Developers Guide
version 1.4

Introduction

QuickCRM app can be customized to adapt display or behavior to your specific context beyond customizations you can make from your CRM admin pages.
This guide will help you implement your customizations.

It is intended for developers familiar with JavaScript, JQuery and HTML5, and, of course, SugarCRM or SuiteCRM development framework.

If you don't have these skills and need help from our Professional Services team, please contact us at support@quickcrm.fr.

Generalities

All customizations should be written in JavaScript and placed in a file named custom.js placed in folder custom/QuickCRM on your CRM server.

This file is automatically included when the app is loaded.
However, in order to improve loading time and to allow Offline mode, the file is stored locally on your device once loaded.
So, whenever you make changes to that file, you have to go to your CRM admin page and click "QuickCRM Update". That way, the app will know that the configuration has changed and your customizations will be reloaded next time the app is opened.

Please note that in your customizations, all modules, fields or dropdown values should be referenced by internal name and not by the label as displayed in the CRM.
For example, you would manipulate modules ProspectLists or AOS_Quotes and not Target Lists or Quotes. You would use field named account_type for an Account type.

Please remember that JavaScript language is case-sensitive.
If JavaScript errors happen when loading your custom file, none of your customizations will be applied.
If errors occur during execution, it might also prevent correct execution of the app.
So, we strongly recommend you test your customizations on a test server before publishing them on your production server.

Release Note

Version 1.4

- New logic hooks:
  - onchange (app version 6.2.2 and later)
  - DuplicateButton
  - CreateFromSubpanelOnly (app version 6.2.3 and later)
  - QCRM.SetFieldRequired (app version 6.2.3 and later)
  - QCRM.SetFieldReadonly (app version 6.2.3 and later)
- New utilities

Version 1.3

- New check_before_save logic hooks (for example, make a field required depending on the value of another field)

Version 1.2

- New before_create and before_update logic hooks

Summary

User Management ... 4

Home Page Management ... 5

- Add Comments, Images or links ... 5
- Hide a module ... 5
- Disable duplication ... 5
- Disable create button ... 5

Fields Management ... 6

- Change field appearance ... 6
- Make a field read-only ... 6
- Make a field required ... 6
- Remove a field from Edit View ... 6
- Auto-Populate fields from a parent ... 6
- Filter relate fields ... 6

Search options ... 8

- Starting with or Containing ... 8
- Results display ... 8

Detail View customization ... 9

- init_view ... 9
- before_display ... 9
- after_display ... 9

Edit View customization ... 10

- check_edit ... 10
- check_delete ... 10
- onchange ... 11
- init_edit ... 11
- before_edit (new records and existing records) ... 11
- before_create (new records) ... 12
- before_update (existing records) ... 12
- check_before_save ... 12
- before_save ... 12
- QCRM.SetFieldReadonly = function (module,field,Bool readonly) ... 12
- QCRM.SetFieldRequired = function (module,field, Bool required) ... 12

Utilities ... 13

User Management

Customizations might depend on current user characteristics.

Within hook functions you can access:

- `Global variable CurrentUserId`
- `QCRM.UserHasRole(role) function`
- `QCRM.UserHasTeam(team) function`

IMPORTANT: Current User Id, Roles and Teams are only available once the user is logged in.

Any hook using these should then be placed within an after_login app hook

Examples

Check if current user has role “management”

```
if (QCRM.UserHasRole ('management')){
  // Do something
}
```

Check if current user belongs to team “Sales”

```
if (QCRM.UserHasTeam ('Sales')){
  // Do something
}
```

Make a field read-only if user does not have role “management”

```
QCRM.SetAppHook('after_login',function(){

        QCRM.SetFieldHook('Accounts','account_type',
          'readonly', !QCRM.UserHasRole ('management'));

});
```

Home Page Management

Add Comments, Images or links

A section with id HomeCustom is available at the top of the home page to place comments, images or links. You can place any html in this section.

Examples

Add an image:

```
$('#HomeCustom').append(
'<img src="http://www.quickcrm.fr/mobile/img/logo.jpg"/>'
);
```

Add a link to a web page:
Important: Please note that links should be opened with the OpenURL function rather than with href

```
$('#HomeCustom').append(
'<a onclick="OpenURL(\'http://mobile.quickcrm.fr\')">QuickCRM</a>'
);
```

Hide a module

Modules can be made completely unavailable from the admin page in the Select Modules page. However, you might want a module to be available for relate fields or in subpanels without its icon being available on the Home Page or in the All Modules popup.

