Check command

The **check** command was released in October 2019 as part of WebSpellChecker 5.5.4x. It combines all available check types (spelling and grammar) of text in a single command.

Command name: check

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

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

The list of parameters can be used and is available only when a spelling **check is enabled**. These parameters are marked yellow.

#	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 the JSON data received from the server. Such an 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 check. The text has to be in the UTF-8 encoding. Any found tags in the text will be interpreted as plan text as well. Note, that you shouldn't use # and & symbols in the text.
4	tokens	• Array of strings, e.g. ["This is a sentence number 1.", "This is a sentence number 2."]		The parameter defines a text separated into tokens that will be sent for check. The text should be presented as an array of strings. Right now, each string is a token which equals one sentence. You can use either **tokens** or **text** at a time in a request. However, using **text** is more common.
5	lang	• Supported languages	en_US	The parameter sets a short code of a language which will be used for check.
6	disable_spelling	• true • false	false	The parameter regulates whether to check text for spelling errors.
7	disable_grammar	• true • false	false	The parameter regulates whether to check text for grammar and style problems.
8	user_dictionary	• user dictionary name (e.g. testdict)		The parameter specifies a user dictionary name which will be used during check spelling.
9	user_wordlist	• additional wordlist		The parameter provides the list of additional comma-separated words which will be used for spell checking.
10	custom_dictionary	• custom dictionary IDs (e.g. 100694)		The parameter specifies global custom dictionary IDs which can be used during check spelling. Each new Dictionary on the creation obtains its unique Dictionary ID. You can find information about your custom dictionaries by logging in to your account panel on webspellchecker.com website (Login –> Dictionaries –> List of Custom Dictionaries).

11	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.
12	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.
13	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.
14	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.
15	min_word_length	• minimal number of letters in a word to be checked	3	The parameter specifies the minimal number of letters in the word which will be checked for spelling. E.g. if 3 is specified, the words with 2 letters and less will be ignored.
16	custom_punctuation	• string of chars (e.g. “-“)	-	The parameter defines a list of characters that should be considered as delimiters during spelling check.
17	short_answer	• true • false	false	The parameter is responsible for shortening every static string JSON key name, like messages or type, down to its first character, like: • *m - matches, message* • *o - offset* • *l - length* • *t - type* • *r - rule* • *s - suggestions*

Response structure

The **result** is an array of objects which contains matches, where **matches** is also an array of objects consisting of attribute-value pairs.

The table below represents the following attribute-value pairs:

Attribute	Type	Value	Description
type	string	• spelling • grammar	
offset	number		
length	number		
ud	boolean	• true	
suggestions	array of strings		

Type: Spelling

```
{
    "result": [
        {
            "matches": [
                {
                    "type": "spelling",
                    "offset": X1,
                    "length": Y1,
                                        "ud": true,
                    "suggestions": ["..."]
                }
            ]
        }
    ]
}
```

Type: Grammar

```
{
    "result": [
        {
            "matches": [
                {
                    "type": "grammar",
                    "offset": X2,
                    "length": Y2,
                    "rule": "...",
                    "message": "...",
                    "suggestions": ["..."]
                }
            ]
        }
    ]
}
```

Example 1.1 [GET]: Check request for American English text with all available check types (output in JSON)

Request URL (GET):

```
http(s)://server_entry_point/?cmd=check&text=this sampl text demonstrates the work of the Web API service.
&lang=en_US&format=json
```

Parameters:

- Command: *check*
- Text: *this sampl text demonstrates the work of the Web API service.*
- Language: *en_US*
- Format: *json*

Request Response:

```
{
    "result": [
        {
            "matches": [
                {
                    "type": "spelling",
                    "offset": 5,
                    "length": 5,
                    "suggestions": [
                        "sample",
                        "sampled",
                        "sampler",
                        "samples",
                        "ample",
                        "amply",
                        "scamp",
                        "stamp"
                    ]
                },
                {
                    "type": "grammar",
                    "offset": 0,
                    "length": 4,
                    "rule": "UPPERCASE_SENTENCE_START",
                    "message": "This sentence does not start with an uppercase letter",
                    "suggestions": [
                        "This"
                    ]
                }
            ]
        }
    ]
}
```


Example 1.2 [GET]: Check request for American English text with all available check types (output in XML)

Request URL (GET):

```
http(s)://server_entry_point/?cmd=check&text=this sampl text demonstrates the work of the Web API service.
&lang=en_US&format=xml
```

Parameters:

- Command: *check*
- Text: *this sampl text demonstrates the work of the Web API service.*
- Language: *en_US*
- Format: *xml*

