id in response, which represents the to-do Id.

Response:

```
{
  "Id":1111
}
```

It is essential to note here that a new task in the case of the ProjectId send field is obligatory, and an existing task and in the case of ProjectId field already cannot be edited!

Search for to-dos

- API endpoint: **https://r3.minicrm.hu/Api/R3/ToDoList/$CardId**
- Parameter: **CardId**
- HTTP method: **GET**
- Data type: **JSON**

You have to identify the Card, for example with that URL you can ask all the to-dos from a card:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/ToDoList/1234"
```

In the response, the result comes down to one block, where the Count shows the number of the found to-dos. Under *Results* you can find the to-dos in different blocks.

The listed to-dos can be filtered by their Status with the Status parameter, the to-do can be Open, Closed, and All. Example for to-do list in Open status:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/ToDoList/1234?Status=Open"
```

Response:


```
{
   "Count":2,
   "Results":[
      {
         "Id":"1111",
         "Status":"Open",
         "Comment":"Call the customer and ask for feedback.",
         "Deadline":"2023-03-20 23:59:00",
         "UserId":"3200",
         "Type":"0",
         "Url":"https://r3.minicrm.hu/Api/R3/ToDo/1111"
      },
      {
         "Id":"2222",
         "Status":"Open",
         "Comment":"Check if it needs further action.",
         "Deadline":"2013-01-22 23:59:00",
         "UserId":"3300",
         "Type":"1566",
         "Url":"https://r3.minicrm.hu/Api/R3/ToDo/2222"
      }
   ]
}
```

Detailed data of a to-do

- API endpoint: **https://r3.minicrm.hu/Api/R3/ToDo/$ToDoId**
- Parameter: **ToDoId**
- HTTP method: **GET**
- Data type: **JSON**

Identification required, example URL:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/ToDo/1111"
```

Querying the data of a selected task. In response, the to-do arrives in an array containing the to-do data.

Response:


```
{
   "Id": 3606,
   "UserId": "User A",
   "ContactId": 0,
   "ProjectId": 7888,
   "AddressId": 4558,
   "SenderUserId": 0,
   "Type": "Test type",
   "Duration": 60,
   "Reminder": 60,
   "Status": "Open",
   "Mode": "Task",
   "Deadline": "1970-01-01 01:00:00",
   "Comment": "This is a task created via API",
   "CreatedBy": "MiniCRM",
   "CreatedAt": "2024-08-19 13:54:44",
   "UpdatedBy": "MiniCRM",
   "UpdatedAt": "2024-08-19 13:56:22",
   "ClosedBy": "MiniCRM",
   "ClosedAt": "0000-00-00 00:00:00",
   "Attachments": {
       "FileName": "test contract.pdf",
       "Url": "https://r3.minicrm.hu/..."
   },
   "Members": [
       {
           "Entity": "User2",
           "EntityId": "109287"
       }
   ],
   "Notes": [
       {
           "Comment": "Test comment",
           "CreatedBy": "108637",
           "CreatedAt": "2024-08-19 13:55:00"
```

```
        }
    ]
}
```

Modifiable fields:

- **ContactId**: It is relevant if our task in the customer service product is a response email type task. In this case, *ContactId* is the ID of the recipient.
- **AddressId**: The system ID of the address specified in the task. This can be queried via the Address endpoint (check the address endpoint chapter)
- **Members**: The type and ID of the stakeholders hired for the task. It can be queried via the Contact endpoint. There can be multiple members in one task.
- **Comment**: The text of the comment.
- **Duration:** You can give the planned duration of the completion of the task. This data is measured in minutes.
- **Reminder:** If your user is connected to the Google Calendar/Outlook Calendar, you can get notifications. You can set the reminder you want to get X minutes before the due date.
- **Status**: This field indicates if the task is open or closed. You can close an active task simply by changing this field's value to "Closed".
- **Attachments:** By inserting an accessible URL that points to a download link to a file, it will be automatically uploaded to the task. If you want to upload multiple attachments, then you can separate them with comma without any space!

Read-only fields:

- **CreatedBy**: Created by (UserId). It can be queried via the Schema/Project endpoint (please verify that chapter).
- **CreatedAt**: Gives you the time of the creation of the task.
- **UpdateAt:** Gives you the time when the task was last modified.
- **UpdatedBy:** Which user made the latest change regarding the task.
- **ClosedAt:** When was the task closed.
- **ClosedBy:** Which user closed the task.
- **Mode**: Can be "Task" or "Email".
- **Notes**: The comments written for the task will all be listed in this array.

General:

- **Attachments:** Here you can get the latest uploaded file's exact name and format and the download link for that.

If you modify an open task with POST and the ClosedAt field with a given date and the array doesn't contain a Status field, the system will close this task.

Update a to-do:

Modify an existing to-do or create a new one. It is advisable to resend only the modified data so that the program can run more efficiently, and it is possible to avoid restoring data that has already been modified meanwhile to previously downloaded values.

```
$ curl -s --user $SystemId:$ApiKey -XPUT "https://r3.minicrm.hu/Api/R3/ToDo/1111" -d
'{
 "Comment":"New text sent in via API",
 "Deadline":"2023-03-25 13:30:00"
}'
```

You can only change the opened tasks. Closed tasks cannot be modified!

The URL of the service is the same as the URL of the download to be done. A download can be initiated with a GET request, and data modification can be initiated with a PUT request. Adding a new task is possible by omitting the identifier (for example, omitting 1111 at the end of the URL).

Response:

```
{
  "Id":1111
}
```

File upload

- API endpoint: **https://r3.minicrm.hu/Api/R3/ToDo/$ToDoId**
- Parameter: **ToDoId**
- HTTP method: **PUT**
- Data type: **JSON**

Single file upload

```
{
   "Attachments": "https://www.testsite1.com/files/test.pdf"
}
```

Multiple file upload

```
{
   "Attachments":
"https://www.testsite1.com/files/test.pdf,https://www.testsite2.com/documents/test_2.pdf"
}
```

You can upload one or more files at once to a task. If you upload any new file then it will be added to the already uploaded attachments, it won't be overwritten.

You cannot upload a file or files that combined have a larger size than 24 MB!

Custom close date and time

You can close the task and add a custom close date. If you close a task like this, it will appear in chronological order in the card history.

This is a great solution if you want to add or update the history of the opportunity card, which cannot be done with other methods.

Important to note that first, you have to create a task to get an ID, and then you can modify it using the example below.

This will only work if the task is in normal Closed or Failed status, it will not work when you close the task to Successful in the system.

Closed or Successful statuses will not appear in the JSON array you get back as a response, it will only be visible in the system, in the opportunity card history.

- API endpoint: **https://r3.minicrm.hu/Api/R3/ToDo/$ToDoId**
- Parameter: **ToDoId**
- HTTP method: **Post**
- Data type: **JSON**

Request example:

