Using WebSpellChecker Server Web API

WebSpellChecker Web API provides a set of spell and grammar checking commands and additional parameters to the WebSpellChecker engine.

Supported requests methods:

- GET
- POST

Supported output types:

- XML
- JSON

1. Overview

Parameter	Commands	
cmd	• check_spelling	The command sets specific parameters and values for spell checking of a given piece of text.
	• grammar_check	The command defines parameters for grammar checking of a given piece of text.
	• user_dictionary	The command defines actions that will be performed with a user dictionary.
	• get_lang_list	The command returns a list of languages available and supported for spell checking in the current version of the application.
	• ver	The command returns a version of the application installed.
	• status	The command returns statuses of the application core engines (Spell Check, Grammar and Thesaurus).

Depending on your tasks and needs, you can form and send your API requests using GET or POST methods. Below you will find templates for both request methods.

1.1. Template of Request URL using GET

Request URL (GET):

```
http(s)://your_host_name:2880/?cmd=[command]&[parameter]=[value]
```

1.2. Template of Request URL using POST

Request URL (POST):

```
http(s)://your_host_name:2880/?
```

Body (Raw):

```
cmd=[command]&[parameter]=[value]&customerid=[encrypted_customer_ID]
```

2. Check Spelling Command

Command name: check_spelling

Here is a list of all possible parameters and values that can be used with the **check_spelling** command.

#	Parameter	Possible Values	Default Value	Description
1	format	• json • xml	json	The parameter set a response format for output data.
2	callback	• callback function name		The parameter specifies a callback function name that will be used to manipulate with the JSON data received from the server. Such approach enables sharing of data bypassing same-origin policy. It can be used only along with “format=json”.

3	out_type	• positions – Return positions and length of misspelled words in a given text and their suggestions. • words – Return misspelled words and their suggestions.	words	The parameter defines a type of data output whether return misspelled words positions in the provided text or exact words.
4	ignore_all_caps	• 0 – Do not ignore all words written in capital letters (e.g. UPPERCASE). • 1 – Ignore all words written in capital letters.	0	The parameter regulates whether to ignore capitalized words or not.
5	ignore_words_with_numbers	• 0 – Do not ignore words that contain numbers (e.g. Number1). • 1 – Ignore words that contain numbers.	0	The parameter regulates whether to ignore words containing numbers or not.
6	ignore_mixed_case	• 0 – Do not ignore words with mixed case letters (e.g. MixedCase). • 1 – Ignore words with mixed case letters.	0	The parameter regulates whether to ignore words written with mixed case letters or not.
7	ignore_domain_names	• 0 – Do not ignore web addresses that start with either “www”, “http:” or “https:” and end with a domain name. • 1 – Ignore web addresses and domain names.	0	The parameter regulates whether to ignore domain names, web addresses or not.
8	text	• plain text		The parameter defines a text which will be sent for check spelling. The text has to be in the UTF-8 encoding. Any found tags in the text will be interpreted as a plan text as well.
9	slang	• Default languages short codes (e.g. en_US) • Additional languages short codes (e.g. ar_SA)	en_US	The parameter sets a shot code of a language which will be used for check spelling.
10	user_dictionary	• user dictionary name (e.g. testdict)		The parameter specifies a user dictionary name which will be used during check spelling.
11	custom_dictionary	• custom dictionary ID value assigned in a DicId parameter.		This parameter specifies a custom dictionary which will be used for check spelling.
12	user_wordlist	• additional wordlist		The parameter provides the list of additional coma-separated words which will be used for spellchecking.
13	version	1.0	1.0	The parameter indicates the version of Web API.
14	customerid	1:wiN6M-YQYOz2-PTPoa2-3yaA92-PmWom-3CEx53-jHqwR3-NYK6b-XR5Uh1-M7YAp4		This parameter specifies a special customer ID value that has to be passed to a request query. ⚠ Starting WebSpellChecker 4.8.6, there is not need to use this parameter for the Server version of the application.

