

Portal v2

Theme documentation

<https://adminlte.io/docs/3.2/>

<https://github.com/ColorlibHQ/AdminLTE>

<https://adminlte.io/themes/v3/>

Webservice REST

Handler: include/Webservices/CustomerPortal.php

Headers:

- Authorization: Basic base64_encode("username:accesskey")
- Portal-Session-Id: the parameter is optional and is returned by the "portal.login". Use in subsequent calls to optimize portal performance.
- Content-Type: application/json

Name	Parameters	Description
portal.info		Allows you to obtain information on the license and logos used by the connected vtenext installation.
portal.login	Body . username: string (*) . password: string (*) . language: string	Allows you to log in to the customer portal by providing the email (username) and password of the contact. The service also returns the accesskey and session id values to be used in subsequent calls as headers (Authorization and Portal-Session-Id).
portal.logout	Headers . Authorization	Allows you to logout the contact from the portal.
portal.send_mail_for_password	Body . email: string (*) . language: string	Allows you to recover the password of a contact.
portal.modules_list	Headers . Authorization Body . language: string (*)	Allows you to obtain a list of modules enabled for the contact's profile.

portal.get_list	Headers . Authorization Body . module: string (*) . language: string . folderid: int . search: array	Allows you to obtain the list of records of a module filtered by the contact profile. The "folderid" parameter is used by the Documents module to get the documents of a specific folder. The "search" parameter allows the paging of the list on the server-side. Ex. <pre>['length' => 50, // Number of records to return 'start' => 0, // Paging offset 'search' => "", // Global search 'search_columns' => [// Search by columns ['index' => 0, // Index of the field 'column' => "", // Name of the field 'search' => "", // Value of the field]], 'ordering' => [// Ordering ['index' => 0, // Index of the field 'column' => "", // Name of the field 'dir' => "", // asc/desc]],]</pre> Note: 1. All searches are performed ONLY on the columns of the list. 2. Column search is performed in "LIKE" mode. 3. Sorting can only be done on one column.
portal.get_blocks	Headers . Authorization Body . module: string (*) . language: string . mode: string (edit, create, detail, list) (*) . app_data: array	Allows you to obtain the list of the blocks and the related fields of a module filtered by the contact profile. The "app_data" parameter represents the record being edited (array with field name and value) and is used by the SDK views to establish the visibility of the fields. This parameter will also be used in the future for managing conditional fields.
portal.get_record	Headers . Authorization Body . module: string (*) . id: int (*)	If the contact has visibility permissions for the indicated record, it allows you to obtain its data.

portal.save_record	Headers . Authorization Body . module: string (*) . id: int (*) . values: encoded (*)	If the contact has write permissions for the indicated record, it allows its saving.
portal.delete_record	Headers . Authorization Body . module: string (*) . id: int (*)	If the contact has delete permissions for the indicated record, it allows its deletion.
portal.write_ticket_comment	Headers . Authorization Body . id: int (*) . comment: string (*)	Allows you to write a comment within the indicated ticket.
portal.get_attachments	Headers . Authorization Body . id: int (*)	Allows you to obtain the list of documents of the specified record filtered by the visibility of the contact profile.
portal.download_attachment	Headers . Authorization Body . relid: int (*) . docid: int	If the contact has permission to view the attachment (relid), it allows its download.
portal.upload_attachment	Headers . Authorization Body . relid: int (*) . title: string (*) File	Allows the uploading of an attachment related to the specified record (relid). You can indicate the name (title) of the document that will be generated.
portal.provide_confidential_info	Headers . Authorization Body . id: int (*) . comments: string . data: string (*) . request_commentid: int (*)	Allows you to respond to a request for confidential information. The "id" parameter indicates the ticket id, the "comments" parameter indicates the unencrypted comment, the "data" parameter indicates the confidential response, and the "request_commentid" parameter indicates the id of the comment to which the confidential response should be provided .
portal.get_home_widgets	Headers . Authorization Body . language: string	Allows you to obtain the widgets configured in the profile associated with the contact.
portal.save_authenticate_cookie	Headers . Authorization Body . contactid: int (*)	Provides an hash to be stored in a cookie to remember the contact's login.

portal.check_authenticate_cookie	Headers . Authorization Body . contactid: int (*) . hash: string (*)	Allows you to verify the hash used to remember the contact's login.
portal.change_password	Headers . Authorization Body . username: string (*) . old_password: string (*) . password: string (*) . language: string	Allows you to change the contact's password.

