# berliCRM

# REST Web services API Tutorial

Version 1.5.2
Rev. from 01.03.2018

# berliCRM: API Reference Manual

© 2004- 2018 crm-now GmbH, All rights reserved.

**Trade Marks**

Many of the designations used by manufacturers and retailers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and crm-now was aware of a trademark claim, the designations have been printed in caps or initial caps. While every precaution has been taken during the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damage resulting from the use of the information contained herein.

For more Information about crm-now you may look up:

**www.crm-now.com**

Manual ID: 532-002-001

# Table of Contents

# 1 Overview

The CRM system provides a simple, powerful and secure application programming interface (the API) to create, modify or delete CRM contents.

This document is a tutorial for developers which intend to use REST based APIs to connect other applications with the CRM system. It is based on the document "API Reference Manual" available as a separated document.

# 2 PHP Samples

This tutorial is using PHP as language and explains the web service operations by examples. By analogy, this can also be transferred to other languages, like Java Script or .Net.

This tutorial relates to separate files:

- a special CURL class for initiating POST and GET operations to the CRM, for getting the responses and for making an error handling (berliCRM_WS_Curl_Class20.php)
- a special PHP file with operation examples which use the CURL class (berliCRM_webserviceexamples.php)
- the API Reference Manual

To exercise these examples you must have CURL and JSON installed at your server.

# 3 WS Curl Class

If you are not familiar with CURL look up http://php.net/curl. CURL is used as the wrapper for the WS_Curl_Class which executes the REST Login and the REST operations for the CRM.

## 3.1 getChallenge Method

This method provides a unique token for the login operation.

## 3.2 login Method

The login method checks the user's credentials and provides a session id. The session id identifies the current session and will be a common parameter in all subsequent operations.

With a default PHP settings a session id is valid for app. 24 minutes. This time extends automatically with each operation for a maximum of 24 hours.

The login method also returns the internal user ID of the logged in user. Web service which require a user ID (such as create) can use this ID to set an entity's ownership.

## 3.3 operation Method

The operation method handles all POST and GET operations. For a list of all supported operations please look up the berliCRM API Reference Manual.

### 3.4 handleReturn Method

The handleReturn method encodes the return data of an operation and sets an error message if operation fails.

# 4 Object IDs

For all REST operations the ID of a data object has the following format:

<module id>**x**<record id>

**Module id:** Each CRM module has an internal unique web service module ID. You can obtain this number look up the entity id of a record by doing a select operation.

**record id:** This is the unique CRM record id. You may obtain such an id also from browser's URL when you open in the CRM a Detail View of a module record as "record=<record id>"

# 5 Operation Examples

In the following web service operations are explained by example program codes.

## 5.1 Logging In

The API does not use the CRM users' password to log in. Instead, the CRM provides a unique access key for each user. To get the user key for a user, go to the "*My Preferences"* menu of that user in the CRM. There you will find an Access Key field.

The Login procedure starts a client session with the CRM server, authenticates the user and returns a sessionId which will be used for all the subsequent communication with the CRM server.

Logging in is a two-step process. In the first step the client obtains the challenge token from the server, which is used along with the user's access key for the login.

**Program reference:** EXAMPLE 1: Login

## 5.2 Getting CRM Systems Information

The API provides two operations to get information about available CRM objects.

### 5.2.1 Get CRM Object Names

The listTypes operation list the names of all the CRM objects available through the API. This list only contains modules the logged in user has access to, controlled by the CRM's privilege system.

**Program reference:** EXAMPLE 2: listTypes

### 5.2.2 Get CRM Object Properties

Provides property information about a given CRM object/module.

The returned description consists of the following fields:

1. *label* - The label used for the name of the module.
2. *name* - The name of the module.
3. *createable* - A boolean value specifying whether the object can be created.
4. *updateable* - A boolean value specifying whether the object can be updated.
5. *deleteable* - A boolean value specifying whether the object can be deleted.
6. *retrieveable* - A boolean value specifying whether the object can be retrieved.
7. *fields* - An array containing the field names and their type information.

Each element in the fields array describes a particular field in the object.

