MediaWiki

API:REST API/Reference

< API:REST API

To view this API's documentation in an interactive sandbox, visit Special:RestSandbox on any Wikimedia wiki. For example: English Wikipedia REST sandbox.

Search

MediaWiki version: **≥ 1.35**

Search result object

The search result object represents a wiki page matching the requested search.

Example

```
{
  "id": 38930,
  "key": "Jupiter",
  "title": "Jupiter",
  "excerpt": "<span class=\"searchmatch\">Jupiter</span> is the fifth planet from the Sun and the
largest in the Solar System. It is a gas giant with a mass one-thousandth that of the Sun, but two-
and-a-half",
  "matched_title": null,
  "description": "fifth planet from the Sun and largest planet in the Solar System",
  "thumbnail": {
    "mimetype": "image/jpeg",
    "size": null,
    "width": 200,
    "height": 200,
    "duration": null,
    "url":
"//upload.wikimedia.org/wikipedia/commons/thumb/2/2b/Jupiter_and_its_shrunken_Great_Red_Spot.jpg/200px
-Jupiter_and_its_shrunken_Great_Red_Spot.jpg"
  }
}
```

Schema

Name	Required?	Type	Description
`id`	Required	integer	Page identifier
`key`	Required	string	Page title in URL-friendly format
`title`	Required	string	Page title in reading-friendly format
`excerpt`	Required	string	*For search pages endpoint:* A few lines giving a sample of page content with search terms highlighted with `<span class=\"searchmatch\">` tags. Excerpts may end mid-sentence *For autocomplete page title endpoint:* Page title in reading-friendly format
`matched_title`	Optional	string	Title of the page redirected from, if the search term originally matched a redirect page or `null` if search term did not match a redirect page.
`description`	Required	string	In Wikimedia projects: Short summary of the page topic based on the corresponding entry on Wikidata or `null` if no entry exists. See Extension:ShortDescription to populate this field in third-party installations.
`thumbnail`	Required	object	Reduced-size version of the page's lead image or `null` if no lead imagine exists ▪ `mimetype` (string): Thumbnail media type ▪ `size` (integer): File size in bytes or `null` if not available ▪ `width` (integer): Maximum recommended image width in pixels or `null` if not available ▪ `height` (integer): Maximum recommended image height in pixels or `null` if not available ▪ `duration` (integer): Length of the video, audio, or multimedia file or `null` for other media types ▪ `url` (string): URL to download the file

Search pages

Path	`/search/page?q=search terms`	**Method**	`GET`
Content type	`application/json`	**Returns**	`pages` object containing array of search results

Searches wiki page titles and contents for the provided search terms, and returns matching pages.

When using this endpoint on your wiki
This endpoint uses the search engine configured in the $wgSearchType configuration setting and

returns results in the namespaces configured by $wgNamespacesToBeSearchedDefault.

Examples

curl

```
# Search English Wikipedia for up to 20 pages containing information about Jupiter
$ curl "https://en.wikipedia.org/w/rest.php/v1/search/page?q=jupiter&limit=20"
```

Python

```
# Search English Wikipedia for up to 20 pages containing information about Jupiter
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/search/page'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}
params = {
    'q': 'jupiter',
    'limit': '20'
}

response = requests.get(url, headers=headers, params=params)
data = response.json()

print(data)
```

PHP

```
<?php
/*
Search English Wikipedia for up to 20 pages containing information about Jupiter
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/search/page";
$params = [ "q" => "jupiter", "limit" => "20" ];
$url = $url . "?" . http_build_query( $params );

$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```
/*
    Search English Wikipedia for up to 20 pages containing information about Jupiter
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/search/page";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}
  let params = {
    'q': 'jupiter',
    'limit': '20'
  };
  let query = Object.keys(params)
             .map(k => k + '=' + encodeURIComponent(params[k]))
             .join('&');
  url = url + '?' + query;

  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/search/page?q=jupiter&limit=20)

Parameters

q required | query | string	Search terms
`limit` optional | query | string	Maximum number of search results to return, between 1 and 100. Default: 50

Responses

200	Success: Results found. Returns a `pages` object containing an array of search results.
200	Success: No results found. Returns a `pages` object containing an empty array.
400	Query parameter not set. Add q parameter.
400	Invalid limit requested. Set `limit` parameter to between 1 and 100.
500	Search error

Autocomplete page title

Path	`/search/title?q=search terms`	**Method**	`GET`
Content type	`application/json`	**Returns**	`pages` object containing array of search results

Searches wiki page titles, and returns matches between the beginning of a title and the provided search terms. You can use this endpoint for a typeahead search that automatically suggests relevant pages by title.

When using this endpoint on your wiki
This endpoint uses the search engine configured in the $wgSearchType configuration setting and returns results in the namespaces configured by $wgNamespacesToBeSearchedDefault. Results may differ depending on the configured search backend. While the default backend only applies basic case folding and prefix matches, more advanced backends may apply more complex variations. In the case of CirrusSearch for instance, matches are based on the Elastic Search completion suggester (https://www.elastic.co/guide/en/elasticsearch/reference/current/search-suggesters.html#completion-suggester).

Examples

curl

```
# Search English Wikipedia for up to 5 pages with titles that start with "solar"
$ curl "https://en.wikipedia.org/w/rest.php/v1/search/title?q=solar&limit=5"
```

Python

```
# Search English Wikipedia for up to 5 pages with titles that start with "solar"
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/search/title'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}
params = {
    'q': 'solar',
    'limit': '5'
}

response = requests.get(url, headers=headers, params=params)
data = response.json()

print(data)
```

PHP

```
<?php
/*
Search English Wikipedia for up to 5 pages with titles that start with "solar"
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/search/title";
$params = [ "q" => "solar", "limit" => "5" ];
$url = $url . "?" . http_build_query( $params );