Register a new REST webservice

Create a new file and execute it (e.g. plugins/script/script.php).

```
<?php
require('../..../config.inc.php');
chdir($root_directory);
require_once('include/utils/utils.php');
require_once('vtlib/Vtecrm/Module.php');
$Vtiger_Utills_Log = true;
global $adb, $table_prefix;
VteSession::start();

SDK::setClass('CustomerPortalRestApi', 'CustomerPortalRestApi2',
'modules/SDK/src/CustomerPortalRestApi2.php');

$parameters = ['param1' => 'string', 'param2' => 'encoded', 'param3' => 'encoded'];
$perm = 'read'; // read, write, readwrite
SDK::setRestOperation('portal.foo', 'modules/SDK/src/CustomerPortalRestApi2.php',
'CustomerPortalRestApi2.foo', $parameters, $perm);
```

Create a new file that contains the CustomerPortalRestApi2 class (e.g. modules/SDK/src/CustomerPortalRestApi2.php).

```
<?php

require_once('include/Webservices/CustomerPortal.php');

class CustomerPortalRestApi2 extends CustomerPortalRestApi {
```

```

public function foo($param1, $param2, $param3) {
    $data = [1, 2, 3, 4, 5];
    // ...
    return $data;
}

}

```

Extend an existing REST webservice

Create a new file and execute it (e.g. plugins/script/script.php).

```

<?php
require('../../config.inc.php');
chdir($root_directory);
require_once('include/utils/utils.php');
require_once('vtlib/Vtecrm/Module.php');
$Vtiger_Utills_Log = true;
global $adb, $table_prefix;
VteSession::start();

SDK::setClass('CustomerPortalRestApi', 'CustomerPortalRestApi2',
'modules/SDK/src/CustomerPortalRestApi2.php');

```

Create a new file that contains the CustomerPortalRestApi2 class (e.g. modules/SDK/src/CustomerPortalRestApi2.php).

```

<?php

require_once('include/Webservices/CustomerPortal.php');

class CustomerPortalRestApi2 extends CustomerPortalRestApi {

    public function get_list($module, $language, $folderid = 0, $search = []) {
        $ret = parent::get_list($module, $language, $folderid, $search);
        // your code here ...
        return $ret;
    }

}