1. *name* - The name of the field, as used internally by the CRM.
2. *label* - The label used for displaying the field name.

3. *mandatory* - This is a boolean that specifies whether the field is mandatory, mandatory fields must be provided when creating a new object.
4. *type* - An map that describes the type information for the field.
5. *default* - The default value for the field.
6. *nillable* - A boolean that specifies whether the field can be set to null.
7. *editable* - A boolean that specifies whether the field can be modified.

The type field is of particular importance. It provides a map that will contain at least an element called name which is the name of the type. The name could be one of the following.

1. *string* - A one line text field.
2. *text* - A multiline text field.
3. *integer* - A non decimal number field.
4. *double* - A field for for floating point numbers.
5. *boolean* - A boolean field, can have the values true or false.
6. *time* - A string of the format hh:mm, format is based on the user's settings time format.
7. *date* - A string representing a date, the type map will contain another element called format which is the format in which the value of this field is expected, it is based on the user's settings date format.
8. *datetime* - A string representing the date and time, the format is based on the user's settings date format.
9. *autogenerated* - These are fields for which the values are generated automatically by the CRM, this is usually an object's id field.
10. *reference* - A field that shows a relation to another object, the type map will contain another element called refersTo which is an array containing the name of modules of which the field can point to.
11. *picklist* - A field that can a hold one of a list of values, the map will contain two elements, picklistValues which is a list of possible values, and defaultValue which is the default value for the picklist.
12. *multipicklist* - A picklist field where multiple values can be selected.
13. *phone* - A field for storing phone numbers
14. *email* - A field for storing email ids
15. *url* - A field for storing urls
16. *skype* - A field for storing skype ids or phone numbers.
17. *password* - A field for storing passwords.
18. *owner* - A field for defining the owner of the field, which could be a group or individual user.

**Program reference:** EXAMPLE 3: describe

## 5.3 Data Objects Operations

The API provides operations to create, retrieve, update and delete CRM entity objects. They are identical for most CRM objects but not for all.

### 5.3.1 Query the CRM's data base

The CRM provides a simple query language for fetching data. This language is quite similar to select queries in MySQL. There are limitations as described in the API Reference manual. Nevertheless, the remaining options have a powerful ways for getting data from the CRM.

The main difference between the API queries and MySQL is, that you do not need to know anything about the tables in the CRM's data base. Your queries operate only with the object names as provided by the listTypes operation.

You can use the limit operator to get less than 100 records.

**Program reference:** EXAMPLE 4: query

A single query always limits its output to 100 records. If you want to get more than 100 records you need to loop with limit and offset operators.

**Program reference:** EXAMPLE 7: loop query

The CRM distinguishes between tasks and events while both are calendar entries and considered as activities. The special query operations are shown in the example.

**Program reference:** EXAMPLE 9: get Event and Tasks

### 5.3.2 Get a single CRM entry

This operation lets you retrieve the master data of a single entry from the CRM system by its object id.

**Program reference:** EXAMPLE 5: retrieve

### 5.3.3 Create single or multiple CRM entries

The create operations can be used to import data. In return you will get a key called id which represents the CRM object's unique id and which can be used for further data operations.

For your data you must consider all mandatory fields in the CRM marked by a red * in the CRM's Edit View GUI, except the "assigned" to field.

If you do not set the "assign to" field content, the user id of the logged in user is used automatically. It is not recommended to assign the data to the admin user. It is recommended to assign each data set to a particular user. This has the advantage that these data can get transferred to another user easily by a CRM GUI operation that deletes a CRM user.

There is no special handling for the mandatory field "lastname" for the Leads module as it is provided by the GUI import operation. If you do not provide contents for "lastname" the creation will fail.

The following example shows how to create a new CRM entry.

**Program reference:** EXAMPLE 10: create product

The following program example shows you how to use the create operation to import multiple contacts at once from a CSV file. Make sure your CSV file is UTF8 (without BOM) coded. Some CRM fields, like multi pick lists or dates, need a special formatting. Consult the CRM manual for a detailed description of these formats.

**Program reference:** EXAMPLE 8: create Contacts from file

A document CRM entry may include an uploaded file. The following example shows you how to upload a file while creating a new document.