Example

Have Target Lists visible under contacts but without an icon on the Home Page (this assumes that Target Lists are available in Select Modules and the subpanel has been added to Contacts)

```
QCRM.SetModuleHook('ProspectLists','ShowTab',false);
```

Disable duplication

```
QCRM.SetModuleHook('ProspectLists','DuplicateButton',false);
```

Disable create button

Creation will then only be available from subpanels

```
QCRM.SetModuleHook('Opportunities','CreateFromSubpanelOnly',true);
```

Fields Management

Change field appearance

Examples

Display in red field discount_c of Opportunities when its value is not 0

```
QCRM.SetFieldHook('Opportunities',discount_c','display',
   function (value){
      if (value != 0)
            return '<span style="color:red;">'+value+'</span>';
      return '';
});
```

Add % after probability field value

```
QCRM.SetFieldHook('Opportunities','probability','display',
   function (value){
      return value+'%';
});
```

Remove a field from Edit View

A field can be removed from the Edit View using the noedit hook.
Please see Edit View customization if you want to hide a field conditionally.

Example

Remove Meetings Location field from Edit view

```
QCRM.SetFieldHook('Meetings','location','noedit',true);
```

Auto-Populate fields from a parent

Fields from a parent can be copied to a record created from the parent subpanel or when selecting the parent from the record Edit view

Example

Copy phone_office and billing address from Account to Contact phone_work and primary address.

```
QCRM.SetFieldHook('Contacts','account_name','copyfields',
 [{copy:'phone_work',from:'phone_office'},
  {copy:'primary',from:'billing'}
 ]
);
```

Filter relate fields

When searching for a record in a relate field, you sometimes want to search a subset of the records based on a filter.
This is the equivalent of the displayParams / initial_filter you would place in editviewdefs.php

Examples

The selected account must have Type Customer

```
QCRM.SetFieldHook('MY_MODULE','account_c','initial_filter',
  function (form){
      return "account_type = 'Customer'";
});
```

The contact selected in a Quote must be a contact of the Quote’s account

```
QCRM.SetFieldHook('AOS_Quotes','billing_contact','initial_filter',
  function (form){
      var account_name =
      QCRM.toDBField('AOS_Quotes','billing_account',form);
      if (account_name && account_name !==''){
        return "account_name LIKE '"+account_name.replace(/'/g,"''")+"'";
      }
      return "";
});
```

Search options

Starting with or Containing

By default, QuickCRM always search fields containing the search value (for example, searching for “ank” would return “Bank”.
In some situations, like zip code or account number, you would like searching for 34 returning 3401 or 3402 but not 7634.

The searchmode option allows you to define this for specific fields.
Search mode can be:

- partial : field value contains the searched text
- start : field value starts with the searched text
- exact : field equals the searched text
- default : partial or start depending on the preferences set in Options / General

Example

Set search mode to “starting with” for field acct_no_c of Accounts

```
QCRM.SetFieldHook('Accounts','acct_no_c','searchmode', 'start');
```

Results display

When searching records with Global search, or for relate fields, results show the “name” field of records.
You can add other fields on records found by the search

Example

When you have multiple accounts with the same name, it’s rather difficult to find the account you are looking for as the list shows accounts that look identical.
You can, for example, add fields “billing_address_city” and “acct_no_c” with:

```
QCRM.SetSearchTemplateFields('Accounts',
['billing_address_city', 'acct_no_c']);
```

Detail View customization

Hooks allow you to customize Detail View, for example, add a button or hide fields depending on other fields.

Available hooks are:

- init_view : executed the first time the view is created. This would allow you, for example to add a button to a view
- before_display : executed once the record is retrieved
- after_display : executed after the data are displayed

On all detail views, Header and Footer sections are available for you to add specific HTML code. These sections have id <MODULE_NAME>DetailsHdr and <MODULE_NAME>DetailsFtr (for example: ContactsDetailsHdr or MeetingsDetailsFtr).
You should make sure you only append html to these headers and footers with one of these hooks

init_view

Example

Add a button on the top of the view

```
QCRM.SetModuleHook('Contacts','init_view',function(){
   $("#ContactsDetailsHdr").append(
      '<button id="MY_ACTION" style="display:none;">DO SOMETHING</button>'
   );
   $("#MY_ACTION").click(function(){DoSomething()});
});
```

before_display

Example

Show or hide the button depending on a field value

```
QCRM.SetModuleHook('Contacts','before_display',function(values){
  if (values.contact_type_c =='')
       $("#MY_ACTION").hide();
  else
       $("#MY_ACTION").show();
});
```

after_display

Example

Show or hide Discount field if Account Type is not "Customer"

```
QCRM.SetModuleHook('Contacts','after_display',function(values){
   if (values.account_type.value !='Customer')
       QCRM.DetailFieldHide('Accounts','discount_amount_c');
   else
       QCRM.DetailFieldShow('Accounts','discount_amount_c');
});
```

Edit View customization

Hooks allow you to customize Edit View, for example, add a change event handler, hide fields conditionally, set a field value depending on another field.