```

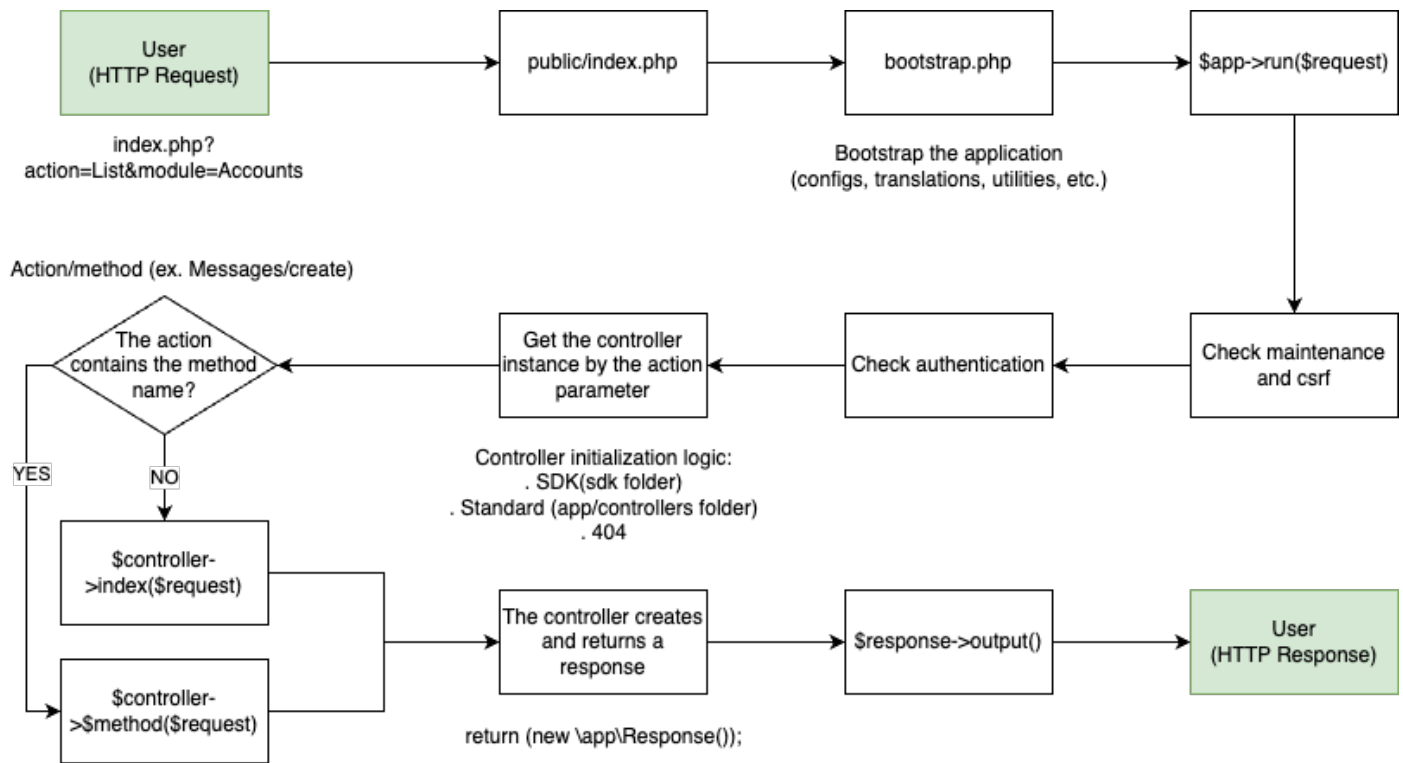
Structure of main folders/files

The portal folder is located at: **VTE_ROOT/portal/v2**

The main folders/files of the new portal are:

Folder/file name	Description
app	Contains the files and the logic of the portal.
app/controllers	Contains the files that manage the portal's default actions (e.g. Login, Logout, Edit, Detail, etc.).
app/fields	Contains the files that manage the portal module fields.
app/modules	Contains custom logic of some vtenext modules (e.g. Documents, Processes, HelpDesk, etc.).
app/PortalModule.php	Class that manages the actions of the portal modules (e.g. Create, Detail, Edit, etc.). The class can be extended to change the default behaviors of a module.
config	Contains the portal configuration files.
public	Contains the portal's public files (e.g. index.php, css, javascript, images, etc.).
resources	Contains the resources used by the portal such as translation files (lang) and templates (templates).
sdk	The folder is used to insert new customizations for the customer.
storage	Contains temporary files (e.g. cache, logs, etc.).
vendor	Contains the external libraries used by the portal.

Request lifecycle



Portal configuration

The portal configuration file is located in "config/portal.config.php".

To **overwrite the parameters** the "config/sdk.config.php" file must be used otherwise, in case of updating the version of vtenext, the changes could be lost.

How to move the Business Portal to another host/folder

Update the "portal_url", "vte_url" and "csrf_secret" parameters in the "config/sdk.config.php" file.

Also change the "default_timezone" parameter if the host has a different timezone from the one in which the vte is located. Change the "portal.url" prop to the link pointing to the v2 folder. e.g. <https://ticket.vtenext.com/v2>

Here is the list of parameters supported by the new portal:

Parameter	Type	Default	Description
-----------	------	---------	-------------

portal_url	String	\$PORTAL_URL (config.inc.php)	Indicates the URL of the customer portal. If the portal folder is inside the vtenext root directory, the variable will be set to the value of the \$PORTAL_URL variable set in the config.inc.php file.
vte_url	String	\$site_URL (config.inc.php)	Indicates the URL of vtenext and is used to obtain its data via the rest API. If the portal folder is inside the vtenext root directory, the variable will be set to the value of the \$site_URL variable set in the config.inc.php file.
default_language	String	it_it	Indicates the default language used in the customer portal. The value can be replaced with a supported language (see "languages" parameter).
languages	Array	['en_us' => 'US English', 'it_it' => 'IT Italiano']	Indicates the languages supported in the customer portal. To add a new language you need to create a new file in the resources/lang folder.
production	Bool	false	This configuration indicates whether errors should be displayed or not. If the environment is production, the value will be set to true to disable error display. If the environment is development, the value will be set to false to allow errors to appear.
default_module	String		Indicates the default module to load after logging into the portal. The value can be replaced with the name of the desired module (must be enabled from profile). If the parameter is empty, the portal home will be loaded.
favicon	String	assets/img/VTENEXT_favico n.ico	Indicates the path of the favicon. The value is relative to the public folder.

