Inhoud

[Contacts 2](#_Toc9941002)

[Contacts related to the distribution partner 2](#_Toc9941003)

[Contacts related to companies 2](#_Toc9941004)

[Employee 2](#_Toc9941005)

[Company owner 2](#_Toc9941006)

[User 3](#_Toc9941007)

[Functional requirements for API 4](#_Toc9941008)

[Distribution partner scope 4](#_Toc9941009)

[Company scope 4](#_Toc9941010)

[Contact properties 5](#_Toc9941011)

[Endpoints 5](#_Toc9941012)

[SQL Changes: 5](#_Toc9941013)

# Contacts

Contacts are person records (or sometimes companies, but we ignore this in this stage of the project for now) in the system. Most of the time a contact is connected to a company, but they are also (always) connected to the distribution\_partner. So a contact is primarily connected to a distribution partner, and, following the contact\_type it can be connected to a company id also.

### Contacts related to the distribution partner

A contact record is always related to 1 distribution\_partner\_id, see SQL Changes for the required additional SQL for the new DB field. This field is missing now. Examples:

* Employee (see chapter Employee)
* Users (contact records of type employee with user connection in user table)

### Contacts related to companies

In addition to the distribution partner relationship, a contact can be connected to a company using the “company\_contact” table. With this relation comes a contact\_type field, relating to the table “contact\_type”, this defines in what role the contact is connected:

* Company owner
* Company contact

With this relation, the same contact record can be added multiple times to the same company, but with a different contact type. We may see more contact\_types later on.

## Employee

An employee is an contact record of type employee for a distribution partner. Example: Martin van der Marel is a contact record of contact\_type employee, belonging to distribution\_partner\_id 1 (CreditDevice B.V.).

To get a full list of employees belonging to CreditDevice B.V., we select all contacts of contact\_type employee with distribution\_partner\_id 1.

## Company owner

A company owner can be connected only once to a company. Only a contact of contact\_type employee can be selected (in front end) to be connected as a company owner. Example: Martin van der Marel is a contact record (id=1) of contact\_type employee, belonging to distribution\_partner\_id 1. To set the company owner for company 1, the end-user selects from all employees of distribution\_partner\_id 1the employee Martin van der Marel. The result is that the system adds a record to the company\_contact table with reference to contact id =1 with contact\_type\_id = 4 (company\_owner).

## User

A user is a **contact** record connected to a “**users**”-table record (user\_id) in “contact” table. This means, this contact is allowed to login and see relevant data for the user.

A user has a distribution\_partner\_id, which defines if the user can see data from the distribution\_partner\_id only, or everyting. The role field defines which permissions a user has.

For now, we only use the scenario where a contact of contact\_type employee can be connected to a user.

# Functional requirements for API

We define the following scopes for the contact operations:

* Distribution partner scope
* Company scope

## Distribution partner scope

At the distribution partner scope, the following functions can be defined:

* Employees
  + Get a list of all employees for distribution partner (GET /employees)
  + Get a employee record (GET /employee/:id)
  + Add an employee (POST /employees)
  + Update an employee (PUT /employee/:id)
  + Delete an employee (date\_deleted) (DELETE /employee/:id)
* Users
  + Get a list of all users (GET /users)
  + Get a user (/GET /user/:id)
  + Add a user (POST /users)
  + Update a user (PUT /users/:id)
  + Delete a user (date\_deleted) (DELETE /user/:id)

## Company scope

At the company scope, the following functions can be defined, meaning, these functions get/put/delete contact related information in combination with a company id.

* Company owner
  + Set an employee as company owner (POST /company/:id/owner)
    - Keep in mind here, setting the owner, must unset the other existing owner (dropping existing relationships)
  + Drop relationship (DELETE /company/:id/owner)
  + Get companyowner (GET /company/:id/owner)
    - Gets a contact-record, if any for a company id
* Company contact
  + Get a list of all contacts for company id (GET /company/:id/contacts)
  + Add contact (POST /company/:id/contacts)
  + Drop relationship (DELETE /company/:id/contact/:id)
  + Edit contact (GET /company/:id/contact/:id
  + Update contact (PUT /company/:id/contact/:id)

# Contact properties

A contact-record has fields to capture the information needed to define the contact either as a company (some of the cases) or a person (most of the cases). A contact record has a relation with 0 or many emails and 0 or many telephone numbers.

Using the GET requests of contacts, we always add the phonenumbers and emailaddresses as “subkeys” to the individual contact entries in the response.

## Endpoints

For the email / phone properties we can define these endpoints:

* Get all phonenumbers for contact (GET /contact/:id/phone)
* Get all emails for contact (GET /contact/:id/email)
* Drop phonenumber (DELETE /contact/:id/phone/:id)
* Drop email (DELETE /contact/:id/email/:id)
* Add phonenumber (POST /contact/:id/phone)
* Add email (POST /contact/:id/email)
* Update phonenumber (PUT /contact/:id/phone/:id)
* Update email (PUT /contact/:id/email/:id)

# SQL Changes:

ALTER TABLE `cd-customer-platform`.`contact`

ADD COLUMN `distribution\_partner\_id` INT(11) NULL AFTER `id`;