$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```
/*
    Search English Wikipedia for up to 5 pages with titles that start with "solar"
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/search/title";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}
  let params = {
    'q': 'solar',
    'limit': '5'
  };
  let query = Object.keys(params)
             .map(k => k + '=' + encodeURIComponent(params[k]))
             .join('&');
  url = url + '?' + query;

  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/search/title?q=solar&limit=5)

Parameters

q required | query | string	Search terms
`limit` optional | query | string	Maximum number of search results to return, between 1 and 100. Default: 50

Responses

200	Success: Results found. Returns a `pages` object containing an array of search results.
200	Success: No results found. Returns a `pages` object containing an empty array.
400	Query parameter not set. Add q parameter.
400	Invalid limit requested. Set `limit` parameter to between 1 and 100.
500	Search error

Pages

MediaWiki version: **≥ 1.35**

Page object

The page object represents the latest revision of a wiki page.

Example

```
{
  "id": 9228,
  "key": "Earth",
  "title": "Earth",
  "latest": {
    "id": 963613515,
    "timestamp": "2020-06-20T20:05:55Z"
  },
  "content_model": "wikitext",
  "license": {
    "url": "//creativecommons.org/licenses/by-sa/3.0/",
    "title": "Creative Commons Attribution-Share Alike 3.0"
  },
  "html_url": "https://en.wikipedia.org/w/rest.php/v1/page/Earth/html"
}
```

Schema

`id` required | integer	Page identifier
`key` required | string	Page title in URL-friendly format
`title` required | string	Page title in reading-friendly format
`latest` required | object	Information about the latest revision, including: ▪ `id` (integer): Revision identifier for the latest revision ▪ `timestamp` (string): Timestamp of the latest revision in ISO 8601 format
`content_model` required | string	Type of content on the page. See the content handlers reference for content models supported by MediaWiki and extensions.
`license` required | map of strings	Information about the wiki's license, including: ▪ `url` (string): URL of the applicable license based on the $wgRightsUrl setting ▪ `title` (string): Name of the applicable license based on the $wgRightsText setting
`html_url` required | string *(Get page only)*	API route to fetch the content of the page in HTML
`html` required | string *(Get page with HTML only)*	Latest page content in HTML, following the HTML specification
`source` required | string *(Get page source, create page, and update page only)*	Latest page content in the format specified by the `content_model` property

Page language object

The page language object represents a wiki page and its language.

Example

```
{
  "code": "pl",
  "name": "polski",
  "key": "Ziemia",
  "title": "Ziemia"
}
```

Schema

`code` required | string	Language code. For Wikimedia projects, see the site matrix on Meta-Wiki.
`name` required | string	Translated language name
`key` required | string	Translated page title in URL-friendly format
`title` required | string	Translated page title in reading-friendly format

Create page

Path	`/page`	**Method**	`POST`
Accepts	`application/json`	**Body**	see schema below
Content type	`application/json`	**Returns**	Page object with `source` property

Creates a wiki page. The response includes a `location` header containing the API endpoint to fetch the new page.

This endpoint is designed to be used with the OAuth extension authorisation process. Callers using cookie-based authentication instead must add a CSRF `token` to the request body. To get a CSRF token, see the Action API.

Examples

curl

```
# Create a user sandbox page on English Wikipedia
$ curl -X POST https://en.wikipedia.org/w/rest.php/v1/page -H "Content-Type: application/json" -H
"Authorization: Bearer $TOKEN" --data '{"source": "Hello, world!", "title": "User:<my
username>/Sandbox", "comment": "Creating a test page with the REST API"}'
```

Python

```python
# Create a user sandbox page on English Wikipedia
import requests
import json

url = 'https://en.wikipedia.org/w/rest.php/v1/page'

# Substitute your OAuth token
headers = {
    'User-Agent'   : 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)',
    'Content-Type' : 'application/json',
    'Authorization': 'Bearer $TOKEN'
}

# Substitute your username
request_data = {
    "source" : "Hello, world!",
    "title"  : "User:<my username>/Sandbox",
    "comment": "Creating a test page with the REST API"
}

response = requests.post( url, headers=headers, data = json.dumps(request_data) )
output = response.json()

print(output)
```

PHP

```php
<?php
/*
Create a user sandbox page on English Wikipedia
*/

$url = 'https://en.wikipedia.org/w/rest.php/v1/page';

// Substitute your username
$fields = [
    'source' => 'Hello, world!',
    'title' => 'User:<my username>/Sandbox',
    'comment' => 'Creating a test page with the REST API'
];

$json = json_encode( $fields );

$token = 'YOUR_OAUTH_TOKEN'; // Substitute your OAuth token
$authorization = 'Authorization: Bearer ' . $token;

$ch = curl_init();

curl_setopt( $ch, CURLOPT_URL, $url );
curl_setopt( $ch, CURLOPT_POST, true );
curl_setopt( $ch, CURLOPT_POSTFIELDS, $json );
curl_setopt( $ch, CURLOPT_HTTPHEADER, array( 'Content-Type: application/json' , $authorization ));
curl_setopt( $ch, CURLOPT_USERAGENT, 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)' );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );

$output = curl_exec( $ch );
curl_close( $ch );

echo( $output );
?>
```