```
{
"Status":"Closed",
"ClosedAt": "2009-01-09"
}
```

Response:

```
{
  "Id":1111
}
```

Templates endpoint

The purpose of this endpoint is to offer the possibility to list all the templates from a specific product. A detailed view of a specific template is also available.

Get the template list for a specific product

- API endpoint: **https://r3.minicrm.hu/Api/R3/TemplateList/$CategoryId**
- Parameter: **CategoryId** (product Id as described here)
- HTTP method: **GET**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/TemplateList/$CategoryId"
```

The system will respond with the company/contact details:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/TemplateList/111"
{
    "Count": 5,
    "Results": [
        {
            "Id": 2647,
            "Type": "Email",
            "Name": "Template with customer name",
            "Url": "https://r3.minicrm.hu/Api/R3/Template/2647"
        },
        {
            "Id": 2646,
            "Type": "Email",
            "Name": "Custom template",
            "Url": "https://r3.minicrm.hu/Api/R3/Template/2646"
        },
        {
            "Id": 2583,
            "Type": "Todo",
            "Name": "Create a custom task",
            "Url": "https://r3.minicrm.hu/Api/R3/Template/2583"
        },
               {
            "Id": 2589,
            "Type": "Sms",
            "Name": "Test",
            "Url": "https://r3.minicrm.hu/Api/R3/Template/2589"
        },
```

```
        {
            "Id": 2591,
            "Type": "File",
            "Name": "Test_file",
            "Url": "https://r3.minicrm.hu/Api/R3/Template/2591"
        },
    ]
}
```

Search for templates

Template details

This endpoint displays specific template details.

- API endpoint: **https://r3.minicrm.hu/Api/R3/Template/$TemplateId**
- Parameter: **TemplateId**
- HTTP method: **GET**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey
"https://r3.minicrm.hu/Api/R3/Template/$TemplateId"
```

Example:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/R3/Template/2583"
```

The system will respond with the company/contact details:

```
{
    "Id": 2583,
    "CategoryId": 111,
    "Type": "To do",
    "Name": "Task template example",
    "Subject": "",
    "Content": "<!DOCTYPE html><html><head><title></title></head><body
data-gramm=\"false\" data-lt-tmp-id=\"lt-122144\" spellcheck=\"false\">\n\nThis is a
test</body></html>",
    "FolderId": 7621,
    "Folders": [
        7621
    ]
}
```

Order endpoint

This endpoint helps you to manage your orders that you manage in MiniCRM.

The listing of the orders and searching in the orders are not supported yet via API.

Creating a new order

To create an order, you need to POST a JSON-encoded request. CustomerId or ReferenceId is required in this request.

- API endpoint: **https://r3.minicrm.hu/Api/Order**
- HTTP method: **POST**
- Data type: **JSON**

The billing address will be used on orders. If there is no billing address, the contact's default address will be used instead.

Using the Customer array is optional, you can set the buyer's data here. If there is an address, it will be used, if not, it will be set as a new default billing address.

In this case, the *VAT number* and *bank account* number will be overwritten by API!

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Order/" -d
{
    "CustomerId": "261",
    "CurrencyCode": "HUF",
    "Customer":
    {
        "Name": "John Doe Ltd.",
        "Country": "36",
        "PostalCode": "0000",
        "City": "AnonymCity",
        "Address": "Anonym Street 1",
        "AccountNumber": "XXXXXXXXXXXXXXXXXXXX",
        "VatNumber": "XXXXXXXXX",
        "EUVatNumber": "XX00000000"
    },
    "Subject": "",
    "Items": [
        {
            "SKU": "",
            "Name": "Cable",
            "Description": "",
```

```
            "PriceNet": "500.00",
            "Quantity": "15.00",
            "Unit": "m",
            "VAT": "27"
        }
    ]
}
```

Authentication is a must, the example is the following: The response will be all the order's data and the ID of the order (Order Id). In the response we also set the calculated fields (such as AmountVat, Amount Net). The response is similar to the "Requesting an order" response's structure.

Search for orders

Detailed data of an order

- API endpoint: **https://r3.minicrm.hu/Api/Order/$OrderId**
- Parameter: **OrderId**
- HTTP method: **GET**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/Order/$OrderId"
```


```
{
    "Id": "83",
    "ProjectId": "272",
    "CustomerId": "261",
    "Number": "",
    "InvoiceMode": "Normal",
    "Customer":
    {
        "Name": "John Doe Ltd.",
        "CountryId": "1",
        "County": "",
        "PostalCode": "0000",
        "City": "Anonym City",
        "Address": "Anonym Street 1",
        "AccountNumber": "92826253-45043722-59650728",
        "VatNumber": "17823443-1-26",
        "EUVatNumber": "HU23982273",
        "RegistrationNumber": "",
        "GroupIdentificationNumber": ""
    },
    "VendorGroupIdentificationNumber": "",
    "Subject": "",
    "PaymentMethod": "Cash",
    "Performance": "2023-09-25",
    "CurrencyCode": "HUF",
    "AmountNet": "7500",
    "AmountVat": "2025",
    "Amount": "9525",
    "Accounting": "Normal",
    "OSS": "0",
    "Items": [
        {
            "Id": "73",
            "SKU": "",
            "EAN": "",
            "Name": "Cable",
            "Description": "",
```

```
            "PriceNet": "500.000000",
            "Quantity": "15.0000",
            "Unit": "m",
            "PriceNetTotal": "7500.0000",
            "VAT": "27%",
            "VATSubtype": "",
            "PriceTotal": "9525.00"
        }
    ],
    "Project":
    {
        "Name": "To be issued (John Doe Ltd.)"
    }
}
```

Order list endpoint

You can get a list of your created orders via API by using the request down below. The listing also returns 100 results. Pagination can be done using the Page parameter, see more details in the Pagination chapter.

- API endpoint: **https://r3.minicrm.hu/Api/Offer/List**
- HTTP method: **GET**
- Data type: **JSON**

```
https://r3.minicrm.hu/Api/Offer/List
```

Response example:


```
{
  "Count": 52,
  "Results": {
    "4": {
      "Id": "4",
      "ProjectId": "24",
      "StornedId": "",
      "Number": "Webshop #24",
      "Type": "Order",
      "Issued": "2021-12-11",
      "Performance": "2021-12-11",
      "Prompt": "2021-12-11",
      "Paid": "",
      "CurrencyCode": "HUF",
      "AmountNet": "766.0000",
      "AmountVat": "214.0000",
      "Amount": "980.0000",
      "Status": "Successful",
      "StatusGroup": "Success",
      "DocumentUrl": ""
    },
```

Update order's data

Modifying an order's data is possible only in "Draft" status. Posting the CustomerId or ReferenceId is a must! Editing the *CurrencyCode* variable is not allowed, You need to POST it while creating an order.

- API endpoint: **https://r3.minicrm.hu/Api/Order/$OrderId**
- Parameter: **OrderId**
- HTTP method: **POST**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Order/$OrderId" -d
{
    "CustomerId": "261",
    "Subject": "Call the customer when the order is ready to go"
}
```

In response, you will see the modified order with the new subject in it.