Available hooks are:

- check_edit : executed when an Edit button or menu is clicked. This can prevent a record being edited depending on conditions based on record field values, user role or user security group (see User Management chapter)
- check_delete: executed when a Delete button or menu is clicked. This can prevent a record being deleted depending on conditions based on record field values, user role or user security group (see User Management chapter)
- onchange : Add a change event handler on a field
- init_edit : executed the first time the view is created. This would allow you, for example to add a change event handler on a field
- before_edit : executed once the edit form is displayed (creation and edition)
- before_create : executed for new records once the edit form is displayed
- before_update : executed once the record is retrieved and the edit form is displayed
- check_before_save : executed before the record is saved
- before_save : executed before the record is saved (allow calculated fields)

In these hooks, you can get or set a field value during edition, show or hide a field or access the field id for event handlers with the following functions:

- QCRM.GetFieldValue (module, field)
  Note: for date and datetime fields, undefined is returned when date is not set. If date is set, a Date object is returned
- QCRM.SetFieldValue (module, field, new_value)
  Note: new_value must be a Date object for date and datetime fields
- QCRM.EditFieldShow (module, field)
- QCRM.EditFieldHide (module, field)
- QCRM.GetEditFieldId (module, field)

check_edit

Example

Opportunities in “Closed Won” sales stage can only be modified by users with role management.

```
QCRM.SetModuleHook('Opportunities','check_edit',function(values){
   var ok = true;
   if (values.sales_stage && values.sales_stage.value == 'Closed Won'){
     // we test if sales_stage field exists and its value is Closed Won
     // the field might not exist in values for a new record
     if (!QCRM.UserHasRole ('management')) ok = false;
   }
   return ok;
});
```

check_delete

Example

Accounts with type Customer can never be deleted in the app.

```
QCRM.SetModuleHook('Accounts','check_delete',function(values){
   var ok = true;
   if (values.account_type && values.account_type.value == 'Customer'){
     // field account_type might not exist in values for a new record
     ok = false;
   }
   return ok;
});
```

onchange

Example

Add a change event handler on Account type to hide the Discount field if type is not “Customer” and set its value to 0.

```
QCRM.SetFieldHook('Accounts', 'account_type', 'onchange',function(newvalue){
   if (newvalue=='Customer')
       QCRM.EditFieldShow('Accounts','discount_amount_c');
   else{
       QCRM.EditFieldHide('Accounts','discount_amount_c');
       QCRM.SetFieldValue('Accounts','discount_amount_c',0);
   }
});
```

For date and datetime fields, newvalue is undefined if the field is not set, and a Date object if the field is set.

init_edit

Example

Add a change event handler on Account type to hide the Discount field if type is not “Customer” and set its value to 0.

```
function HandleType(){
   var type = QCRM.GetFieldValue('Accounts','account_type');
   // hide the discount field if type is not Customer
   if (type=='Customer')
       QCRM.EditFieldShow('Accounts','discount_amount_c');
   else{
       QCRM.EditFieldHide('Accounts','discount_amount_c');
       QCRM.SetFieldValue('Accounts','discount_amount_c',0);
   }
}

QCRM.SetModuleHook('Accounts','init_edit',function(){
   // Add change event handler on account type
   $(QCRM.GetEditFieldId('Accounts','account_type')).change(HandleType);
});
```

before_edit (new records and existing records)

This hook is called once the edit view is opened and values are loaded (existing records).

Example

Show or hide the discount field based on current values

```
QCRM.SetModuleHook('Contacts','before_edit',function(values,new_record){
  // parameters :
  // values : name_value_list of record or empty object for a new record
  // new_record : boolean (true when a new record is created)
  HandleType();
});
```

before_create (new records)

This hook is exactly similar to before_edit, but is called only when creating a record

before_update (existing records)

This hook is exactly similar to before_edit, but is called only when updating an existing record

check_before_save

The function should return an empty string when no error is found, or the error message if an error was found.

Example

Make field margin_c required if Opportunity Sales Stage is Won.

```
QCRM.SetModuleHook('Opportunities','check_before_save',function(values){
  // parameters :
  // values : values of record to be saved
       var margin=values.margin_c;
       var sales_stage = values.sales_stage;
       if (sales_stage=='Closed Won' && (margin=='' || margin=='EMPTY')){
              return 'Margin is required for won opportunities';         }

       return '';

});
```

before_save

Example

Set accnt_no field to upper case

```
QCRM.SetModuleHook('Accounts','before_save',function(values){
  // parameters :
  // values : name_value_list of record (as expected by Sugar/SuiteCRM APIs)
  values.acct_no.value  = values.acct_no.value.toUpperCase();
});
```

QCRM.SetFieldReadonly = function (module,field,Bool readonly)

Set a field readonly or not dynamically during edit.

QCRM.SetFieldRequired = function (module,field, Bool required)

Set a field required or not dynamically during edit.
If required is set the label is changed with a red asterisk.

Utilities

Date management

function toDBDate(Date date) : returns date in database format for date type fields

function toDBDateTime(Date date) : returns date in database format for datetime type fields

function fromDBDate(String str) : returns Date object from string in database format for date type fields

function fromDBDateTime(String str) : returns Date object from string in database format for datetime type fields