JavaScript

```
/*
    Create a user sandbox page on English Wikipedia

    Substitute your OAuth token for $TOKEN.
    Substitute your username for <my username>.
*/

async function doFetch() {
  const response = await fetch(
    "https://en.wikipedia.org/w/rest.php/v1/page",
    {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer $TOKEN'
      },
      body: JSON.stringify({
        "source" : "Hello, world!",
        "title"  : "User:<my username>/Sandbox",
        "comment": "Creating a test page with the REST API"
      }),
      'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
    }
  );
  const data = await response.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Request schema

`source` required | string	Page content in the format specified by the `content_model` property
`title` required | string	Page title. See the manual for information about page titles in MediaWiki. For Wikimedia projects, visit individual wikis for policies about page title formats and characters. For example, for English Wikipedia, visit Article titles.
`comment` required | string	Reason for creating the page. To allow the comment to be filled in by the server, use `"comment": null`.
`content_model` optional | string	Type of content on the page. Defaults to `wikitext`. See the content handlers reference for content models supported by MediaWiki and extensions.
`token` optional | string	CSRF token required when using cookie-based authentication. Omit this property when authorising using OAuth.

Responses

201	Success: Page created. Returns page object with `source` property.
400	Missing `token` when using cookie-based authentication. Add a CSRF token to the request body, or use an OAuth authorisation flow.
400	Bad content model. Include a valid `content_model` based on available content handlers.
409	Page already exists
415	Unsupported Content-Type. Add the request header `Content-Type: application/json`.

Update page

Route	`/page/{title}`	**Content type**	`application/json`
Method	`PUT`	**Returns**	Page object with `source` property

Updates or creates a wiki page. This endpoint is designed to be used with the OAuth extension authorisation process. Callers using cookie-based authentication instead must add a CSRF `token` to the request body. To get a CSRF token, see the Action API.

To update a page, you need the page's latest revision ID and the page source. First call the get page source endpoint, and then use the `source` and `latest.id` to update the page. If `latest.id` doesn't match the page's latest revision, the API resolves conflicts automatically when possible. In the event of an edit conflict, the API returns a 409 error.

To create a page, omit `latest.id` from the request.

Examples

curl

```
# Update the sandbox page on English Wikipedia with "Hello, world!"
$ curl -X PUT https://en.wikipedia.org/w/rest.php/v1/page/Wikipedia:Sandbox -H "Content-Type:
application/json" -H "Authorization: Bearer $TOKEN" --data '{"source": "Hello, world!", "comment":
"Testing out the REST API", "latest": { "id": 555555555 }}'
```

Python

```
# Update the sandbox page on English Wikipedia with "Hello, world!"
import requests
import json

url = "https://en.wikipedia.org/w/rest.php/v1/page/Wikipedia:Sandbox"

# Substitute your OAuth token
headers = {
    "User-Agent" : "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)",
    "Content-Type" : "application/json",
    "Authorization" : "Bearer $TOKEN"
}

# Use the get page endpoint to get the latest revision ID
request_data = {
    "source" : "Hello, world!",
    "comment": "Testing out the REST API",
    "latest" : { "id": 555555555 }
}

response = requests.put( url, headers=headers, data = json.dumps(request_data) )
output = response.json()

print(output)
```

PHP

```
<?php
/*
Update the sandbox page on English Wikipedia with "Hello, world!"
*/

$page = 'Wikipedia:Sandbox';
$endpoint = 'https://en.wikipedia.org/w/rest.php/v1/page/';
$url = $endpoint . $page;

// Use the get page endpoint to get the latest revision ID
$fields = [
    'source' => 'Hello, world!',
    'comment' => 'Testing out the REST API',
    'latest' => [
        'id' => 555555555
    ]
];

$json = json_encode( $fields );

$token = 'YOUR_OAUTH_TOKEN'; // Substitute your OAuth token
```

```
$authorization = 'Authorization: Bearer ' . $token;

$ch = curl_init();

curl_setopt( $ch, CURLOPT_URL, $url );
curl_setopt( $ch, CURLOPT_CUSTOMREQUEST, 'PUT' );
curl_setopt( $ch, CURLOPT_POSTFIELDS, $json );
curl_setopt( $ch, CURLOPT_HTTPHEADER, array('Content-Type: application/json' , $authorization ));
curl_setopt( $ch, CURLOPT_USERAGENT, 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)' );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );

$output = curl_exec( $ch );
curl_close( $ch );

echo( $output );
?>
```

JavaScript

```
/*
    Update the sandbox page on English Wikipedia with "Hello, world!"

    Substitute your OAuth token for $TOKEN.
    Use the get page endpoint to get the latest revision ID.
*/

async function doFetch() {
  const response = await fetch(
    "https://en.wikipedia.org/w/rest.php/v1/page/Wikipedia:Sandbox",
    {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': 'Bearer $TOKEN'
      },
      body: JSON.stringify({
        "source": "Hello, world!",
        "comment": "Testing out the REST API",
        "latest": { "id": 555555555 }
      }),
      'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
    }
  );
  const data = await response.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Parameters

`title` required | path	Wiki page title

Request schema

`source` required | string	Page content in the format specified by the `content_model` property
`comment` required | string	Summary of the edit. To allow the comment to be filled in by the server, use `"comment": null`.
`latest` optional | object	Object identifying the base revision of the edit. You can fetch this information from the get page source endpoint.
`latest.id` optional | integer	Identifier for the revision used as the base for the new `source`, required for updating an existing page. To create a page, omit this property.
`content_model` optional | string	Type of content on the page. Defaults to `wikitext` for new pages or to the existing page's content model. See the content handlers reference for content models supported by MediaWiki and extensions.
`token` optional | string	CSRF token required when using cookie-based authentication. Omit this property when authorising using OAuth.

