

Using AI

Tips on how to use LLM to help writing/refactoring vite code

- [Skill - Review SDK files](#)
- [Skill - Review plugin](#)

Skill - Review SDK files

Skill to (try to) port all custom files to vtenext 26 and make them compatible with vtenext 26.

Install to Codex

Create the folder `review-sdk-26` in the skills folder of your IDE example: `[...]/.codex/skills/` with the structure:

`review-sdk-26/`

```
|— SKILL.md
|— agents/
|   |— openai.yaml
```

SKILL.md

```
---
name: review-sdk-26
description: Revise the SDK code to be compatible with VTENEXT version 26.*
---

# review-sdk-26

Review and update PHP files (>=7.x) to make them:
* Compatible with PHP 8.3
* Aligned with the new project guidelines
* More robust, typed, and modern

You are working on existing VTENEXT application code. DO NOT rewrite everything from scratch.
Keep the original logic, improving it only where necessary.

The project's root directory is specified in the config.inc.php file in `$root_directory`. If
that directory is not accessible, use the directory containing the file config.inc.php.

## When to use

This skill should be used after the update to VTENEXT 26.* to ensure that all custom PHP files
```

are compatible with PHP 8.3 and follow the new project guidelines.

Input

- * List of PHP files, if they are not provided execute ``tools/skills/review-sdk-26/get_php_customizations.php`` to get the list of files to scan and review. The list of files should be provided in batches of 10 if there are too many files to review at once. Do not analyze, process or propose changes to files that are not in the list obtained.

Instructions

1. Prepare the environment

- * Create folder `modules/SDK/src/review_sdk_26`, if already exists empty it
- * If there are too many files, run the skill on 10 files at a time (batch)
- * If you are running a batch, do not empty the folder
- * Also copies unpatched files to the `review_sdk_26` folder to maintain the same structure and allow manual review of all files
- * If you find any hardcoded credentials or sensitive data while analyzing the code, stop everything and let me know to move this data to a separate configuration file

2. PHP 8.3 Compatibility Analysis

- * Fix weak comparisons as described
[here](<https://usermanual.vtenext.com/books/developers/page/weak-comparisons>)
- * Add type casts where necessary to functions that require a string as parameter, for example ``trim()`` ``strpos()`` and ``stripos()``
- * Do not replace functions if the existing ones work on PHP 8.3
- * For empty string comparisons, replace with the ``empty()`` function
- * For comparisons with ``$mode`` or ``$sdk_mode``, the creation mode is usually checked and the empty string comparison is fine. ``=== ````

example

old code:

```
```php
if ($recordid == '') {
 ...
}
```

new one:

```
```php
if (empty($recordid)) {
```
```

### ### 3. Refactoring

#### #### RequestHandler

- \* As described [here](<https://usermanual.vtenext.com/books/developers/page/requesthandler>), access to the superglobal variables `$_REQUEST`, `$_GET`, and `$_POST` is no longer allowed.
- \* Replace reading their values with the `RH::r`, `RH::g`, and `RH::p` methods of the `RequestHandler` class found in `include/utils/RequestHandler.php`.
- \* When writing to superglobals, use the `RH::push_*` and `RH::pop_*` methods as described in the link above.
- \* Remove calls to `vtlib_purify()`; RequestHandler already handles them internally.

#### #### Database best practices and escaping

- \* Apply the new best practices as described [here](<https://usermanual.vtenext.com/books/developers/page/escaping-rules>)
- \* In particular, replace `query_result` with `query_result_no_html` and `fetchByAssoc(...)` with `fetchByAssoc(..., -1, false)`
- \* If HTML is generated in PHP and assigned to a smarty variable, use the `Vtenext\Types\HtmlString` class to enclose the HTML string
- \* In custom reports the method `getSDKBlock()` should be return a `Vtenext\Types\HtmlString` object instead of a string

#### #### Other substitutions

- \* Remove include/require of the file `Smarty_setup.php` that no longer exists
- \* Replace references to the `vtigerCRM_Smarty` class with `VteSmarty`
- \* Replace `session_start()` con `VteSession::start()`
- \* For reading and writing in session use the methods of the `VteSession` class found in `include/VteSession.php`

### ### 4. Coding style

- \* Apply the guidelines described [here](<https://usermanual.vtenext.com/books/developers/page/coding-style>)
- \* Add license header to the top of the file or update it if already present as described in

the link above

- \* Remove the closing tag `?>` at the end of PHP files

### ### 5. Other examples

If you are modifying a method called `get_dependents_list` or `get_related_list` and towards the end there is a piece of code similar to ``$return_value['CUSTOM_BUTTON'] = $button;``, ensure that the `$button` variable is of type ``\Vtenext\Types\HtmlString``, for example in this way:

```
...
```

```
$return_value['CUSTOM_BUTTON'] = new \Vtenext\Types\HtmlString($button);
```

```
...
```

### ### 6. Important Rules

- \* Do NOT change the functional behavior
- \* Do NOT introduce external dependencies
- \* Maintain backward compatibility if possible
- \* Clear code > "smart" code
- \* Do NOT directly modify the original files
- \* Do NOT return complete updated code
- \* Perform a final check on the files with ``php -l`` or with ``php8.3 -l`` if the former is not at version 8.3
- \* When you have finished analyzing all the files suggest to re-check the files copied into the folder

### ## Output

- \* List modified files
- \* Notify the user that modified files will be found in that folder so they can manually verify them and overwrite the originals after their review

## openai.yaml

```
interface:
 display_name: "Review SDK files"
 short_description: "Review the code of SDK customizations and make it compatible with
vtenext 26"
 default_prompt: "Use $review-sdk-26 to review code of SDK customizations and make it
compatible with vtenext 26."
```

# Prompt

Select the skill by digit \$review-sdk... and indicate the path to the folder.  
example:

```
[$review-sdk-26]([...]/.codex/skills/review-sdk-26/SKILL.md) in the folder [vtenext folder].
The skill contains links to web pages with details and usage examples; read those pages.
```

# Skill - Review plugin

Skill to review the code of a specific folder (plugin) and make it compatible with vtenext 26.

## Install to Codex

Create the folder `review-plugin-26` in the skills folder of your IDE example: `[...]/.codex/skills/` with the structure:

review-plugin-26/

```
|— SKILL.md
|— agents/
| |— openai.yaml
```

## SKILL.md

```

name: review-plugin-26
description: Revise the php code to be compatible with VTENEXT version 26.*

review-plugin-26

Review and update PHP files (>=7.x) to make them:
* Compatible with PHP 8.3
* Aligned with the new project guidelines
* More robust, typed, and modern

You are working on a plugin for VTENEXT application which needs to be reviewed to the new
version. DO NOT rewrite everything from scratch.
Keep the original logic, improving it only where necessary.

Instructions

1. Prepare the environment
```

- \* Duplicate the folder
- \* If there are too many files, run the skill on 10 files at a time (batch)
- \* If you are running a batch, do not empty the folder
- \* Also copies unpatched files to the new folder to maintain the same structure and allow manual review of all files
- \* If you find any hardcoded credentials or sensitive data while analyzing the code, stop everything and let me know to move this data to a separate configuration file

### ### 2. PHP 8.3 Compatibility Analysis

- \* Fix weak comparisons as described  
[here](https://usermanual.vtenext.com/books/developers/page/weak-comparisons)
- \* Add type casts where necessary to functions that require a string as parameter, for example ``trim()``, ``strpos()`` and ``stripos()``
- \* Do not replace functions if the existing ones work on PHP 8.3
- \* For empty string comparisons, replace with the ``empty()`` function
- \* For comparisons with ``$mode`` or ``$sdk_mode``, the creation mode is usually checked and the empty string comparison is fine. ``=== ````

#### #### example

old code:

```
```php
if ($recordid == '') {
...

```

new one:

```
```php
if (empty($recordid)) {
...

```

### ### 3. Refactoring

#### #### RequestHandler

- \* As described [here](https://usermanual.vtenext.com/books/developers/page/requesthandler), access to the superglobal variables ``$_REQUEST``, ``$_GET``, and ``$_POST`` is no longer allowed.
- \* Replace reading their values `[]` with the ``RH::r``, ``RH::g``, and ``RH::p`` methods of the ``RequestHandler`` class found in `include/utils/RequestHandler.php`.
- \* When writing to superglobals, use the ``RH::push_*`` and ``RH::pop_*`` methods as described in

the link above.

- \* Remove calls to ``vtlib_purify()``; RequestHandler already handles them internally.

#### #### Database best practices and escaping

- \* Apply the new best practices as described

[here](https://usermanual.vtenext.com/books/developers/page/escaping-rules)

- \* In particular, replace ``query_result`` with ``query_result_no_html`` and ``fetchByAssoc(...)`` with ``fetchByAssoc(..., -1, false)``

- \* If HTML is generated in PHP and assigned to a smarty variable, use the

``\Vtenext\Types\HtmlString`` class to enclose the HTML string

- \* In custom reports the method ``getSDKBlock()`` should be return a ``\Vtenext\Types\HtmlString`` object instead of a string

#### #### Other substitutions

- \* Remove include/require of the file ``Smarty_setup.php`` that no longer exists

- \* Replace references to the ``vtigerCRM_Smarty`` class with ``VteSmarty``

- \* Replace ``session_start()`` con ``VteSession::start()``

- \* For reading and writing in session use the methods of the ``VteSession`` class found in ``include/VteSession.php``

#### ### 4. Coding style

- \* Apply the guidelines described

[here](https://usermanual.vtenext.com/books/developers/page/coding-style)

- \* Add license header to the top of the file or update it if already present as described in the link above

- \* Remove the closing tag ``?>`` at the end of PHP files

#### ### 5. Other examples

If you are modifying a method called `get_dependents_list` or `get_related_list` and towards the end there is a piece of code similar to ``$return_value['CUSTOM_BUTTON'] = $button;``, ensure that the `$button` variable is of type ``\Vtenext\Types\HtmlString``, for example in this way:

```
...
```

```
$return_value['CUSTOM_BUTTON'] = new \Vtenext\Types\HtmlString($button);
```

```
...
```

#### ### 6. Important Rules

- \* Do NOT change the functional behavior

```
* Do NOT introduce external dependencies
* Maintain backward compatibility if possible
* Clear code > "smart" code
* Do NOT directly modify the original files
* Do NOT return complete updated code
* Perform a final check on the files with `php -l` or with `php8.3 -l` if the former is not at
version 8.3
* When you have finished analyzing all the files suggest to re-check the files copied into the
folder
```

```
Output
```

```
* List modified files
* Notify the user that modified files will be found in that folder so they can manually verify
them and overwrite the originals after their review
```

## openai.yaml

```
interface:
 display_name: "Review Plugin"
 short_description: "Review the code of a specific folder (plugin) and make it compatible
with vtenext 26"
 default_prompt: "Use $review-plugin-26 to review code of a specific folder (plugin) and make
it compatible with vtenext 26."
```

## Prompt

Select the skill by digit \$review-plugin... and indicate the path to the folder.

example:

```
[$review-plugin-26]([...]/.codex/skills/review-plugin-26/SKILL.md) on the folder [your
folder].
The skill contains links to web pages with details and usage examples; read those pages.
```