login_logo	String	assets/img/VTENEXT_login.png	Indicates the path of the logo to load on the login page. The value is relative to the public folder. [UPDATE] The logo must be loaded in vtenext settings > Logos.
login_background	String		Indicates the path of the background to load on the login page. The value is relative to the public folder.
header_logo_sm	String	assets/img/VTENEXT_toggle.png	Indicates the path of the icon to load in the minimized sidebar. The value is relative to the public folder. [UPDATE] The icon must be loaded in vtenext settings > Logos.
header_logo_lg	String	assets/img/VTENEXT_header.png	Indicates the path of the icon to load in the expanded sidebar. The value is relative to the public folder. [UPDATE] The icon must be loaded in vtenext settings > Logos.
helpdesk_logo	String	assets/img/helpdesk.png	Indicates the logo used to display the response given by customer support. The value is relative to the public folder.
sidebar_theme	String	sidebar-dark-primary	Indicates the class of the main sidebar. It can have a "dark" or "light" brightness. It can also have a color variant, such as "primary", "success", "warning", "info", "danger".
enable_sidebar_search	Bool	false	Enable/disable the search bar in the main sidebar.
csrf_secret	String	\$csrf_secret (config.inc.php)	Indicates the secret key used to generate a csrf token. If the portal folder is inside the vtenext root directory, the variable will be set to the value of the \$csrf_secret variable set in the config.inc.php file.

upload_dir	String		Indicates the name of the folder used for uploading files.
browser_title_prefix	String		Indicates the prefix label to use for the browser title. The value can be replaced with the desired label.
browser_title_suffix	String	customer_portal	Indicates the suffix label to use for the browser title. The value can be replaced with the desired label.
remember_cookie_name	String	portal_login_hash	Indicates the name of the cookie used to remember user authentication.
login_expire_time	Int	2592000 (one month)	Indicates the expiration of the cookie used to remember user authentication. The value can be replaced with the desired number of seconds.
default_timezone	String	Europe/Rome	Indicates the default time zone used in the customer portal. This configuration must be the same as the \$default_timezone variable set in the vtenext config.inc.php file.
module_icons	Array	[]	With this configuration you can override the default icons used for modules enabled in the customer portal. The default form icons are found in app/layouts/PortalLayout.php. The names of the icons can be found here https://fonts.google.com/icons .
sdk_languages	Array	[]	With this configuration you can add new labels or modify existing ones. You need to create a new file in the sdk folder.
sdk_global_php	Array	[]	With this configuration you can add php files to load on each page (they should contain classes/functions). You need to create a new file in the sdk folder.

sdk_global_js	Array	[]	With this configuration you can add js to load globally. You need to create a new file in the public/assets/sdk folder.
sdk_module_js	Array	[]	With this configuration you can add js to load for a specific module. You need to create a new file in the public/assets/sdk folder.
sdk_global_css	Array	[]	With this configuration you can add css to load globally. You need to create a new file in the public/assets/sdk folder.
sdk_controllers	Array	[]	With this configuration you can add custom actions ("action" field in the URL). The array represents an association between the action name and the file that contains the controller class to handle the request. You need to create a new file in the sdk folder.
sdk_module	Array	[]	With this configuration you can add customizations on a specific module. The array represents an association between the module name and the file that contains the module's extended class. You need to create a new file in the sdk folder.
sdk_menu	Array	[]	With this configuration you can add custom menu items in the sidebar. You need to create a new file in the sdk folder.

API Reference - Main classes, methods, functions and variables

\app\Request

Method	Arguments	Description
get()	\$keys = null, \$purify = false	Allows you to get one or more parameters from the global variable \$_GET. If the second argument is set to "true", the parameters are purified through the HTML Purifier library. Example: <pre>\$request->get('foo'); \$request->get(['foo', 'bar']); \$request->get('foo', true);</pre>
post()	\$keys = null, \$purify = false	Allows you to get one or more parameters from the global variable \$_POST. If the second argument is set to "true", the parameters are purified through the HTML Purifier library. Example: <pre>\$request->post('foo'); \$request->post(['foo', 'bar']); \$request->post('foo', true);</pre>
files()	\$key	Allows you to get files uploaded via the HTTP POST method and organized via the global variable \$_FILES. Example: <pre>\$request->files('attachments');</pre>
cookie()	\$keys = null, \$purify = false	Allows you to obtain one or more parameters from the global variable \$_COOKIE. If the second argument is set to "true", the parameters are purified through the HTML Purifier library. Example: <pre>\$request->cookie('foo');</pre>

server()	\$keys = null, \$purify = false	Allows you to get one or more parameters from the global variable \$_SERVER. If the second argument is set to "true", the parameters are purified through the HTML Purifier library. Example: \$request->server('');
isGet()		Returns true if the request method is GET.
isPost()		Returns true if the request method is POST.
isAjax()		Returns true if the request is AJAX.
purify()	\$input	Purify the \$input variable through the HTML Purifier library.