Responses

200	Success: Page updated. Returns page object with `source` property.
201	Success: Page created. Returns page object with `source` property.
400	Missing `token` when using cookie-based authentication. Add a CSRF token to the request body, or use an OAuth authorisation flow.
400	Bad content model. Ensure that the `content_model` property in the request body matches the `content_model` for the target page.
409	Page already exists. To update the existing page, provide the base revision identifier in `latest.id` in the request body.
409	Edit conflict. The error response includes the differences between the base revision specified in the request and the latest published revision. See the Wikidiff2 docs for information about the diff format. Requires Wikidiff2 extension 1.10+.
415	Unsupported Content-Type. Add the request header `Content-Type: application/json`.

Get page

Route	`/page/{title}/bare`	**Content type**	`application/json`
Method	`GET`	**Returns**	Page object with `html_url` property

For Subpages, the title requires the `/` to be encoded to `%2F`.

Returns the standard page object for a wiki page, including the API route to fetch the latest content in HTML, the license, and information about the latest revision.

Examples

curl

```
# Get information about the Jupiter article on English Wikipedia
$ curl "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/bare"
```

Python

```
# Get information about the Jupiter article on English Wikipedia
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/bare'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}


response = requests.get(url, headers=headers)
data = response.json()

print(data)
```

PHP

```
<?php
/*
Get information about the Jupiter article on English Wikipedia
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/bare";


$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```
/*
    Get information about the Jupiter article on English Wikipedia
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/bare";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}


  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/page/Earth/bare)

Parameters

`title` required | path	Wiki page title

Responses

200	Success: Page found. Returns page with `html_url` property.
301	Title normalisation redirect
404	Title or revision not found

Get page with HTML

Route	`/page/{title}/with_html`	**Content type**	`application/json`
Method	`GET`	**Returns**	Page object with `html` property

Returns information about a wiki page, including the license, latest revision, and latest content in HTML.

Examples

curl

```
# Get HTML and metadata for the Jupiter article on English Wikipedia
$ curl "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/with_html"
```

Python

```
# Get HTML and metadata for the Jupiter article on English Wikipedia
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/with_html'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}


response = requests.get(url, headers=headers)
data = response.json()

print(data)
```

PHP

```
<?php
/*
Get HTML and metadata for the Jupiter article on English Wikipedia
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/with_html";


$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```
/*
    Get HTML and metadata for the Jupiter article on English Wikipedia
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/with_html";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}


  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/page/Earth/with_html)

Parameters

`title` required | path | string	Wiki page title
`redirect` optional | query | bool	Set to no to not follow wiki redirects.

Responses

200	Success: Page found. Returns page with `html` property.
301	Title normalisation redirect
307	Wiki redirect
404	Title or revision not found

Get page source

Route	`/page/{title}`	**Content type**	`application/json`
Method	`GET`	**Returns**	Page object with `source` property

Returns the content of a wiki page in the format specified by the `content_model` property, the license, and information about the latest revision.

Examples

curl

```
# Get source and metadata for the Jupiter article on English Wikipedia
$ curl "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter"
```

Python

```
# Get source and metadata for the Jupiter article on English Wikipedia
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/page/Jupiter'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}


response = requests.get(url, headers=headers)
data = response.json()

print(data)
```

PHP

```
<?php
/*
Get source and metadata for the Jupiter article on English Wikipedia
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter";


$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```
/*
    Get source and metadata for the Jupiter article on English Wikipedia
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}


  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/page/Earth)

Parameters

`title` required | path	Wiki page title

Responses

200	Success: Page found. Returns page with `source` property.
301	Title normalisation redirect
404	Title or revision not found

Get HTML

Route	`/page/{title}/html`	**Content type**	`text/html`
Method	`GET`	**Returns**	Page HTML in HTML format

Returns the latest content of a wiki page in HTML.

Examples

curl

```
# Get the content of the Jupiter article on English Wikipedia in HTML
$ curl "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/html"
```

Python

```python
# Get the content of the Jupiter article on English Wikipedia in HTML
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/html'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}


response = requests.get(url, headers=headers)
data = response.json()

print(data)
```

PHP

```php
<?php
/*
Get the content of the Jupiter article on English Wikipedia in HTML
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/html";


$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```javascript
/*
    Get the content of the Jupiter article on English Wikipedia in HTML
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/html";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}


  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/page/Earth/html)

Parameters

`title` required | path | string	Wiki page title
`redirect` optional | query | bool	Set to no to not follow wiki redirects.
`flavor` optional | query | string	Choose the flavor of HTML to return. Currently available flavors: ▪ view – HTML for viewing the page. ▪ edit – HTML suitable for modifying and submitting back for editing. Contains annotations to support conversion back to wikitext. Support for more flavors may be added in the future.

Responses

200	Success. Returns page HTML in HTML format.
301	Title normalisation redirect
307	Wiki redirect
404	Title or revision not found

Get languages

Route	`/page/{title}/links/language`	**Content type**	`application/json`
Method	`GET`	**Returns**	Array of page languages

Searches connected wikis for pages with the same topic in different languages. Returns an array of page language objects that include the name of the language, the language code, and the translated page title.