**Program reference:** EXAMPLE 11: create and upload Document file

Any quote, sales order, invoice or purchase order entry consist of its master data and line items. The line items are the products or services you want to have in a sales order. You must have at least one line item included.

**Program reference:** EXAMPLE 15: create Sales Order

### 5.3.4 Update an existing CRM entry

The update operation can be used to modify the field contents of an existing CRM entry.

Be aware that every field of a CRM object will be emptied if no value is supplied on an update operation. Therefore, it is recommended that you use a retrieve operation to get an entry, do your modifications to the data and use the update operation to transfer your modifications to the CRM.

**Program reference:** EXAMPLE 6: update

### 5.3.5 Revise an existing CRM entry

The revise operation lets you modify a field contents of a CRM data set without sending all object data.

As explained in the API reference manual, a revise operation could fail without an error message, if you try to revise a field unknown to the logged in user. A field could be unknown if you provide a wrong field name of if the user does not have access privileges as set by a profile attached to the user's role.

Therefore, if you are not sure, you should try a data set modification with the update operation first.

**Program reference:** EXAMPLE 18: revise

### 5.3.6 Delete an existing CRM entry

The delete operation makes a CRM entry invisible to CRM user's by moving it to the recycling bin. It could get restored from the recycling bin. There is no web service operation which deletes an entry permanently from your CRM.

**Program reference:** EXAMPLE 21: delete

## 5.4 Special Data Objects Operations

### 5.4.1 Get related campaign entries

The retrieve operation for campaigns covers only the campaign's master data. If you have contacts, leads or accounts related to a campaign, the get_campaign_entities operation provides the related IDs

**Program reference:** EXAMPLE 16: retrieve Campaign related entities

### 5.4.2 Convert a Lead

The convertlead operation provides:

- the creation of a contact, an account and a potential CRM entries
- a data transfer of the lead's master date to the created CRM entries
- the transfer of the lead's related lists contents

You can decide which module gets created and whether the new created contact should get attached to an existing account. But you cannot define how the data transfer of related lists will get performed. Related list entries will always get transferred to the contact.

Make sure you consider all mandatory fields for the Potential and the related Account and Contact.

**Program reference:** EXAMPLE 19: convert Lead

### 5.4.3 Get Entities related to a Document

In the CRM documents can be related to one or more CRM entries. This get_document_relations function provides you a list of all entity modules and the related IDs which are related to a particular document.

**Program reference:** EXAMPLE 12: retrieve related document entries

### 5.4.4 Set Entities related to a Document

The update_document_relations operation let you set the relationship of documents to other entries. With a single operation you can add additional relationships or you can substitute relationships controlled by the "preserve" parameter.

**Program reference:** EXAMPLE 13: relate document to other entries

### 5.4.5 Get Document Attachment

The retrieve operation for documents covers only the document's master data. If you have a file attached the retrievedocattachment operation provides this file.

The file returned is base64 coded. The example shows how you can make it a download file for a browser.

**Program reference:** EXAMPLE 14: retrieve Document file

### 5.4.6 Change CRM User Password

The changePassword operation lets you change the password of a user. It is recommended that you follow the rules as described in the CRM manual to us a strong password. For security reasons you must provide the old password.

**Program reference:** EXAMPLE 20: changePassword

### 5.4.7 sync

The sync operation let you find out which data had been changed or got deleted related to a given time stamp as exists in the CRM's data base. The time stamp used for the sync operation is a long representation of the number of seconds since unix epoch.

When defining a time stamp you need to consider, that the time shown as modified time in an entities Detail View in the CRM is not the time which is in the data base. The internal CRM time is defined by the CRM system setup. For Central Europe for instance the internal time differs one or two hours (summer and winter time) from the time displayed in a CRM's Detail View.

**Program reference:** EXAMPLE 17: sync

## 5.5 Logging Out

The logout operation should be part of every web service application to make a used session invalid for further use.

**Program reference:** EXAMPLE 22: logout

# 6 Acknowledgement

This tutorial was originally provided by vtiger™. With the friendly permission of vtiger™ it has been modified to meet the specific requirements of crm-now's berliCRM version.