\app\Response

Method	Arguments	Description
__construct()	\$content = "", \$statusCode = 200, \$headers = []	Initializes a new \app\Response() object with the response content (\$content), return code (\$statusCode), and default headers (\$headers).
setContent()	\$content	Set the content of the response.
setStatusCode()	\$statusCode	Set the response return code.
setHeader()	\$header, \$replace = true	Set a new header in the response. With the second argument it is possible to indicate whether or not the header must replace a previous header already set.
setMimeType()	\$mimeType = 'text/html'	Sets the response content mime.
json()	\$data	Set response content with \$data converted to JSON format and 'application/json' mime.
redirect()	\$page	Performs a redirect to \$page.
downloadFile()	\$fullpath	Allows downloading of a file located in \$fullpath.
output()		Outputs the content, response code, and headers you set.

\app\Session

Method	Arguments	Description
set()	\$key, \$value = ''	Set the value \$value with key \$key in \$_SESSION.
get()	\$key	Allows you to get the value with key \$key from \$_SESSION.
flash()	\$key	Allows you to get the value with key \$key from \$_SESSION. Next, the \$key will be deleted from \$_SESSION.
remove()	\$key	Delete the \$key from \$_SESSION.
hasKey()	\$key	Returns true if \$key exists in \$_SESSION.
setMulti()	\$keys	Allows you to write multiple values to \$_SESSION.
removeMulti()	\$keys	Delete multiple values in \$_SESSION.
append()	\$key, \$value = ''	Set the \$key as an array in \$_SESSION and the value \$value is added to it.

\app\Config

Method	Arguments	Description
has()	\$key	Returns true if \$key exists in the global configuration.
get()	\$key	Allows you to get the value with key \$key from the global configuration.
set()	\$key, \$value	Set the value \$value with key \$key in the global configuration.
getAll()		It allows you to obtain a key-value list with all the global configuration.
setMulti()	\$values	Allows you to write multiple values into the global configuration.
clear()	\$key	Delete the \$key from the global configuration.
clearAll()		Delete all values from the global configuration.

\app\PortalModule

Variable	Default	Description
\$hasComments	false	Indicates whether the module supports comments.
\$hasAttachments	false	Indicates whether the module supports attachments.
\$enableEdit	true	Indicates whether the module can be modified (edit mode).
\$formColumns	3	Indicates the number of columns to use for displaying fields in Create, Edit and Detail.
\$listTemplate	List.tpl	Indicates the template used for displaying a list (action List).
\$referenceListTemplate	sections/ReferenceList.tpl	Indicates the template used for displaying a related list (uitype 10).
\$detailTemplate	Detail.tpl	Indicates the template used to display the detail of a record (action Detail).
\$editTemplate	Edit.tpl	Indicates the template used to display the edit of a record (action Edit).
\$notAuthorizedTemplate	PageNotAuthorized.tpl	Indicates the template used to display a permission error (e.g. record not found or permission errors).

Method	Arguments	Description
__construct()	\$module	\$module indicates the name of the module to obtain an instance of the class. If the module has been extended via SDK then an instance of the extended class will be returned.
prepareList()	\$viewer, \$request	Method used for displaying a list (action List).
postProcessList()	\$viewer, \$request	This method can be used by extended classes to insert/modify data set in prepareList().
prepareEdit()	\$viewer, \$request	Method used to display the edit of a record (action Edit).
postProcessEdit()	\$viewer, \$request	This method can be used by extended classes to insert/modify the data set in prepareEdit().
prepareDetail()	\$viewer, \$request	Method used to display the detail of a record (action Detail).

postProcessDetail()	\$viewer, \$request	This method can be used by extended classes to insert/modify data set in prepareDetail().
saveRecord()	\$request	Method used to save a record (action Save).
postProcessSaveValues()	\$request, &\$values	This method can be used by extended classes to insert/modify data set in saveRecord().
isPermitted()	\$module, \$action = self::ACTION_LIST, \$recordValues = []	Returns if the contact has permission to perform a certain action. List of supported actions: . ACTION_LIST . ACTION_CREATE . ACTION_EDIT . ACTION_DETAIL . ACTION_DELETE . ACTION_SAVE . ACTION_ADD_COMMENTS . ACTION_UPLOAD_ATTACHMENTS . ACTION_CHANGE_PWD . ACTION_SOLVE_TICKET

\app\clients\PortalRestClient

Method	Arguments	Description
get()	\$restName, \$queryParameters = [], \$headers = []	Allows you to perform a GET request to vtenext.
post()	\$restName, \$queryParameters = [], \$bodyParameters = [], \$headers = []	Allows you to perform a POST request to vtenext.
postMultipart()	\$restName, \$queryParameters = [], \$bodyParameters = [], \$fileParameters = [], \$headers = []	Allows you to perform a multipart POST request to vtenext.
patch()	\$restName, \$queryParameters = [], \$bodyParameters = [], \$headers = []	Allows you to execute a PATCH request to vtenext.
delete()	\$restName, \$queryParameters = [], \$headers = []	Allows you to execute a DELETE request to vtenext.
postDownload()	\$restName, \$queryParameters = [], \$bodyParameters = [], \$headers = []	Allows you to perform a POST request to download a vtenext file in stream mode.