Request Response:

```
<result>
        <result>
                <matches>
                        <matches>
                                <type>spelling</type>
                                <offset>5</offset>
                                <length>5</length>
                                <suggestions>
                                        <suggestions>sample</suggestions>
                                        <suggestions>sampled</suggestions>
                                        <suggestions>sampler</suggestions>
                                        <suggestions>samples</suggestions>
                                        <suggestions>ample</suggestions>
                                        <suggestions>amply</suggestions>
                                        <suggestions>scamp</suggestions>
                                        <suggestions>stamp</suggestions>
                                </suggestions>
                        </matches>
                        <type>grammar</type>
                        <offset>0</offset>
                        <length>4</length>
                        <rule>UPPERCASE_SENTENCE_START</rule>
                        <message>This sentence does not start with an uppercase letter</message>
                        <suggestions>
                                <suggestions>This</suggestions>
                        </suggestions>
                </matches>
        </result>
</result>
```

Example 1.3 [GET]: Check request for American English text as two tokens with all available check types (output in JSON)

Request URL (GET):

```
https://server_entry_point/?cmd=check&tokens=["this sampl text.", " It demonstrate the work of the Web API service."]&lang=en_US
```

Parameters:

- Command: *check*
- Tokens: *["this sampl text.", " It demonstrate the work of the Web API service."]*
- Language: *en_US*
- Format: *json*

Request Response:

```
{
    "result": [
        {
            "matches": [
                {
                    "type": "spelling",
                    "offset": 5,
                    "length": 5,
                    "suggestions": [
                        "sample",
                        "sampled",
                        "sampler",
                        "samples",
                        "ample",
                        "amply",
                        "scamp",
                        "stamp"
                    ]
                },
                {
                    "type": "grammar",
                    "offset": 0,
                    "length": 4,
                    "rule": "UPPERCASE_SENTENCE_START",
                    "message": "This sentence does not start with an uppercase letter",
                    "suggestions": [
                        "This"
                    ]
                }
            ]
        },
        {
            "matches": [
                {
                    "type": "grammar",
                    "offset": 4,
                    "length": 11,
                    "rule": "IT_VBZ",
                    "message": "Did you mean demonstrates?",
                    "suggestions": [
                        "demonstrates"
                    ]
                }
            ]
        }
    ]
}
```


Example 1.4 [GET]: Check request for American English text as two tokens with all available check types and shorntened response (output in JSON)

Request URL (GET):

```
https://server_entry_point/?cmd=check&tokens=["this sampl text.", " It demonstrate the work of the Web API service."]&lang=en_US&short_answer=true
```

Parameters:

- Command: *check*
- Tokens: *["this sampl text.", " It demonstrate the work of the Web API service."]*
- Language: *en_US*
- Format: *json*
- Short Answer: *true*

Request Response:

```
{
    "r": [
        {
            "m": [
                {
                    "t": "spelling",
                    "o": 5,
                    "l": 5,
                    "s": [
                        "sample",
                        "sampled",
                        "sampler",
                        "samples",
                        "ample",
                        "amply",
                        "scamp",
                        "stamp"
                    ]
                },
                {
                    "t": "grammar",
                    "o": 0,
                    "l": 4,
                    "r": "UPPERCASE_SENTENCE_START",
                    "m": "This sentence does not start with an uppercase letter",
                    "s": [
                        "This"
                    ]
                }
            ]
        },
        {
            "m": [
                {
                    "t": "grammar",
                    "o": 4,
                    "l": 11,
                    "r": "IT_VBZ",
                    "m": "Did you mean demonstrates?",
                    "s": [
                        "demonstrates"
                    ]
                }
            ]
        }
    ]
}
```


Example 1.4 [POST]: Check request for American English text with all available check types (output in JSON)

Here we use the same request and parameters as described in **example 1.1** but form it as a POST request.

Request URL (POST):

```
https://server_entry_point/?
```

Body (Raw):

```
cmd=check&text=this sampl text demonstrates the work of the Web API service.&lang=en_US&format=json
```

Request Response:

```
{
    "result": [
        {
            "matches": [
                {
                    "type": "spelling",
                    "offset": 5,
                    "length": 5,
                    "suggestions": [
                        "sample",
                        "sampled",
                        "sampler",
                        "samples",
                        "ample",
                        "amply",
                        "scamp",
                        "stamp"
                    ]
                },
                {
                    "type": "grammar",
                    "offset": 0,
                    "length": 4,
                    "rule": "UPPERCASE_SENTENCE_START",
                    "message": "This sentence does not start with an uppercase letter",
                    "suggestions": [
                        "This"
                    ]
                }
            ]
        }
    ]
}
```