Examples

curl

```
# Find articles from other Wikipedias linked to the Jupiter article on English Wikipedia
```

```
$ curl "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/links/language"
```

Python

```python
# Find articles from other Wikipedias linked to the Jupiter article on English Wikipedia
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/links/language'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}


response = requests.get(url, headers=headers)
data = response.json()

print(data)
```

PHP

```php
<?php
/*
Find articles from other Wikipedias linked to the Jupiter article on English Wikipedia
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/links/language";


$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```javascript
/*
    Find articles from other Wikipedias linked to the Jupiter article on English Wikipedia
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/links/language";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}


  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
```

```
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/page/Earth/links/language)

Parameters

`title` required | path	Wiki page title

Responses

200	Success: Languages found. Returns array of page languages.
404	Title not found

Get files on page

Route	`/page/{title}/links/media`	**Content type**	`application/json`
Method	`GET`	**Returns**	`files` object containing array of files

Returns information about media files used on a wiki page, up to 100 files.

Examples

curl

```
# Get media files used on the Jupiter article on English Wikipedia
$ curl "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/links/media"
```

Python

```
# Get media files used on the Jupiter article on English Wikipedia
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/links/media'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}
```

```
response = requests.get(url, headers=headers)
data = response.json()

print(data)
```

PHP

```php
<?php
/*
Get media files used on the Jupiter article on English Wikipedia
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/links/media";


$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```javascript
/*
    Get media files used on the Jupiter article on English Wikipedia
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/links/media";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}


  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/page/Earth/links/media)

Parameters

`title` required | path	Wiki page title

Responses

200	Success: Media files found. Returns `files` object containing array of files.
200	Success: No media files found. Returns `files` object containing an empty array.
404	Title not found
400	Page contains more than 100 media files

Transform

The transform endpoint provides on-the-fly round-trip conversion between wikitext and HTML.

MediaWiki version: **≥ 1.41**

This documentation does not yet cover all available features of the transform endpoint. See the documentation for the older RESTBase endpoint for additional information: https://en.wikipedia.org/api/rest_v1/#/Transforms

Convert Wikitext to HTML

Route	`/transform/wikitext/to/html/{title}`	**Content type**	`application/json`
Method	`POST`	**Payload**	Transform request body with `source`

Returns an HTML document.

Converts wikitext to HTML.

Examples

curl

```
# Render wikitext in the context of the Jupiter page on English Wikipedia, without modifying the page:
$ curl -X POST -H "Content-Type: application/json" --data '{ "wikitext": "== Hello Jupiter ==" }'
'https://en.wikipedia.org/w/rest.php/v1/transform/wikitext/to/html/Jupiter'
```

Parameters

`title` required | path	Wiki page title, used for context

Responses

200	Success: the response body contains the rendered HTML.

Convert HTML to Wikitext

Route	`/transform/html/to/wikitext/{title}`	**Content type**	`application/json`
Method	`POST`	**Payload**	Transform request body with `HTML`

Returns a wikitext document.

Converts wikitext to HTML.

Examples

curl

```
# Generate wikitext from HTML, in the context of the Jupiter page on English Wikipedia:
$ curl -X POST -H "Content-Type: application/json" --data '{ "html": "<h2> hello World </h2>" }'
'https://en.wikipedia.org/w/rest.php/v1/transform/html/to/wikitext/Jupiter'
```

Parameters

`title` required | path	Wiki page title, used for context

Responses

200	Success: the response body contains the wikitext.

Convert Wikitext to lint errors

Route	`/transform/wikitext/to/lint/{title}`	**Content type**	`application/json`
Method	`POST`	**Payload**	Transform request body with `wikitext`

Returns a JSON array of errors.

Transform wikitext into lint errors.

Examples

curl

```
# Get lint errors for wikitext in the context of the Jupiter page on English Wikipedia, without
modifying the page:
$ curl -X POST -H "Content-Type: application/json" --data '{ "wikitext": "== Hello Jupiter ==
[[Link|text]]<table><table>" }'
'https://en.wikipedia.org/w/rest.php/v1/transform/wikitext/to/lint/Jupiter'
```

Parameters

`title` required | path	Wiki page title, used for context

Responses

200	Success: the response body contains the lint errors.

Transform request body

Payload structure for transform requests.

Example

For converting wikitext to HTML:

```
{
  "wikitext": "Hello World"
}
```

For converting HTML to Wikitext:

```
{
  "html": "<h2>Hello World</h2>"
}
```

Media files

MediaWiki version: **≥ 1.35**

File object

The file object represents a file uploaded to a wiki.

Example

```
{
  "title": "The Blue Marble.jpg",
  "file_description_url": "//commons.wikimedia.org/wiki/File:The_Blue_Marble.jpg",
  "latest": {
    "timestamp": "2011-12-07T05:11:41Z",
    "user": {
      "id": 811185,
      "name": "Ultimate Roadgeek"
    }
  },
  "preferred": {
    "mediatype": "BITMAP",
    "size": null,
    "width": 599,
    "height": 599,
    "duration": null,
    "url": "https://upload.wikimedia.org/wikipedia/commons/thumb/7/78/The_Blue_Marble.jpg/599px-
The_Blue_Marble.jpg"
  },
  "original": {
    "mediatype": "BITMAP",
    "size": 7011595,
    "width": 3000,
    "height": 3002,
    "duration": null,
    "url": "https://upload.wikimedia.org/wikipedia/commons/7/78/The_Blue_Marble.jpg"
  },
  "thumbnail": {
    "mediatype": "BITMAP",
    "size": null,
    "width": 1023,
    "height": 1024,
    "duration": null,
    "url": "https://upload.wikimedia.org/wikipedia/commons/thumb/7/78/The_Blue_Marble.jpg/1023px-
The_Blue_Marble.jpg"
  }
}
```

Schema

`title` required | string	File title
`file_description_url` required | string	URL for the page describing the file, including license information and other metadata
`latest` required | object	Object containing information about the latest revision to the file, including: ▪ `timestamp` (string): Last modified timestamp in ISO 8601 format format ▪ `user` (object): Object containing information about the user who uploaded the file ▪ `id` (integer): User identifier ▪ `name` (string): Username
`preferred` required | object	Information about the file's preferred preview format, original format, and thumbnail format, including: ▪ `mediatype` (string): File type, one of: BITMAP, DRAWING, AUDIO, VIDEO, MULTIMEDIA, UNKNOWN, OFFICE, TEXT, EXECUTABLE, ARCHIVE, or 3D ▪ `size` (integer): File size in bytes or `null` if not available ▪ `width` (integer): Maximum recommended image width in pixels or `null` if not available ▪ `height` (integer): Maximum recommended image height in pixels or `null` if not available ▪ `duration` (integer): Length of the video, audio, or multimedia file or `null` for other media types ▪ `url` (string): URL to download the file
`original` required | object	
Get file only `thumbnail` required | object	

Get file

Route	`/file/{title}`	**Content type**	`application/json`
Method	`GET`	**Returns**	File

Returns information about a file, including links to download the file in thumbnail, preview, and original formats.