Helpers

Function	Arguments	Description
----------	-----------	-------------

preprint()	\$var	print_r formatted with <pre> tag
predump()	\$var	var_dump formatted with <pre> tags
encodeForHtml()	\$value, \$charset = 'UTF-8'	Encode the \$value value to be inserted into an HTML page.
encodeForHtmlAttr()	\$value, \$enclosing = ''	Encode the \$value value to be placed inside an HTML tag attribute.
encodeForJs()	\$value, \$enclosing = ''	Encode the \$value value to be placed inside a javascript <script>.
htmlAttr()	\$attributes	Encode a list of attributes to be placed inside an HTML tag.
config()	\$key	Allows you to get the value with key \$key from the global configuration.
trans()	\$key, \$args = []	Translate the label \$key.
listUrl()	\$module, \$extraParams = []	Generate a link to open a list.
createUrl()	\$module, \$extraParams = []	Generates a link to open the creation of a record.
createDocUrl()	\$module, \$folderId, \$extraParams = []	Generates a link to open the creation of a document.
detailUrl()	\$module, \$record, \$extraParams = []	Generates a link to open the detail of a record.
editUrl()	\$module, \$record, \$extraParams = []	Generates a link to open the editing of a record.
downloadUrl()	\$record, \$documentId, \$extraParams = []	Generates a link to download a document.
docFolderUrl()	\$module, \$folderId, \$extraParams = []	Generates a link to open a specific folder in the documents module.
returnUrl()	\$request	Generate a return link (e.g. "Cancel" action).
portalLanguage()		Returns the language used in the portal.
setPortalLanguage()	\$language	Set the \$language in the portal.
getBrowserTitle()	\$title	Returns the title to be set in an HTML page of the portal.
getModuleLabel()	\$module	Returns the translation of the module \$module.
getSingleModuleLabel()	\$module	Returns the singular translation of the module \$module.

getMaxUploadSize()		Returns the maximum upload size in the portal.
setPortalCookie()	\$name, \$value = "", \$expires_or_options, \$httponly = false	Set a cookie in the portal.
unsetPortalCookie()	\$name, \$httponly = false	Removes a cookie from the portal.
csrfToken()		Returns the csrf token.
csrfInputName()		Returns the name of the input for sending the csrf token.
flashPortalError()	\$error	Allows you to display a timed error message.
flashPortalMessage()	\$message	Allows you to display a timed message.
customerId()		Returns the ID of the authenticated contact.
customerEmail()		Returns the email of the authenticated contact.
customerUsername()		Returns the name and surname of the authenticated contact.
resourcever()	\$filename	Allows the versioning of css and javascript files.
basePath()	\$path = ""	Returns the path to the portal base folder. If \$path is specified, a path is created and returned starting from the base folder.
appPath()	\$path = ""	Returns the path to the portal's "app" folder. If \$path is specified, a path is created and returned starting from the "app" folder.
configPath()	\$path = ""	Returns the path to the portal's "config" folder. If \$path is specified, a path is created and returned starting from the "config" folder.
resourcesPath()	\$path = ""	Returns the path to the portal's "resources" folder. If \$path is specified, a path is created and returned starting from the "resources" folder.
langPath()	only \$	Returns the path to the indicated language file \$lang.

storagePath()	\$path = ""	Returns the path to the portal's "storage" folder. If \$path is specified, a path is created and returned starting from the "storage" folder.
publicPath()	\$path = ""	Returns the path to the portal's "public" folder. If \$path is specified, a path is created and returned starting from the "public" folder.
vtePath()	\$path = ""	Returns the path to the vtenext folder. If \$path is indicated, a path is created and returned starting from the vtenext folder.
sdkPath()	\$path = ""	Returns the path to the portal's "sdk" folder. If \$path is indicated, a path is created and returned starting from the "sdk" folder.
sdkAssetsPath()	\$path = ""	Returns the path to the portal's "public/assets/sdk" folder. If \$path is specified, a path is created and returned starting from the "public/assets/sdk" folder.
publicRelPath()	\$assetPath	Returns a relative path starting from the "public" folder.