Example 2.1: Check Spelling Request using GET (Output in XML)

Request URL (GET):

```
http(s)://your_host_name:2880/?cmd=check_spelling&format=xml&text=This sampl text demonstrates the work of the WebSpellChecker Web API service.&out_type=words&slang=en_US
```

Parameters:

- Command: check_spelling

- Format: XML
- Text: “This sampl text demonstrates the work of the WebSpellChecker Web API service.”
- Output: words
- Language: American English (en_US)

Request Response:

```
<?xml version="1.0" encoding="utf-8"?>
<check_spelling>
    <misspelling>
        <word>sampl</word>
        <ud>false</ud>
        <suggestions>
            <suggestion>sample</suggestion>
            <suggestion>sampled</suggestion>
            <suggestion>sampler</suggestion>
            <suggestion>samples</suggestion>
            <suggestion>ample</suggestion>
            <suggestion>amply</suggestion>
            <suggestion>scamp</suggestion>
            <suggestion>stamp</suggestion>
        </suggestions>
    </misspelling>
</check_spelling>
```

Example 2.2: Check Spelling Request (Output in JSON)

Here is a sample of the POST request to check spelling for a given piece of text.

Request URL (GET):

```
http(s)://your_host_name:2880/?cmd=check_spelling&format=json&text=This sampl text demonstrates the work of the WebSpellChecker Web API service.&out_type=words&slang=en_US
```

Parameters:

- Command: check_spelling
- Format: JSON
- Text: “This sampl text demonstrates the work of the WebSpellChecker Web API service.”
- Output: words
- Language: American English (en_US)

Request Response:

```
[
    {
        "word": "sampl",
        "ud": false,
        "suggestions": [
            "sample",
            "sampled",
            "sampler",
            "samples",
            "ample",
            "amply",
            "scamp",
            "stamp"
        ]
    }
]
```

Example 2.3: Check Spelling Request using POST (Output in JSON)

Here we use the same request and parameters as described in example above but form it as a POST request.

Request URL (GET):

```
http(s)://your_host_name:2880/?
```

Body (Raw):

```
cmd=check_spelling&format=json&text=This sampl text demonstrates the work of the WebSpellChecker Web API service.&out_type=words&slang=en_US
```

Request Response:

```
[
   {
       "word": "sampl",
       "ud": false,
       "suggestions": [
           "sample",
           "sampled",
           "sampler",
           "samples",
           "ample",
           "amply",
           "scamp",
           "stamp"
       ]
   }
]
```

3. Grammar Check Command

Command name: grammar_check

Here is a list of all possible parameters and values that can be used with the **grammar_check** command.

#	Parameter	Possible Values	Default Value	Description
1	format	• json • xml	json	The parameter set a response format for output data.
2	callback	• callback function name		The parameter specifies a callback function name that will be used to manipulate with the JSON data received from the server. Such approach enables sharing of data bypassing same-origin policy. It can be used only along with “format=json”.
3	text	• plain text		The parameter defines a text which will be sent for grammar checking. The text has to be in the UTF-8 encoding. Any found tags in the text will be interpreted as a plan text as well.
4	slang	• Default languages short codes (e.g. en_US) • Additional languages short codes (e.g. uk_UA)	en_US	The parameter sets a shot code of a language which will be used for grammar checking.

Example 3.1: Grammar Check Request using GET (output in XML)

Request URL (GET):

```
http(s)://your_host_name:2880/?cmd=grammar_check&format=xml&text=web API provides a gramar checking command that will help you builds a custom solution.&slang=en_US
```

Parameters:

- Command: grammar_check
- Format: XML
- Text: “Web API provides a gramar checking command that will help you builds a custom solution.”
- Language: American English (en_US)

Request Response:

```
<?xml version="1.0" encoding="utf-8"?>
<grammar_check>
    <grammar_problem>
        <phrase>you builds</phrase>
        <description>Pronoun "you" conflicts with verb "builds."</description>
        <problem_id>437780848</problem_id>
        <suggestions>
            <suggestion>you build</suggestion>
            <suggestion>you, builds</suggestion>
        </suggestions>
    </grammar_problem>
</grammar_check>
```

Example 3.2: Grammar Check Request using GET (Output in JSON)

Request URL (GET):

```
http(s)://your_host_name/spellcheck/script/ssrv.fcgi?cmd=grammar_check&format=json&text=Web API provides a gramar checking command that will help you builds a custom solution.&slang=en_US
```

Parameters:

- Command: grammar_check
- Format: JSON
- Text: “Web API provides a gramar checking command that will help you builds a custom solution.”
- Language: American English (en_US)

Request Response:

```
[
    {
        "sentence": "web API provides a gramar checking command that will help you builds a custom solution",
        "matches": [
            {
                "message": "This sentence does not start with an uppercase letter",
                "offset": 0,
                "length": 3,
                "rule": {
                    "id": "UPPERCASE_SENTENCE_START"
                },
                "suggestions": [
                    "Web"
                ]
            }
        ]
    }
]
```

Example 3.3: Grammar Check Request using POST (Output in JSON)

Here we use the same request and parameters as described in example above but form it as a POST request.