Examples

curl

```
# Get File:The_Blue_Marble.jpg on Wikimedia Commons
$ curl "https://en.wikipedia.org/w/rest.php/v1/file/File:The_Blue_Marble.jpg"
```

Python

```
# Get File:The_Blue_Marble.jpg on Wikimedia Commons
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/file/File:The_Blue_Marble.jpg'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}


response = requests.get(url, headers=headers)
data = response.json()

print(data)
```

PHP

```
<?php
/*
Get File:The_Blue_Marble.jpg on Wikimedia Commons
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/file/File:The_Blue_Marble.jpg";


$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```
/*
    Get File:The_Blue_Marble.jpg on Wikimedia Commons
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/file/File:The_Blue_Marble.jpg";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}


  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://commons.wikimedia.org/w/rest.php/v1/file/File:The_Blue_Marble.jpg)

Parameters

parameter	description
`title` required | path	File title

Responses

200	Success: file found. Returns file.
404	Title not found

History

MediaWiki version: **≥ 1.35**

Revision object

The revision object represents a change to a wiki page.

Example

```
{
  "id": 931281281,
  "page": {
    "id": 38930,
    "title": "Jupiter"
  },
  "size": 126009,
  "minor": false,
  "timestamp": "2019-12-18T01:39:24Z",
  "user": {
    "id": 27015025,
    "name": "InternetArchiveBot"
  },
  "comment": "Bluelinking 2 books for [[WP:V|verifiability]].) #IABot (v2.1alpha3",
  "delta": 231
}
```

Schema

`id` required | integer	Revision identifier
Get revision only `page` required | object	Object containing information about the page, including: ▪ `page_id` (integer): Page identifier ▪ `title` (string): Page title in reading-friendly format
`user` required | object	Object containing information about the user that made the edit, including: ▪ `name` (string): Username ▪ `id` (integer): User identifier For anonymous users, the `name` contains the originating IP address, and the `id` is `null`.
`timestamp` required | string	Time of the edit in ISO 8601 format
`comment` required | string	Comment or edit summary written by the editor. For revisions without a comment, the API returns `null` or `""`.
`size` required | integer	Size of the revision in bytes
`delta` required | integer	Number of bytes changed, positive or negative, between a revision and the preceding revision (example: `-20`). If the preceding revision is unavailable, the API returns **`null`**.
`minor` required | boolean	Set to **`true`** for edits marked as minor
`html_url` required | string *(Get revision only)*	API route to fetch the content of the revision in HTML
`html` required | string *(Get revision with HTML only)*	Revision content in HTML, following the HTML specification

`source` required | string *(Get revision source only)*	Revision content in the format specified by the `content_model` property

Get page history

Route	`/page/{title}/history`	**Content type**	`application/json`
Method	`GET`	**Returns**	Page history segment

Returns information about the latest revisions to a wiki page, in segments of 20 revisions, starting with the latest revision. The response includes API routes for the next oldest, next newest, and latest revision segments, letting you scroll through page history.

Examples

curl

```
# Get revisions made to the Jupiter article on English Wikipedia by bots prior to revision 939967546
$ curl "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/history?filter=bot&older_than=939967546"
```

Python

```
# Get revisions made to the Jupiter article on English Wikipedia by bots prior to revision 939967546
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/history'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}
params = {
    'filter': 'bot',
    'older_than': '939967546'
}

response = requests.get(url, headers=headers, params=params)
data = response.json()

print(data)
```

PHP

```
<?php
/*
Get revisions made to the Jupiter article on English Wikipedia by bots prior to revision 939967546
*/
```

```php
$url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/history";
$params = [ "filter" => "bot", "older_than" => "939967546" ];
$url = $url . "?" . http_build_query( $params );

$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```javascript
/*
    Get revisions made to the Jupiter article on English Wikipedia by bots prior to revision 939967546
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/history";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}
  let params = {
    'filter': 'bot',
    'older_than': '939967546'
  };
  let query = Object.keys(params)
              .map(k => k + '=' + encodeURIComponent(params[k]))
              .join('&');
  url = url + '?' + query;

  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/history?filter=bot&older_than=939967546)

Parameters

`title` required | path	Wiki page title
`older_than` optional | query	Accepts a revision ID. Returns the next 20 revisions older than the given revision ID.
`newer_than` optional | query	Accepts a revision ID. Returns the next 20 revisions newer than the given revision ID.
`filter` optional | query	Filter that returns only revisions with certain tags, one of: ▪ `reverted`: Returns only revisions that revert an earlier edit ▪ `anonymous`: Returns only revisions made by anonymous users ▪ `bot`: Returns only revisions made by bots ▪ `minor`: Returns only revisions marked as minor edits The API supports one filter per request.

Responses

200	Success: Revisions found. Returns a page history segment.
200	Success: No revisions found. Returns a page history segment with an empty `revisions` array.
400	Revision identifier must be greater than 0
400	Parameters `older_than` and `newer_than` cannot both be specified
400	Invalid parameter
404	Title or revision not found

Response schema

`latest` required | string	API route to get the latest revisions
`older` optional | string	If available, API route to get the prior revisions
`newer` optional | string	If available, API route to get the following revisions
`revisions` required | array	Array of 0-20 revision objects

Get page history counts

Route	`/page/{title}/history/counts/{type}`	**Content type**	`application/json`
Method	`GET`	**Returns**	History count object

Returns data about a page's history.