Adding an order item

- API endpoint: **https://r3.minicrm.hu/Api/Order/$OrderId**
- Parameter: **OrderId**
- HTTP method: **POST**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Order/83" -d
{
    "CustomerId": "261",
    "Items":
        [{
        "SKU": "",
        "Name":"Linksys IP phone",
        "Description": "",
        "PriceNet": "12000.00",
        "Quantity": "3.00",
        "Unit": "piece",
        "VAT": "27"
        }]
}
```

Editing an order item by including the ItemId, You can overwrite the given order item. You can change the item itself or change its quantity.

- API endpoint: **https://r3.minicrm.hu/Api/Order/$OrderId**
- Parameter: **OrderId**
- HTTP method: **POST**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Order/83" -d
{
    "CustomerId": "261",
    "Items":
[
    {
        "Id": 74,
        "SKU": "",
        "Name": "Cisco IP phone",
        "Description": "",
        "PriceNet": "15000.00",
        "Quantity": "2.00",
        "Unit": "piece",
        "VAT": "27"
    }
]}
```

In response You will get all the modified order's data, and calculated fields will update automatically (such as AmountVat, AmountNet)

A shortened example for the response:

```
"Items": [
        {
            "Id": "73",
            "SKU": "",
            "EAN": "",
            "Name": "Cable",
            "Description": "",
            "PriceNet": "500.000000",
            "Quantity": "15.0000",
            "Unit": "m",
            "PriceNetTotal": "7500.0000",
            "VAT": "27%",
            "VATSubtype": "",
            "PriceTotal": "9525.00"
        },
        {
            "Id": "74",
            "SKU": "",
            "EAN": "",
            "Name": "Cisco IP phone",
```

```
            "Description": "",
            "PriceNet": "15000.000000",
            "Quantity": "2.0000",
            "Unit": "piece",
            "PriceNetTotal": "30000.0000",
            "VAT": "27%",
            "VATSubtype": "",
            "PriceTotal": "38100.00"
        }
    ],
```

Editing custom order fields

If you created custom fields on the order's card, You can edit them, without any status restriction.

- API endpoint: **https://r3.minicrm.hu/Api/Order/$OrderId/Project**
- Parameter: **OrderId**
- HTTP method: **POST**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Order/83/Project" -d
'
{
   "Text1479": "Please deliver the order between 9am and 5pm"
}'
```

Editing the status of an order

Finalizing an order

Sets the order's status from "Draft" to "In progress". Editing on works, when the order is in "Draft" status!

- API endpoint: **https://r3.minicrm.hu/Api/Order/$OrderId/Finalize**
- Parameter: **OrderId**
- HTTP method: **POST**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Order/83/Finalize"
```

Editing an order

Sets the order's status from "In progress" to "Draft". Editing on works, when the order is in "Draft" status!

- API endpoint: **https://r3.minicrm.hu/Api/Order/$OrderId/Edit**
- Parameter: **OrderId**
- HTTP method: **POST**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Order/83/Edit"
```

Completing an order

Sets the order's status from "In progress" to "Completed".

- API endpoint: **https://r3.minicrm.hu/Api/Order/$OrderId/Complete**
- Parameter: **OrderId**
- HTTP method: **POST**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Order/83/Complete"
```

Setting an order to Successful

Sets the order's status from "Completed" or "Failed" to "Paid".

- API endpoint: **https://r3.minicrm.hu/Api/Order/$OrderId/Paid**
- Parameter: **OrderId**
- HTTP method: **POST**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Order/83/Paid"
```

Setting an order to Unsuccessful

Sets the order's status from "Completed" or "Unsuccessful" to "Failed".

- API endpoint: **https://r3.minicrm.hu/Api/Order/$OrderId/Storno**
- Parameter: **OrderId**
- HTTP method: **POST**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Order/83/Storno"
```

Status changes can only be made by using the method mentioned above.

For example, if you would like to set a "Paid" order to "In progress" the response from the API will be ***"405 Method not allowed"***

Deleting an order item

- API endpoint:
  https://r3.minicrm.hu/Api/Order/$OrderId/DeleteItem/$ItemId
- Parameter: **OrderId**, **ItemId**
- HTTP method: **GET**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/Order/83/DeleteItem/74"
```

In response You will get all the modified order's data, and calculated fields will update automatically (such as AmountVat, AmountNet)

Offer endpoint

On this endpoint, you can manage your offers in MiniCRM. You can create the API request to send the offer into the system in JSON-encoded format or in an XML file.

Via this endpoint, you will be able to:

- Create an offer in your system
- Change the status of the Offer
- Modify or add information to the card fields

Creating and issuing offers

The system can process similar structures to requests you get as a response in cases of offers. The function could be used by the POST request. The data can be processed by the system in JSON-encoded format. In case of offers, the CustomerId and ReferenceId are mandatory information to give.

- API endpoint: **https://r3.minicrm.hu/Api/Offer**
- HTTP method: **POST**
- Data type: **JSON**

For the customers the invoicing address will be used, if there is no such address then the default address will be used on the offers. The usage of Customer Block within the request is optional, here you can provide the details of the customer. If the address already exists in the contact information we would use that, if not then it will be saved as a new (default) invoicing address.

The VAT number and Bank account will be overwritten by the API! The offers will be created in the first, default status, which is “Preparing offer”.

Authentication is a must in the request! You must include the CustomerId or ReferenceId! Editing the CurrencyCode variable is not allowed, You need to POST it while creating an offer.

Here is an example request:

```
$ curl -s --user $SystemId:$ApiKey -XPOST "http://r3.minicrm.hu/Api/Offer/" -d '
{
    "CustomerId": "20404",
    "InvoicePadCounter": "00005",
    "Number": "2024-E/00001",
    "Type": "Offer",
    "Media": "PKI",
    "Language": "RO",
    "Customer": {
        "Name": "John Doe",
        "CountryId": "40",
        "Country": "Romania",
        "PostalCode": "000000",
        "City": "Test",
        "Address": "Test Address",
        "AccountNumber": "",
        "VatNumber": "",
        "EUVatNumber": "",
        "RegistrationNumber": "",
        "GroupIdentificationNumber": ""
    },
    "Vendor": {
        "Name": "John Doe",
        "CountryId": "40",
        "PostalCode": "000000",
        "City": "Test",
        "Address": "John Doe Street, no. 1",
        "AccountNumber": "",
        "IBAN": "",
        "SWIFT": "",
        "VatNumber": "XXXXXX",
        "EUVatNumber": "ROXXXXXX",
        "GroupIdentificationNumber": "",
        "RegistrationNumber": ""
    },
    "ExtraData": "",
    "Subject": "",
    "PaymentMethod": "WiredTransfer",
    "Issued": "2024-01-01",
    "Performance": "2024-01-08",
    "Prompt": "2024-01-08",
    "Paid": "",
    "CurrencyCode": "RON",
    "ConversionRate": "",
    "AmountNet": "23.00",
    "AmountVat": "0.00",
    "Amount": "23.00",
    "VATMode": "NormalVAT",
    "Accounting": "Normal",
```

```
   "DocumentId": "6",
   "DocumentUrl":
"https://r3-test.minicrm.eu/documents/XYZ/Download/S3/?t=doc&inline=0&e=XYZ",
   "Items": [
       {
           "SKU": "",
           "EAN": "",
           "Name": "Test",
           "Description": "",
           "PriceNet": "23.0000",
           "Quantity": "1.00",
           "Unit": "piece",
           "PriceNetTotal": "23",
           "VAT": "19",
           "VATSubtype": "",
           "PriceTotal": "23.00"
       }
   ],
   "Project": {
       "String1911": "Test"
   }
}'
```

The API response will encompass comprehensive order data, including the unique Order ID. This response is augmented with dynamically computed fields, such as AmountVat and Amount Net. The structure of this response aligns with that of the 'Requesting an Order' response.

Search for offers

Detailed data of an offer

- API endpoint: **https://r3.minicrm.hu/Api/Offer/OfferId**
- Parameter: **OfferId**
- HTTP method: **GET**
- Data type: **JSON**

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/Offer/932"
```


```
{
  "Id": "932",
  "InvoicePadCounter": "00008",
  "Number": "AJ2021-E/00008",
  "Type": "Offer",
  "InvoiceMode": "Normal",
  "Media": "PKI",
  "Language": "HU",
  "Customer": {
    "Name": "John Doe",
    "CountryId": "36",
    "County": "",
    "PostalCode": "XXXX",
    "City": "Budapest",
    "Address": "XXX",
    "AccountNumber": "",
    "VatNumber": "",
    "EUVatNumber": "",
    "RegistrationNumber": "",
    "GroupIdentificationNumber": ""
  },
  "Vendor": {
    "Name": "John Doe INC",
    "CountryId": "36",
    "PostalCode": "XXXX",
    "City": "XXX",
    "Address": "XXX",
    "AccountNumber": "XXXXXXXXXXXXXXXXXXXX",
    "IBAN": "XXXXXXXXXXXXXXXXXXXX",
    "SWIFT": "XXXXXXXX",
    "VatNumber": "XXXXXXXXX",
    "EUVatNumber": "XXXXXXXX",
    "GroupIdentificationNumber": "",
    "RegistrationNumber": ""
  },
  "ExtraData": "{\"VendorMiscellaneous\":\"XXXXXXXX\"}",
  "Subject": "",
  "PaymentMethod": "WiredTransfer",
```

```
  "Issued": "2021-10-15",
  "Performance": "2021-10-15",
  "Prompt": "2021-10-15",
  "Paid": "",
  "CurrencyCode": "HUF",
  "ConversionRate": "",
  "AmountNet": "23",
  "AmountVat": "0",
  "Amount": "23",
  "VATMode": "VATExempt",
  "Accounting": "Normal",
  "DocumentId": "819",
  "DocumentUrl":
"https://r3.minicrm.hu/41274/Download/S3/?t=doc&inline=0&e=XXXXXXXXXX",
  "Items": [
    {
      "Id": "1306",
      "SKU": "",
      "EAN": "",
      "Name": "testr",
      "Description": "",
      "PriceNet": "23.0000",
      "Quantity": "1.00",
      "Unit": "darab",
      "PriceNetTotal": "23",
      "PriceTotal": "23"
    }
  ],
  "Project": {
    "Enum1907": "",
    "Float1910": "",
    "Int1909": "",
    "Int1912": "",
    "String1911": "TEST",
    "String1913": "",
    "String1914": "",
    "String1915": "",
    "StatusId": "3426"
  }
}
```

Offer list endpoint

You can get a list of your created offers via API by using the request down below. The listing also returns 100 results. Pagination can be done using the Page parameter, see more details in the Pagination chapter.

- API endpoint: **https://r3.minicrm.hu/Api/Offer/List**
- HTTP method: **GET**
- Data type: **JSON**

```
https://r3.minicrm.hu/Api/Offer/List
```

Response example:


```
{
  "Count": 12,
  "Results": {
    "9": {
      "Id": "9",
      "ProjectId": "264",
      "StornedId": "",
      "Number": "AJ2022-E/00001",
      "Type": "Offer",
      "Issued": "2022-04-14",
      "Performance": "2022-02-03",
      "Prompt": "2022-02-03",
      "Paid": "",
      "CurrencyCode": "EUR",
      "AmountNet": "678.0000",
      "AmountVat": "183.0000",
      "Amount": "862.0000",
      "Status": "Accepted",
      "StatusGroup": "Success",
      "DocumentUrl": ""
},
```

Updating an Offer

Update Offer's opportunity card fields

- API endpoint: **https://r3.minicrm.hu/Api/Offer/$OfferId/Project**
- Parameter: **OfferId**
- HTTP method: **POST**
- Datatype: **JSON**

Example JSON::

```
$ curl -s --user $SystemId:$ApiKey -POST "https://r3.minicrm.hu/Api/Offer/10/Project"
-d
{
"StatusId": "Paid",
"String1616": "Test"
}'
```

Adding a new item

- API endpoint: **https://r3.minicrm.hu/Api/Offer/$OfferId**
- Parameter: **Offer**
- HTTP method: **POST**
- Datatype: **JSON**