Request URL (POST):

```
http(s)://your_host_name:2880/?
```

Body (Raw):

```
cmd=grammar_check&format=json&text=web API provides a gramar checking command that will help you builds a
custom solution.&slang=en_US
```

Request Response:


```
[
    {
        "sentence": "web API provides a gramar checking command that will help you builds a custom solution",
        "matches": [
            {
                "message": "This sentence does not start with an uppercase letter",
                "offset": 0,
                "length": 3,
                "rule": {
                    "id": "UPPERCASE_SENTENCE_START"
                },
                "suggestions": [
                    "Web"
                ]
            }
        ]
    }
]
```


4. User Dictionary Command

Command name: user_dictionary

Here is a list of all possible parameters and values that can be used with the **user_dictionary** command.

#	Parameter	Possible Values	Default Value	Description
1	format	• json • xml	json	The parameter sets a response format for output data.
2	callback	• callback function name		The parameter specifies a callback function name that will be used to manipulate with the JSON data received from the server. Such approach enables sharing of data bypassing same-origin policy. It can be used only along with “format=json”.

3	action	• create – Create a new user dictionary. • rename – Rename an existing user dictionary. • delete – Delete an existing user dictionary. • addword – Add a new word to a specified user dictionary. • deleteword – Remove a word from a specified user dictionary. • editword – Edit a word in a specified user dictionary. • check – Check if a specified user dictionary exists on the server. • getdict – Get content of a specified user dictionary (for JSON only).		The parameter defines an action that can be used to manipulate a user dictionary.

Here is a list of all possible parameters and values that can be used with the user_dictionary **action** parameter.

#	Action Parameter	Parameters	Possible Values	Description
1	create	name	• name of a new user dictionary	The action that creates a new user dictionary.
		wordlist	• coma-separated words which will be added to a new dictionary	
2	delete	name	• name of a selected user dictionary	The action that deletes a selected user dictionary.
3	rename	name	• name of a selected user dictionary	The action that renames a specified dictionary and sets a new name.
		new_name	• a new name for a chosen user dictionary	
4	check	name	• name of a chosen user dictionary	The action that checks if a specified user dictionary exists on the server.
5	getdict	name	• name of a required user dictionary	The action requests content of a specified user dictionary. ⚠ The **getdict** action is available only for the **JSON** format.
6	addword	name	• name of a chosen user dictionary	The action adds new word(s) to a specified user dictionary. If you are adding more than one word at a time, all new words must be separated with commas accordingly.
		word	• a new word which will be added to a specified user dictionary	
7	deleteword	name	• name of a chosen user dictionary	The action removes a word from a specified user dictionary.

		word	• word which will be removed from a specified user dictionary	
8	editword	name	• name of a chosen user dictionary	The action replaces a word in a specified user dictionary with a new one.
		word	• word which will be edited	
		new_word	• a new word which replaces a word picked for editing	

Example 4.1: Create User Dictionary (XML)

Request URL (GET):

```
http(s)://your_host_name:2880/?
cmd=user_dictionary&format=xml&action=create&name=user_dictionary&wordlist=SCAYT, SpellCheckAsYouType, WSC,
WebSpellChecker, WProofreader
```

Parameters:

- Command: user_dictionary
- Action: Create
- Name: “user_dictionary”
- Wordlist: “SCAYT, SpellCheckAsYouType, WSC, WebSpellChecker, Proofreader”
- Format: XML

Request Response:

```
<?xml version="1.0" encoding="utf-8"?>
<dictionary>
    <name>user_dictionary</name>
    <action>create</action>
</dictionary>
```

Example 4.2: Get User Dictionary Content (JSON)