Examples

curl

```
# Get the number of edits to a page between revisions 384955912 and 406217369
$ curl "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/history/counts/edits?
from=384955912&to=406217369"
```

Python

```
# Get the number of edits to a page between revisions 384955912 and 406217369
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/history/counts/edits'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}
params = {
    'from': '384955912',
    'to': '406217369'
}

response = requests.get(url, headers=headers, params=params)
data = response.json()

print(data)
```

PHP

```
<?php
/*
Get the number of edits to a page between revisions 384955912 and 406217369
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/history/counts/edits";
$params = [ "from" => "384955912", "to" => "406217369" ];
$url = $url . "?" . http_build_query( $params );

$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```
/*
    Get the number of edits to a page between revisions 384955912 and 406217369
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/history/counts/edits";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}
  let params = {
    'from': '384955912',
    'to': '406217369'
  };
  let query = Object.keys(params)
             .map(k => k + '=' + encodeURIComponent(params[k]))
             .join('&');
  url = url + '?' + query;

  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/page/Jupiter/history/counts/edits?from=384955912&to=406217369)

Parameters

`title` required | path	Wiki page title
`type` required | path	Type of count, one of: ▪ `anonymous`: Edits made by anonymous users. Limit: 10,000 ▪ `bot`: Edits made by bots. Limit: 10,000 ▪ `editors`: Users or bots that have edited a page. Limit: 25,000 ▪ `edits`: Any change to page content. Limit: 30,000 ▪ `minor`: Edits marked as minor. If the minor edit count exceeds 2,000, the API returns a 500 error. Limit: 1,000 ▪ `reverted`: Edits that revert an earlier edit. Limit: 30,000
`from` optional | query	*For `edits` and `editors` types only* Restricts the count to between two revisions, specified by revision ID. The `from` and `to` query parameters must be used as a pair. The result excludes the edits or editors represented by the `from` and `to` revisions.
`to` optional | query	

Responses

200	Success
400	Invalid parameter or combination of parameters
404	Title or revision not found
500	Minor edit count exceeds 2000

Response schema

`count` required | integer	The value of the data point up to the type's limit. If the value exceeds the limit, the API returns the limit as the value of `count` and sets the `limit` property to **`true`**.
`limit` required | boolean	Returns **`true`** if the data point exceeds the type's limit.

Get revision

Path	`revision/{id}/bare`	**Method**	`GET`
Content type	`application/json`	**Returns**	Revision

Returns details for an individual revision.

Examples

curl

```
# Get information about revision 764138197
$ curl "https://en.wikipedia.org/w/rest.php/v1/revision/764138197/bare"
```

Python

```
# Get information about revision 764138197
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/revision/764138197/bare'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}


response = requests.get(url, headers=headers)
data = response.json()

print(data)
```

PHP

```
<?php
/*
Get information about revision 764138197
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/revision/764138197/bare";


$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```
/*
    Get information about revision 764138197
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/revision/764138197/bare";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}


  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/revision/764138197/bare)

Parameters

id required | path | string	Revision ID

Responses

200	Success: Revision found. Returns a revision.
404	Revision not found

Get revision source

Path	`revision/{id}`	**Method**	`GET`
Content type	`application/json`	**Returns**	Revision

Returns details for an individual revision.

Examples

curl

```
# Get the wikitext of revision 764138197
$ curl "https://en.wikipedia.org/w/rest.php/v1/revision/764138197"
```

Python

```
# Get the wikitext of revision 764138197
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/revision/764138197'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}


response = requests.get(url, headers=headers)
data = response.json()

print(data)
```

PHP

```
<?php
/*
Get the wikitext of revision 764138197
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/revision/764138197";


$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```
/*
    Get the wikitext of revision 764138197
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/revision/764138197";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}


  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/revision/764138197)

Parameters

`id` required | path | string	Revision ID

Responses

200	Success: Revision found. Returns a revision.
404	Revision not found

Get revision HTML

Path	`revision/{id}/html`	**Method**	`GET`
Content type	`text/HTML`	**Returns**	Revision

Returns HTML for an individual revision.

Examples

curl

```
# Get HTML of revision 764138197
$ curl "https://en.wikipedia.org/w/rest.php/v1/revision/764138197/html"
```

Python

```python
# Get HTML of revision 764138197
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/revision/764138197/html'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}


response = requests.get(url, headers=headers)
data = response.json()

print(data)
```

PHP

```php
<?php
/*
Get HTML of revision 764138197
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/revision/764138197/html";


$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```javascript
/*
    Get HTML of revision 764138197
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/revision/764138197/html";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}


  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/revision/764138197/html)

Parameters

id required | path | string	Revision ID

Responses

200	Success: Revision found. Returns the revision content as HTML.
404	Revision not found

Get revision information with HTML

Path	`revision/{id}/with_html`	**Method**	`GET`
Content type	`application/json`	**Returns**	Revision

Returns HTML and meta-data for an individual revision.