Example JSON:

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Offer/42" -d
{
"CustomerId": $CustomerId,
"Items": [
        {
            "SKU": "",
            "EAN": "",
            "Name": "Linksys IP telefon MÓDOSÍTOTT",
            "Description": "",
            "PriceNet": "12000.000000",
            "Quantity": "3.0000",
            "Unit": "darab",
            "PriceNetTotal": "36000.0000",
            "VAT": "27%",
            "VATSubtype": "",
            "PriceTotal": "45720.00"
        }
]
}
```

Modifying an item

- API endpoint: **https://r3.minicrm.hu/Api/Offer/$OfferId**
- Parameter: **Offer**
- HTTP method: **POST**
- Datatype: **JSON**

Example JSON:

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Offer/42" -d
{
"CustomerId": $CustomerId,
"Items": [
        {
            "Id": "1778", //Id of an already added item in the offer
            "SKU": "",
            "EAN": "",
            "Name": "Linksys IP telefon Modified",
            "Description": "",
            "PriceNet": "12000.000000",
            "Quantity": "3.0000",
            "Unit": "darab",
            "PriceNetTotal": "36000.0000",
            "VAT": "27%",
            "VATSubtype": "",
            "PriceTotal": "45720.00"
        }
]
}
```

Deleting an item

- API endpoint: **https://r3.minicrm.hu/Api/Offer/$Offerid/DeleteItem/$ItemId**
- Parameters: **OfferId, ItemId**
- HTTP method: **POST**

```
$ curl -s --user $SystemId:$ApiKey -GET "https://r3.minicrm.hu/Api/Offer/42/DeleteItem/242" -d
```

Invoice endpoint

Using this endpoint, the user will be able to create new invoices and to manage the existing ones in MiniCRM. Also, this can be used to search invoices.

During this chapter, we will often use the variable $InvoiceId. This is the actual document ID, and it is different from the card ID displayed in the card URL.

Create a new invoice

To be able to create a new invoice, it will need:

- API endpoint: **https://r3.minicrm.hu/Api/Invoice**
- Parameter: **CustomerId** (contact's identifier) or **ReferenceId** (project external id)
- HTTP method: **POST**
- Data type: **JSON**

Creating a new invoice is only possible with an existing Contact or Business within your system.

These can have two different identifiers:

1. **CustomerId** - Is the Id which is given by the system after creating a Business or a Contact.
2. **ReferenceId** - This is an external identifier that is not created by MiniCRM you can set it for a Business or Contact, in this case when creating the contact with XML sync or later on via API.

For more information regarding the contact creation and address handling, please check the (Company/contact endpoint and Address endpoint chapters).

The billing address will be used on invoices and if there is no billing address, then the contact's default address will be used instead.

Using the customer data in an array is optional, you can set the buyer's data right into the request.

If there is an address, that address will be used, if not it will be set as a new default billing address.

In this case, the VAT number and the bank account number will be overwritten by the API.

To create a new invoice, the authentication will be required as in the following example:

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Invoice/" -d '
{
    "Number": "2023-E/00001",
    "Type": "Invoice",
    "Media": "PKI",
    "Customer": {
        "Name": "John Doe",
        "CountryId": "40",
        "PostalCode": "000000",
        "City": "Example City",
        "Address": "Example Street 1",
        "AccountNumber": "12345678-12345678-12345678",
        "VatNumber": "RO654321"
    },
    "Vendor": {
        "Name": "Anonym Company",
        "CountryId": "40",
        "PostalCode": "000000",
        "City": "Anytown",
        "Address": "Any Street 2",
        "AccountNumber": "87654321-87654321-87654321",
        "VatNumber": "RO123456",
        "RegistrationNumber": "",
        "Miscellaneous": ""
    },
    "Subject": "",
    "PaymentMethod": "Cash",
    "Issued": "2023-11-27",
    "Performance": "2023-11-27",
    "Prompt": "2023-11-30",
    "Paid": "0000-00-00",
    "CurrencyCode": "EUR",
    "AmountNet": "2000.00",
    "AmountVat": "540.00",
    "Amount": "2540.00",
    "IsReverseCharge": "0",
    "DocumentId": "66",
    "DocumentUrl": "https://UrlOfTheDocument",
    "DocumentFileName": "Invoice-01-2023-e-00001-1.pdf",
    "Items": [
        {
```

```
            "Name": "Product_name",
            "Description": "",
            "PriceNet": "1 000,00",
            "VAT": "19%",
            "Quantity": "2",
            "Unit": "piece",
            "PriceNetTotal": "2 000,00",
            "PriceTotal": "2 380,00"
        }
    ]
}
```

Specifications:

- "Media": "PKI" represents the default type of invoice (the electronic version) while the “Paper” represents the paper-type invoice.

Search for invoices

Detailed data of an invoice

This endpoint will search for a specified invoice, and it will return all the invoice details.

- API endpoint: **https://r3.minicrm.hu/Api/Invoice**
- Parameter: **InvoiceId** (a created invoice identifier)
- HTTP method: **GET**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey "https://r3.minicrm.hu/Api/Invoice/$InvoiceId"
```

The system will respond with the invoice details:


```
{
  "Id": "12",
  "Number": "2023-E/00014",
  "Type": "Invoice",
  "Media": "PKI",
  "Customer": {
    "Name": "John Doe",
    "CountryId": "40",
    "PostalCode": "000000",
    "City": "Example City",
    "Address": "Example Street 1",
    "AccountNumber": "12345678-12345678-12345678",
    "VatNumber": "RO654321"
  },
  "Vendor": {
    "Name": "Anonym Company",
    "CountryId": "40",
    "PostalCode": "000000",
    "City": "Anytown",
    "Address": "Any Street 2",
    "AccountNumber": "87654321-87654321-87654321",
    "VatNumber": "RO123456",
    "RegistrationNumber": "",
    "Miscellaneous": ""
  },
  "Subject": "",
  "PaymentMethod": "Cash",
  "Issued": "2023-11-27",
  "Performance": "2023-11-27",
  "Prompt": "2023-11-27",
  "Paid": "0000-00-00",
  "CurrencyCode": "EUR",
```

```
  "AmountNet": "2000.00",
  "AmountVat": "540.00",
  "Amount": "2540.00",
  "IsReverseCharge": "0",
  "DocumentUrl": "https://UrlOfTheDocument",
  "DocumentFileName": "invoice-2023-e-00014-1.pdf",
  "Items": [
    {
      "Name": "Product_name",
      "Description": "",
      "PriceNet": "1 000,00",
      "VAT": "19%",
      "Quantity": "2",
      "Unit": "piece",
      "PriceNetTotal": "2 000,00",
      "PriceTotal": "2 380,00"
    }
  ]
}
```

Invoice list endpoint

You can get a list of your issued invoices via API by using the request down below. It's important the list won't contain the invoices that aren't issued.
The listing also returns 100 results. Pagination can be done using the Page parameter, see more details in the Pagination chapter.

