_include_wsc55x_server_web_api

1. Overview

There are the main commands that are available for Web API requests:

Parameter	Commands	Description
cmd	**check**	RECOMMENDED Global check command that checks text for **all enabled types of errors** (spelling, grammar, etc.) and its parameters. Check more in Check command.
	check_spelling	Check spelling command and its parameters.
	grammar_check	Grammar check command and its parameters.
	user_dictionary	User dictionary command and actions that can be performed with user dictionaries.
	detect_language	Detect the language of given text. Check more in Detect language command.
	get_info	Get general information about the languages enabled and other permissions for the current version of the application build.
	get_lang_list	DEPRICATED A list of languages available and supported for **spell checking** only in the current version of the application. It won't return languages where there is only grammar check (e.g. Chinese).
	ver	The current version of the application installed.
	status	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):

```
https://your_host_name:443/virtual_directory/api?cmd=[command]&[parameter]=[value]
```

1.2. Template of Request URL using POST

Endpoint / Request URL (POST):

```
https://your_host_name:443/virtual_directory/api?
```

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 response format for output data.
2	callback	• callback function name		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	A type of data output specifying the way to 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	Ignore capitalized words.
5	ignore_words_with_numbers	• 0 – Do not ignore words that contain numbers (e.g. Number1). • 1 – Ignore words that contain numbers.	0	Ignore words containing numbers.
6	ignore_mixed_case	• 0 – Do not ignore words with mixed case letters (e.g. MixedCase). • 1 – Ignore words with mixed case letters.	0	Ignore words written with mixed case letters.
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	Ignore domain names, web addresses.
8	text	• plain text		A piece of 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	lang	• Supported languages	en_US	A short code of a language which will be used for spell checking.
10	user_dictionary	• user dictionary name (e.g. testdict)		A user dictionary name which will be used during spell checking.
11	custom_dictionary	• custom dictionary ID value assigned in a DicId parameter.		A custom dictionary which will be used for spell checking.
12	user_wordlist	• additional wordlist		The list of additional comma-separated words which will be used for spell checking.
13	version	1.0	1.0	The version of Web API.
14	customerid	1:wiN6M-YQYOz2-PTPoa2-3yaA92-PmWom-3CEx53-jHqwR3-NYK6b-XR5Uh1-M7YAp4		A special customer ID value (activation key) that has to be passed to a request query. It's obtained upon subscription to the Cloud services (paid or trial). Applicable only for the Cloud version.

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

Request URL (GET):

```
http(s)://server_endpoint/?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)

Request URL (GET):

```
http(s)://server_server_endpoint/?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.

Endpoint / Request URL (POST):

```
https://your_host_name:443/virtual_directory/api?
```

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	A response format for output data.
2	callback	• callback function name		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		A piece of 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	lang	• Supported languages	en_US	A short 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)://server_endpoint/?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)://server_endpoint/?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.

Entry point / Request URL (POST):

```
https://your_host_name:443/virtual_directory/api?
```

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	A response format for output data.
2	callback	• callback function name		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. • addwords – Add new words to a specified user dictionary. V5.28.0 • deleteword – Remove a word from a specified user dictionary. • deletewords – Remove words from a specified user dictionary. V5.28.0 • 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).		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	Create a new user dictionary.
		wordlist	• comma-separated words which will be added to a new dictionary	
2	delete	name	• name of a selected user dictionary	Delete a selected user dictionary.
3	rename	name	• name of a selected user dictionary	Rename 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	Check if a specified user dictionary exists on the server.
5	getdict	name	• name of a required user dictionary	Request 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	Add a new word to a specified user dictionary.
		word	• a new word which will be added to a specified user dictionary	
	addwords	wordlist	• a list of coma-separated words (also possible to add just one word)	V5.28.0
7	deleteword	name	• name of a chosen user dictionary	Remove a word from a specified user dictionary.

		word	• word which will be removed from a specified user dictionary	
	deletewords	wordlist	• a list of coma-separated words (also possible to remove just one word)	V5.28.0
8	editword	name	• name of a chosen user dictionary	Replace 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	

Starting from WebSpellChecker Server version 5.28.0.0, we have introduced two additional action parameters: "addwords" and "deletewords," which can be used as equivalents to the existing "addword" and "deleteword" parameters. However, for logical consistency, we recommend using these new parameters in combination with the "wordlist" parameter.

Example 4.1: Create User Dictionary (XML)

Request URL (GET):

```
http(s)://server_endpoint/?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, WProofreader”
- 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)://server_endpoint/?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",
        "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):

```
https://your_host_name:443/virtual_directory/api?
```

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)://server_endpoint/?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)://server_endpoint/?cmd=ver
```

```
{
    "Copyright": "(c) 2000-2019 WebSpellChecker LLC",
    "ProductWebsite": "webspellchecker.com",
    "ProgramVersion": "5.x.x.x x64 master:xxxxxxx (xxxx) #xx",
    "PackageVersion": "5.x.x.x master:xxxxxxx (xxx) #xx"
}
```

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)://server_endpoint/?cmd=status
```

```
{
    "SpellCheckEngine": {
        "active": true
    },
    "GrammarCheckEngine": {
        "active": true
    },
    "ThesaurusEngine": {
        "active": true
    }
}
```

8. HTTP Status Codes

Refer to Overview of HTTP Status Codes section for more information on HTTP responses you may get when integrating WebSpellChecker and testing REST API.