Examples

SDK view

Here is an example of how to change the visibility of portal fields via view SDK.

```
<?php
require('../../config.inc.php');
chdir($root_directory);
require_once('include/utils/utils.php');
require_once('vtlib/Vtecrm/Module.php');
$Vtiger_Utills_Log = true;
global $adb, $table_prefix;
VteSession::start();

$module = '';
$src = '';
$mode = 'constrain';
$success = 'continue';
```

```
SDK::addView($module, $src, $mode, $success);
```

```
<?php

global $sdk_mode, $table_prefix;

switch ($sdk_mode) {
    case 'portal.create':
        $readonly = 100;
        $success = true;
        break;
    case 'portal.edit':
    case 'portal.detail':
        if ($col_fields['field3'] !== 'Open') {
            $readonly = 99;
            $success = true;
        }
        if (in_array($fieldname, ['field1', 'field2'])) {
            $readonly = 100;
            $success = true;
        }
        break;
}
```

Creating a new controller

Here is an example of how to create a new controller to manage a custom action.

- Edit "config/sdk.config.php" to insert a new controller

```
return [
    'sdk_controllers' => [
        'SampleVte' => 'controllers/SampleVteController.php',
    ]
];
```

- Implement the SampleVteController class in "sdk/controllers/SampleVteController.php"

```
<?php
```

```

class SampleVteController extends \app\controllers\BaseController {
    []
    []public function index($request) {
        []return $this->displaySomething($request);
    }

    []protected function displaySomething($request) {
        []$parameter1 = $request->get('parameter1', true);
        []$parameter2 = $request->get('parameter2', true);

        []$this->viewer->assign('PARAMETER1', $parameter1);
        []$this->viewer->assign('PARAMETER2', $parameter2);

        []$layout = \app\LayoutFactory::getPortalLayout($this->viewer, $this->client, $request);
        []$output = $this->fetchWithLayout('sdk/SampleVte.tpl', $layout);

        []return new \app\Response($output);
    }
    []
}

```

- Create a new template in "resources/templates/sdk/SampleVte.tpl"

```

{extends file='layouts/PortalLayout.tpl'}

{block name=content}
[]<h1>Sample Vte</h1>
{/block}

```

- Modify "config/sdk.config.php" indicating the sdk file for inserting the new entries in the side menu

```

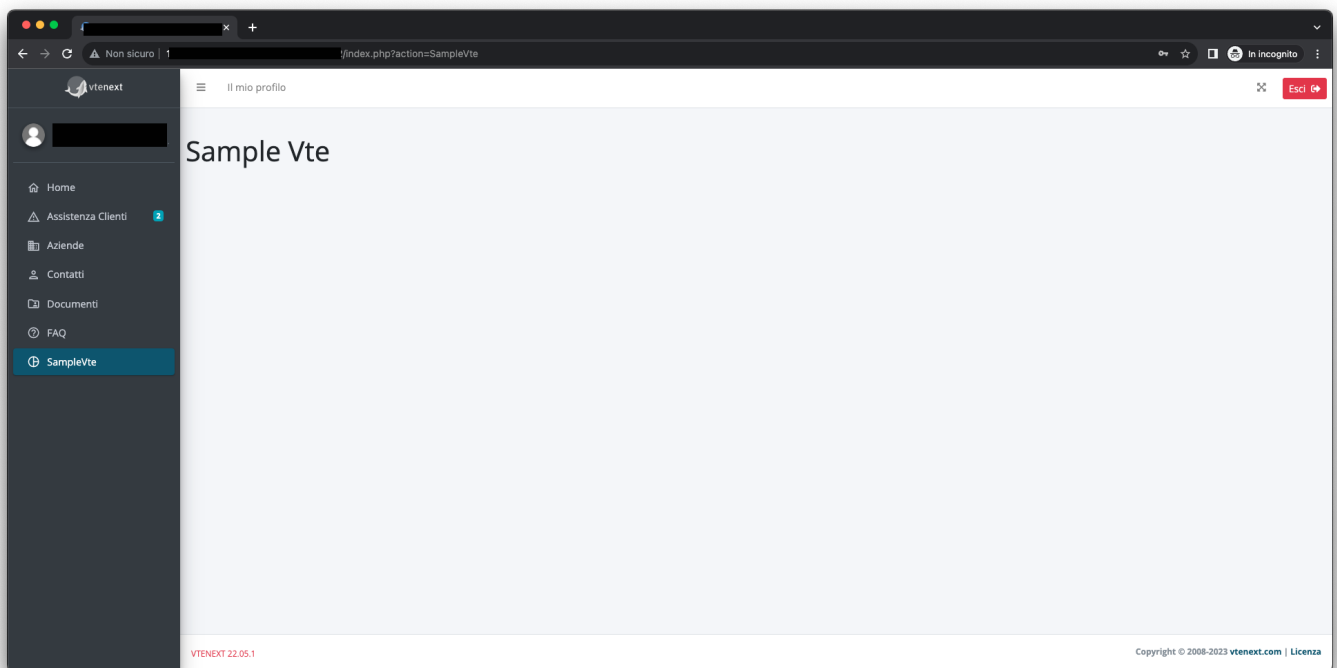
return [
    []'sdk_controllers' => [
        []'SampleVte' => 'controllers/SampleVteController.php',
    ],
    []'sdk_menu' => [
        []'samplevte-menu.php',
    ]
];