- API endpoint: **https://r3.minicrm.hu/Api/Invoice/List**
- HTTP method: **GET**
- Data type: **JSON**

```
https://r3.minicrm.hu/Api/Invoice/List
```

Response example:

```
{
  "Count": 3,
  "Results": [
    "145": {
      "Id": "123",
      "ProjectId": "1234",
      "StornedId": "",
      "Number": "2024-E-EUR/00001",
      "Type": "Invoice",
      "Issued": "2024-02-19",
      "Performance": "2024-02-15",
      "Prompt": "2024-02-15",
      "Paid": "",
      "CurrencyCode": "EUR",
      "AmountNet": "1599.0000",
      "AmountVat": "432.0000",
      "Amount": "2031.0000",
      "Status": "In Preparation",
      "StatusGroup": "Open",
      "DocumentUrl":""
    },
    {...},
    {...}
  ]
}
```

Update an invoice

Mark an invoice as paid

Using this endpoint, it will be possible to set the invoice status to "Paid".

- API endpoint: **https://r3.minicrm.hu/Api/Invoice/$InvoiceId/Paid**
- Parameter: **InvoiceId** (a created invoice identifier)
- HTTP method: **POST**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey -XPOST
"https://r3.minicrm.hu/Api/Invoice/$InvoiceId/Paid"
```

Optionally, it is possible to use an object with the customer data (to change customer data if needed).

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Invoice/16/Paid"
-d '{
    "Customer": {
        "Name": "John Doe",
        "CountryId": "40",
        "PostalCode": "000000",
        "City": "Example City",
        "Address": "Example Street",
        "AccountNumber": "12345678-12345678-12345678",
        "VatNumber": "RO654321"
    },
}'
```

When an invoice is marked as paid, the system response will include also the details from the invoice card:

```
{
  "Id": "17",
  "Number": "2023-E/00016",
  "Type": "Invoice",
  "Media": "PKI",
  "Customer": {
    "Name": "John Doe",
    "CountryId": "40",
    "PostalCode": "000000",
    "City": "Example City",
```

```
    "Address": "Example Street 1",
    "AccountNumber": "12345678-12345678-12345678",
    "VatNumber": "RO654321"
  },
  "Vendor": {
    "Name": "Anonym Company",
    "CountryId": "40",
    "PostalCode": "000000",
    "City": "Anytown",
    "Address": "Any Street 2",
    "AccountNumber": "87654321-87654321-87654321",
    "VatNumber": "RO123456",
    "RegistrationNumber": "",
    "Miscellaneous": ""
  },
  "Subject": "Invoice no. EUR-0-00016",
  "PaymentMethod": "Cash",
  "Issued": "2023-11-27",
  "Performance": "2023-11-27",
  "Prompt": "2023-11-27",
  "Paid": "2023-11-27",
  "CurrencyCode": "EUR",
  "AmountNet": "12000.00",
  "AmountVat": "3240.00",
  "Amount": "15240.00",
  "IsReverseCharge": "0",
  "DocumentId": "68",
  "DocumentUrl": "https://UrlOfTheDocument",
  "DocumentFileName": "invoice-2023-e-00016-1.pdf",
  "Items": [
    {
      "Name": "Product_name",
      "Description": "",
      "PriceNet": "12 000,00",
      "VAT": "19%",
      "Quantity": "1",
      "Unit": "piece",
      "PriceNetTotal": "12 000,00",
      "PriceTotal": "14 280,00"
    }
  ]
}
```

Storn an invoice

By using this endpoint, the user will be able to storn a specific invoice.

- API endpoint: **https://r3.minicrm.hu/Api/Invoice/$InvoiceId/Storno**
- Parameter: **InvoiceId** (a created invoice identifier)
- HTTP method: **POST**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey -XPOST
"https://r3.minicrm.hu/Api/Invoice/$InvoiceId/Storno"
```

In this particular case, the system will generate automatically a new storno invoice for the specified invoice.

The response will contain the details of the storned invoice:


```
{
  "Id": "17",
  "Number": "2023-E/00017",
  "Type": "Invoice",
  "Media": "PKI",
  "Customer": {
    "Name": "John Doe",
    "CountryId": "40",
    "PostalCode": "000000",
    "City": "Example City",
    "Address": "Example Street 1",
    "AccountNumber": "12345678-12345678-12345678",
    "VatNumber": "RO654321"
  },
  "Vendor": {
    "Name": "Anonym Company",
    "CountryId": "40",
    "PostalCode": "000000",
    "City": "Anytown",
    "Address": "Any Street 2",
    "AccountNumber": "87654321-87654321-87654321",
    "VatNumber": "RO123456",
    "RegistrationNumber": "",
    "Miscellaneous": ""
  },
  "Subject": "Invoice no. 2023-E/00017 Storno invoice of the invoice no.
2023-E/00016",
  "PaymentMethod": "Cash",
  "Issued": "2023-11-27",
  "Performance": "2023-11-27",
  "Prompt": "2023-11-27",
```

```
  "Paid": "2023-11-27",
  "CurrencyCode": "EUR",
  "AmountNet": "12000.00",
  "AmountVat": "3240.00",
  "Amount": "15240.00",
  "IsReverseCharge": "0",
  "DocumentId": "68",
  "DocumentUrl": "https://UrlOfTheDocument",
  "DocumentFileName": "storno_invoice-2023-e-00017.pdf",
  "Items": [
    {
      "Name": "Product_name",
      "Description": "",
      "PriceNet": "12 000,00",
      "VAT": "19%",
      "Quantity": "1",
      "Unit": "piece",
      "PriceNetTotal": "12 000,00",
      "PriceTotal": "14 280,00"
    }
  ]
}
```

Editing custom fields (invoice card fields)

This endpoint allows the user to fill or change the custom fields that are located on the invoice card.

- API endpoint: **https://r3.minicrm.hu/Api/Invoice/$InvoiceId/Project**
- Parameter: **InvoiceId** (a created invoice identifier)
- HTTP method: **POST**
- Data type: **JSON**

The request URL should be something like:

```
$ curl -s --user $SystemId:$ApiKey -XPOST
"https://r3.minicrm.hu/Api/Invoice/$InvoiceId/Project"
```

The user should specify the name of the field he wants to update:

```
{
   "FieldName": "Test",
}
```

Stock endpoint

Using this endpoint, you will be able to interact with product and/or service stocks that it is used for products like invoices, Offer, and Orders.

There are special endpoints for these products:
- **Invoice endpoint**: https://r3.minicrm.hu/Api/Invoice/List
- **Offer endpoint**: https://r3.minicrm.hu/Api/Offer/List
- **Order endpoint**: https://r3.minicrm.hu/Api/Order/List

These endpoints will list only **issued documents**.

Pagination limits also apply, and you may find more details here.

Additional pagination details for stock-related products

For example, in the invoice product (but this also applies to orders and offers), the invoices that have been most recently modified (not the datasheet related to the invoice) come first:


```
{
  "Count": 3,
  "Results": [
    "145": {
      "Id": "145",
      "ProjectId": "3529",
      "StornedId": "",
      "Number": "AJ2024-E/00001",
      "Type": "Offer",
      "Issued": "2024-02-19",
      "Performance": "2024-02-15",
      "Prompt": "2024-02-15",
      "Paid": "",
      "CurrencyCode": "HUF",
      "AmountNet": "1599.0000",
      "AmountVat": "432.0000",
      "Amount": "2031.0000",
      "Status": "In Preparation",
      "StatusGroup": "Open",
      "DocumentUrl":
```

```
"https://r3.minicrm.hu/xxxxx/Download/S3/?t=doc&inline=0&e=2h99asbfjn1zvi9e7yjw0jas6ewk27"
    },
    {...},
    {...}
  ]
}
```

List filtering parameters

Page (*optional*)

```
https://r3.minicrm.hu/Api/Invoice/List?Page=1
```

UpdatedSince (*optional*): The date of the last modification of the invoice. Invoices modified more recently than this date will be included in the list

```
https://r3.minicrm.hu/Api/Invoice/List?UpdatedSince=2024-02-12 10:00
```

StatusGroup (*optional*): The current status group of the datasheet related to the invoice (Lead, Open, Success, Failed).

```
https://r3.minicrm.hu/Api/Invoice/List?StatusGroup=Success
```

Filters can be combined to obtain a more refined search for your documents:

```
https://r3.minicrm.hu/Api/Invoice/List?StatusGroup=Success&UpdatedSince=2024-02-12 10:00&Page=2
```

Adding a new product

- API endpoint: **https://r3.minicrm.hu/Api/R3/Product/**
- HTTP method: **PUT**
- Data type: **JSON**

```
{
   "Name": "Apple Pie",
   "SKU": "APPLE_PIE",
   "EAN": "123456789",
   "FolderName": "Basic Services",
   "VAT": "19%",
   "PriceNet": 10,
   "CurrencyCode": "RON",
   "UnitName": "Unit_Piece",
   "Quantity": 5,
   "Warehouse_Name": "Warehouse Name",
   "Description": "Delicious apple pie.",
   "Deleted": 0
}
```

SKU: If not empty and already existing in the system, the system **will update the existing data instead of creating a new product.**

FolderName: If not specified or does not exist, it will be placed in the first one recorded in the system.

Custom-created fields can also be sent.

The response includes the generated MiniCRM product ID:

```
{ "Id": 1 }
```

Update a product

- API endpoint: **https://r3.minicrm.hu/Api/R3/Product/$Id**
- Parameter: **Id**: replace with the MiniCRM product ID found in the response when creating a new product. The expected JSON is the same as when creating a new product. If not specified, the default currency will be used for CurrencyCode (as stored in the database, e.g., HUF in Hungary, RON in Romania, etc.), and for VAT, it will be the highest VAT rate for the given country (theoretically, the default everywhere), which is 27% in Hungary, 19% in Romania, etc.
- HTTP method: **PUT**
- Data type: **JSON**

```
{
   "Name": "Apple Pie (updated)",
   "SKU": "APPLE_PIE",
   "EAN": "123456789",
   "FolderName": "Basic Services",
   "VAT": "19%",
   "PriceNet": 10,
   "CurrencyCode": "RON",
   "UnitName": "Unit_Piece",
   "Quantity": 5,
   "Warehouse_Name": "Warehouse Name",
   "Description": "Delicious apple pie.",
   "Deleted": 0
}
```

If the update was successful, the system will reply with the product id

```
{ "Id": 1 }
```

Delete a product

To be able to delete a product, you need to update it using the parameter *Deleted* and use 1 (one) for its value

- API endpoint: **https://r3.minicrm.hu/Api/R3/Product/$Id**
- Parameter: **Id**: replace with the MiniCRM product ID found in the response when creating a new product. The expected JSON is the same as when creating a new product. If not specified, the default currency will be used for CurrencyCode (as stored in the database, e.g., HUF in Hungary, RON in Romania, etc.), and for VAT, it will be the highest VAT rate for the given country (theoretically, the default everywhere), which is 27% in Hungary, 19% in Romania, etc.
- HTTP method: **PUT**
- Data type: **JSON**

```
{
   "Deleted": 1
}
```

If the deletion was successful, the system will reply with the product id:

```
{ "Id": 1 }
```

CallLog API Endpoint

This interface is created for VOIP providers. The metadata of phone calls can be sent to MiniCRM via CallLog API so they will appear in the history section on the opportunity card of the relevant customer.

Before sending requests to this endpoint, you should provide the IP addresses for authorization. Please use our contact data for this purpose.

Create a new call log entry

- URL: **https://r3.minicrm.io/Api/CallLog**
- HTTP method: **POST**
- Sent data must be in **JSON** format:
  - *ApiKey*: the VOIP API key, the admin can generate the key in MiniCRM / Settings / System. It is a separate key from the "generic" MiniCRM API key, which does not work with the VOIP API
  - *UserExtension*: internal phone number of the user who initiated / received the call. Internal user numbers can be set in MiniCRM / Profile. If the PBX does not use internal numbers, the user's full phone number can be sent in this field.
  - *Data*: arrays of history items, each consisting of:
    - *Number* (string): the phone number of the other party in the call
    - *Duration* (int): call duration in seconds
    - *CallType* (int):
      - 0: Outgoing
      - 1: Incoming
      - 2: Lost
    - *Date* (string/datetime): the date and time the call started, in the UTC time zone
    - *ReferenceId* (string) [optional]: Unique identifier of the recorded call, if the call was recorded.

Response

If everything went well, the API responds with the HTTP 200 / OK response code. In case of errors, HTTP 4XX or HTTP 5XX codes are sent along with the error description.

The response is a JSON-encoded object with summary statistics of the history records processed:

- **Missing**: The number of records where at least one of the parameters is missing.
- **NotFound**: the number of records where the other party's phone number was not found.
- **Processed**: Successful processing, history records saved in MiniCRM.

Request example #1