Request URL (GET):

```
http(s)://your_host_name:2880/?cmd=user_dictionary&format=json&action=getdict&name=user_dictionary
```

Parameters:

- Command: user_dictionary
- Action: getdict
- Name: “user_dictionary”
- Format: JSON

Request Response:

```
{"name":"user_dictionary","action":"getdict","wordlist":["SCAYT"," SpellCheckAsYouType"," WSC","
WebSpellChecker"," Proofreader"]}
```


```
{
    "name": "user_dictionary",
    "action": "getdict",
    "wordlist": [
        "SCAYT",
        "SpellCheckAsYouType",
        "WSC",
        "WebSpellChecker",
        "WProofreader"
    ],
    "modificationTime": 1571762101
}
```

Example 4.3: Get User Dictionary Content using POST (Output in JSON)

Using the same request and parameters as described in example 4.2 but form it as a POST request.

Request URL (POST):

```
http(s)://your_host_name:2880/?
```

Body (Raw):

```
cmd=user_dictionary&format=json&action=getdict&name=user_dictionary
```

Request Response:

```
{
    "name": "user_dictionary",
    "action": "getdict",
    "wordlist": [
        "SCAYT",
        "SpellCheckAsYouType",
        "WSC",
        "WebSpellChecker",
        "WProofreader"
    ],
    "modificationTime": 1571762101
}
```

5. Get Languages List Command

Command name: get_lang_list

Example 5.1: Get Languages List (JSON)

Request URL (GET):

```
http(s)://your_host_name:2880/?cmd=get_lang_list
```

Parameters:

- Command: get_lang_list

⚠ By default, the output format for **get_lang_list** command is JSON.

Request Response:


```
{
    "langList": {
        "ltr": {
            "en_US": "American English",
            "en_GB": "British English",
            "fr_FR": "French",
            "it_IT": "Italian",
            "de_DE": "German",
            "es_ES": "Spanish",
            "pt_BR": "Brazilian Portuguese",
            "da_DK": "Danish",
            "nl_NL": "Dutch",
            "nb_NO": "Norwegian Bokmal",
            "pt_PT": "Portuguese",
            "sv_SE": "Swedish",
            "el_GR": "Greek",
            "en_CA": "Canadian English",
            "fr_CA": "Canadian French",
            "fi_FI": "Finnish",
            "uk_UA": "Ukrainian"
        },
        "rtl": {}
    }
}
```


6. Check Version Command

ⓘ **Command name:** ver

⚠ By default, the output format for **ver** command is a simple HTML page.

Example 6.1: Check Application Version

Request URL (GET):
`http(s)://your_host_name:2880/?cmd=ver`

```
<html>
    <style> * { font-family:"Verdana,Helvetica";font-size:10px}</style>
    <body bgcolor=white>
        <body>
            <center>
                <table border=1>
                    <tr>
                        <td colspan=2>
                            <center>
                                <b>(c) 2000-2017 WebSpellChecker LLC
                                    <br>All rights reserved.
                                    <br>www.webspellchecker.net
                                </b>
                            </center>
                        </td>
                    </tr>
                    <tr>
                        <td><b>Program name:</b></td>
                        <td>ssrv</td>
                    </tr>
                    <tr>
                        <td>
                            <b>Program version:</b>
                        </td>
                        <td>4.9.5 x64 master:f7f1a0d (452) #62 for Linux</td>
                    </tr>
                    <tr>
                        <td colspan=2></td>
                    </tr>
                </table>
            </center>
        </body>
</html>
```

7. Check Engines Status Command

Command name: status

By default, the output format for **status** command is a simple text page.

Example 7.1: Check Engines Status

Request URL (GET):

```
http(s)://your_host_name:2880/?cmd=status
```

```
Spell Check Engine is ACTIVE
Grammar Engine is NOT ACTIVE
Thesaurus Engine is ACTIVE

Please find the description of the statuses that are received from SSRV component below:

'ACTIVE' status means that an Engine works properly.
'NOT ACTIVE' status means that an Engine is not enabled or does not work properly. Also 'NOT ACTIVE' status
appears for the Grammar-check Engine in case with the installation under the Linux-based environments.
For more details and assistance, please contact the technical support team at support@webspellchecker.net.
```