Get started with WProofreader Cloud (autoSearch)

This guide outlines the main steps that you need to follow in order to integrate and configure WProofreader with the **auto-detecting functionality**. All the described steps are provided for the **Cloud** version of WProofreader.

WProofreader add-on for rich text editors enables grammar and spell check, combining spelling and grammar suggestions while you type or work with your text in a floating dialog.

Its auto-searching feature enables detecting new editable fields on the page and proofreading the text they contain automatically on hover selection. No additional actions or plugins are required for enabling WProofreader in a specific WYSIWYG editor or HTML editable element. A single configuration applies to almost all editors and controls. Here is the list of WProofreader supported integrations.

Supported integrations

The autoSearch feature enables detecting new editable fields on the page and proofreading the text they contain automatically on hover selection. No additional actions or plugins are required for enabling WProofreader in a specific WYSIWYG editor or HTML editable control. A single configuration applies to almost all editors and controls.

HTML integration	WYSIWYG editors
HTML editable controls: <input> disabled by default <textarea> **Elements with contenteditable attribute set to 'true':** <iframe>	• CKEditor 4 • CKEditor 5* • TinyMCE 4,5 • Froala Editor • Kendo UI • Quill • Redactor • Summernote • RadEditor -Telerik ASP.NET Editor • ProseMirror • Trix • Slate.js** • Draft** ⚠ * The dialog mode of WProofreader is not supported with real-time collaboration features or multi-root editor modes of CKEditor 5. ** Integration with Slate.js and Draft won't work properly with IE11.

WProofreader also has a plugin for WordPress 4.x and WordPress 5.x. For more details, check the official repository of WProofreader plugin for WordPress.

1. Obtain activation key

Sign up for the trial version to get started with WProofreader Cloud. Once you have subscribed, you will receive an email with your service ID, an activation key, required for the WProofreader service activation.

⚠ If you already have one, you can **skip** this step.

2. Initialize WProofreader

There are two approaches for initializing WProofreader in your web app: using the **config variable** or **inline data attributes**.

Option A. Initializing using config variable

1. Add config with options

Define the required configuration options in WEBSPELLCHECKER_CONFIG.

- The configuration options can be added as a script directly on the page or loaded from *.js file (e.g. wscbundle_config.js) where you define these options.
- It can be added to any location on a web page before or after the wscbundle.js script. However, if you add wscbundle.js **asynchronously**, the configuration options must be loaded **before** the script.

Required options for the WProofreader Cloud:

- Specify your **serviceId**. It is a required option for the service activation.
- Enable autoSearch functionality using the **autoSearch** parameter.

This is an example of the defining WEBSPELLCHECKER_CONFIG in a separate script directly on a web page.

```
<script>
        window.WEBSPELLCHECKER_CONFIG = {
                autoSearch: true,
                lang: 'auto', // set the default language
        serviceId: 'your-service-ID' // the activation key for the Cloud-based version
   };
</script>
```

Alternatively, you can create a *.js file (e.g. wscbundle_config.js) with WEBSPELLCHECKER_CONFIG on your end and then load it from the file on your web page as shown in example below.

This is an example of the **wscbundle_config.js** file with WEBSPELLCHECKER_CONFIG.

wscbundle_config.js

```
window.WEBSPELLCHECKER_CONFIG = {
        autoSearch: true,
        lang: 'en_US',
    serviceId: 'your-service-ID'
};
```

Here is an example of the script that you need to add on your web page with the path to **wscbundle_config.js**.

```
<script type="text/javascript" src="[path_to_config]/wscbundle_config.js"></script>
```

2. Add WProofreader script

Add the **wscbundle.js** script on your web page.

```
<script type="text/javascript" src="https://svc.webspellchecker.net/spellcheck31/wscbundle/wscbundle.js"><
/script>
```

Option B. Initializing using inline data attributes

In general, the initialization of WProofreader using inline attributes is a good option if you want to have a single script with the base options. In order to define an additional option as an inline data attribute, use '**date-wsc-option_name**', e.g. 'data-wsc-lang'.

You can find the full list of options available here.

```
<script
   data-wsc-serviceid="your-service-ID"
   data-wsc-autosearch="true"
   data-wsc-lang="en_US"
   src="https://svc.webspellchecker.net/spellcheck31/wscbundle/wscbundle.js">
 </script>
```

There is a limitation which must be taken into account when using inline data attributes for the WProofreader options. As for now, only options which have **boolean** or **string** types can be used as data attributes. The options with **array** or **number** type **are not supported** (e.g. **actionItems** , **suggestionsCount**, **moreSuggestionsCount).**

3. Adjust default options

You can adjust the default options for your needs. Here is a full list of WProofreader API options and their possible values.

You can check the demos of WProofreader integrations with a various rich text editors on our website.