```
$ curl -s --user $SystemId:$ApiKey -v -X POST -H "Content-Type: application/json"
"https://r3.minicrm.hu/Api/CallLog" -d '
{
  "ApiKey":"<API_KEY>",
  "UserExtension":"001",
  "Data":[{
      "Date":"2016-03-02 16:00:12",
      "Number":"0620123456",
      "CallType": 2,
      "Duration": 132,
      "ReferenceId":"UniqueReferenceId"
     }]
}'
```

Response:

```
{
    "Skipped": 0,
    "Processed": 1,
    "Exists": 0
}
```

Request example #2

The first record contains errors, the second is from an unknown number, and the third item can be saved.

```
$ curl -s --user $SystemId:$ApiKey -v -XPOST "https://r3.minicrm.hu/Api/CallLog" -d '
 {
    "ApiKey": "<API_KEY>",
    "UserExtension": "001",
    "Data": [
        {
            "Date": "2016-aíYA03-02 16:00:12",
            "CallType": 2,
            "Duration": 132,
            "ReferenceId": "UniqueReferenceId"
        },
        {
            "Date": "2016-03-02 16:00:12",
            "Number": "0620123456",
            "CallType": 2,
            "Duration": 435,
            "ReferenceId": "UniqueReferenceId"
        },
        {
            "Date": "2016-03-03 12:12:12",
            "Number": "0620123456",
            "CallType": 2,
            "Duration": 251,
            "ReferenceId": "UniqueReferenceId"
        }
    ]
}'
```

Response example #2

```
{
    "Skipped": 1,
    "Processed": 1,
    "Exists": 1
}
```

Search for phone calls

Listening calls in MiniCRM

If the ReferenceId is sent with the calls, it is possible to listen to the calls in MiniCRM. Without ReferenceId, only call metadata appears in the datasheet history. An optional feature is the playback of the recorded call (if any).

There are two things to do.

1. Any recorded call must submit a unique ID in the "ReferenceId" field.
2. Then, send the URL of the recorded call to suport@minicrm.ro. It should be a template that we will use to generate the URL from which MiniCRM can download the recorded audio file

URL Example:

```
https://yourdomain.com/recordings.php?Rec={%RefereceId%}
```

HTTPS – the communication is over a secure channel, recorded calls are sensitive data. yourdomain.com is the server used by the VoIP service provider to store records.

We will download the files from the following IP addresses: 195.228.254.37, 195.228.254.43, and 195.228.156.188. It is recommended to enter an IP restriction on the endpoint to ensure that only these IP addresses can access them.

Integrating Webforms through API with Advanced Webform Addon

To use this method, you need the *Advanced webform* addon turned on!

By this, you can create a web form within your MiniCRM system and connect it with your own custom solution to send in data via API to your webform and create a card within your system.

Key advantages

- Unified Request: Submit business, primary contact, and opportunity card details in a single API request.
- No duplications: Using the endpoint MiniCRM will automatically identify existing contacts, and assign new opportunities to them. You don't need to look up contact by email first in a separate request.

- API endpoint: **https://r3.minicrm.hu/Api/Signup**
- HTTP method: **POST**
- Data type: **JSON**

Creating a Webform

1. Create a webform in your system with the desired fields included.
2. Get fields names from the HTML code of the form
    a. Access the HTML code of the webform in the webform editor in MiniCRM. Click "To see a detailed description, click here." under "Inserting a form into your website".

    b. Copy each field's name. From this:

    <input name="Contact[1891][FirstName]" id="Contact_Name_1331" language="EN" type="text" />

    Copy out this: Contact[1891][FirstName]

    c. Copy the form's identifier, called FormHash. From this:

    <form FormHash="61647-1ot9wrbkba0d0022zptc0m3agmw1vb" action="https://r3.minicrm.hu/Api/Signup" method="post">

    Copy out this: 61647-1ot9wrbkba0d0022zptc0m3agmw1vb

Form Submission Via API

Example API Request using `curl` to send a POST request:

```
$ curl -s --user $SystemId:$ApiKey -XPOST "https://r3.minicrm.hu/Api/Signup" \
  --form "FormHash=61647-1ot9wrbkba0d0022zptc0m3agmw1vb" \
  --form "Contact[6786][FirstName]=API test" \
  --form "Contact[6786][Email]=api@example.com" \
  --form "ToDo[6788][Comment]=Test"
```

API Response Explanation
Upon form submission, the API will return a JSON response similar to this:

```
{
 "Url": "",
 "Info": "The provided data has been successfully processed.",
 "Warning": "",
 "Error": "",
 "Message": "",
 "ErrorFieldId": "",
 "Data": {
    "ProjectHash": "<PROJECT_HASH>",
    "ContactHash": "<CONTACT_HASH>",
    "Email": "api@example.com",
    "Name": "John Doe"
  }
}
```

```
{
 "Url": "",
 "Info": "The provided data has been successfully processed.",
 "Warning": "",
 "Error": "",
 "Message": "",
 "ErrorFieldId": "",
 "Data": {
    "ProjectHash": "<PROJECT_HASH>",
    "ContactHash": "<CONTACT_HASH>",
    "Email": "api@example.com",
    "Name": "API test"
  }
}
```

Response Fields

- Info: Confirmation message indicating successful data processing.
- Error: If it's not empty, the submitted data is not saved and has to be corrected according to the error message (for example, missing fields, invalid email etc…). The error message can be shown to the end user to help correct the error.
- ErrorFieldId: The id of the field that failed the checks. It can be used to highlight the field for the end user.
- Data: Contains essential data hashes and details for further reference.

CONTACT

Please don't hesitate to contact us[3] in case of any question related to the API and/or other technical question you may have using the following email addresses below:

- Romanian customer support: suport@minicrm.ro
- Hungarian customer support: help@minicrm.hu
- International customer support (please write in English): help-en@minicrm.io

Also, if you use MiniCRM, you will find our phone number written directly into your MiniCRM account clicking the ⓘ button from the right top corner (the phone numbers are specific to each country and can change over time - this is why we didn't include them here).

[3] If you are a dev, and you are writing to us on behalf of a customer, please mention the customer name or his company name in your email.