```

- Edit the "sdk/samplevte-menu.php" file

```
<?php

return [
    [
        'text' => trans('SampleVte'),
        'active' => false,
        'prefix' => [
            'type' => 'icon',
            'icon_style' => 'material',
            'icon_name' => 'pie_chart',
        ],
        'action' => [
            'type' => 'link',
            'link_href' => "index.php?action=SampleVte",
        ],
    ],
];
```



Extending a module

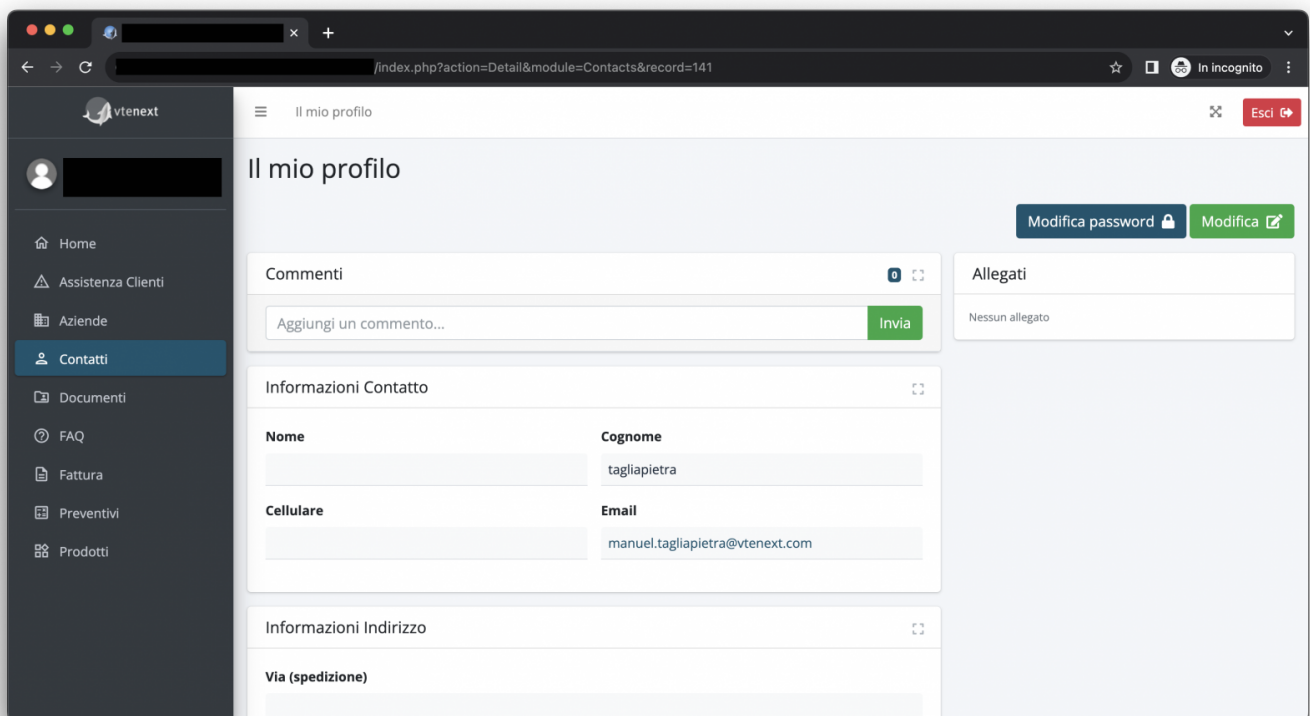
Here is an example of how to extend the functionality of a module.

- Edit "config/sdk.config.php"

```
return [  
    ['sdk_module' => [  
        ['Contacts' => 'modules/ContactsModule.php'],  
    ]  
];
```

- Implement the ContactsModule class in "sdk/modules/ContactsModule.php"

```
<?php  
  
class ContactsModule extends \app\PortalModule {  
    [  
    public $hasComments = true;  
  
    public $formColumns = 2;  
  
    public function canAddComments() {  
        return true;  
    }  
}
```



Revision #12

Created 2023-09-15 08:30:50 UTC by manuel.tagliapietra

Updated 2025-03-04 16:03:44 UTC by manuel.tagliapietra