Examples

curl

```
# Get information about revision 764138197 along with rendered HTML
$ curl "https://en.wikipedia.org/w/rest.php/v1/revision/764138197/with_html"
```

Python

```
# Get information about revision 764138197 along with rendered HTML
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/revision/764138197/with_html'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}


response = requests.get(url, headers=headers)
data = response.json()

print(data)
```

PHP

```
<?php
/*
Get information about revision 764138197 along with rendered HTML
*/
$url = "https://en.wikipedia.org/w/rest.php/v1/revision/764138197/with_html";


$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```
/*
    Get information about revision 764138197 along with rendered HTML
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/revision/764138197/with_html";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}


  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/revision/764138197/with_html)

Parameters

id required | path | string	Revision ID

Responses

200	Revision found. Returns a revision.
404	Revision not found

Compare revisions

Route	`revision/{from}/compare/{to}`	**Content type**	`application/json`
Methods	`GET`	**Returns**	Wikidiff2 JSON diff format

Returns data that lets you display a line-by-line comparison of two revisions. (See an example (https://en.wikipedia.beta.wmflabs.org/w/index.php?diff=388864&oldid=388863&title=Main_Page&type=revision).) Only text-based wiki pages can be compared.

To use this endpoint on your wiki
Install Wikidiff2 1.9.0 or later.

Examples

curl

```
# Compare revision 847170467 to 851733941
$ curl "https://en.wikipedia.org/w/rest.php/v1/revision/847170467/compare/851733941"
```

Python

```
# Compare revision 847170467 to 851733941
import requests

url = 'https://en.wikipedia.org/w/rest.php/v1/revision/847170467/compare/851733941'
headers = {
    'User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'
}


response = requests.get(url, headers=headers)
data = response.json()

print(data)
```

PHP

```
<?php
/*
Compare revision 847170467 to 851733941
*/
```

```
$url = "https://en.wikipedia.org/w/rest.php/v1/revision/847170467/compare/851733941";


$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch, CURLOPT_USERAGENT, "MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)" );
$output = curl_exec( $ch );
curl_close( $ch );

echo($output);
?>
```

JavaScript

```
/*
    Compare revision 847170467 to 851733941
*/
async function doFetch() {
  let url = "https://en.wikipedia.org/w/rest.php/v1/revision/847170467/compare/851733941";
  let headers = {'Api-User-Agent': 'MediaWiki REST API docs examples/0.1
(https://www.mediawiki.org/wiki/API_talk:REST_API)'}


  const rsp = await fetch(url, headers);
  const data = await rsp.json();
  return data;
}

async function fetchAsync()
{
  try {
    let result = await doFetch();
    console.log(result);
  } catch( err ) {
    console.error( err.message );
  }
}

fetchAsync();
```

Try it out (https://en.wikipedia.org/w/rest.php/v1/revision/847170467/compare/851733941)

Parameters

`from` required | path	Revision identifier to use as the base for comparison
`to` required | path	Revision identifier to compare to the base

Responses

200	Success: Revisions found
400	Revision IDs are for different pages or for pages that can't be compared
400	Invalid content type
403	Revision not publicly accessible
404	Revision not found
500	Wikidiff2 extension 1.9.0 or later not installed

Response schema

`from` required | object	Information about the base revision used in the comparison
`to` required | object	Information about the revision being compared to the base revision

`from.id`
`to.id`
required | integer
	Revision identifier

`from.slot_role`
`to.slot_role`
required | string
	Area of the page being compared, usually `main`

`from.sections`
`to.sections`
required | array
	Array of objects representing section headings, including:
	▪ `level` (integer): Heading level, 1 through 4
	▪ `heading` (string): Text of the heading line, in wikitext
	▪ `offset` (integer): Location of the heading, in bytes from the beginning of the page

`diff` required | array of objects	Each object in the `diff` array represents a line in a visual, line-by-line comparison between the two revisions.
`diff.type` required | integer	The type of change represented by the diff object, either: ▪ `0`: A line with the same content in both revisions, included to provide context when viewing the diff. The API returns up to two context lines around each change. ▪ 1: A line included in the `to` revision but not in the `from` revision. ▪ 2: A line included in the `from` revision but not in the `to` revision. ▪ 3: A line containing text that differs between the two revisions. (For changes to paragraph location as well as content, see type 5.) ▪ 4: When a paragraph's location differs between the two revisions, a type 4 object represents the location in the `from` revision. ▪ 5: When a paragraph's location differs between the two revisions, a type 5 object represents the location in the `to` revision. This type can also include word-level differences between the two revisions.
`diff.lineNumber` optional | integer	The line number of the change based on the `to` revision.
`diff.text` required | string	The text of the line, including content from both revisions. For a line containing text that differs between the two revisions, you can use `highlightRanges` to visually indicate added and removed text. For a line containing a new line, the API returns the text as `""` (empty string).

`diff.highlightRanges` optional | array of objects	An array of objects that indicate where and in what style text should be highlighted to visually represent changes. Each object includes: ▪ `start` (integer): Where the highlighted text should start, in the number of bytes from the beginning of the line. ▪ `length` (integer): The length of the highlighted section, in bytes. ▪ `type` (integer): The type of highlight: ▪ `0` indicates an addition. ▪ `1` indicates a deletion.
`diff.moveInfo` optional | object	Visual indicators to use when a paragraph's location differs between the two revisions. `moveInfo` objects occur in pairs within the diff. ▪ `id` (string): The ID of the paragraph described by the diff object. ▪ `linkId` (string): The ID of the corresponding paragraph. ▪ For type 4 diff objects, `linkId` represents the location in the `to` revision. ▪ For type 5 diff objects, `linkId` represents the location in the `from` revision. ▪ `linkDirection` (integer): A visual indicator of the relationship between the two locations. You can use this property to display an arrow icon within the diff. ▪ `0` indicates that the `linkId` paragraph is lower on the page than the `id` paragraph. ▪ `1` indicates that the `linkId` paragraph is higher on the page than the `id` paragraph.
`diff.offset` required | object	The location of the line in bytes from the beginning of the page, including: ▪ `from` (integer): The first byte of the line in the `from` revision. A `null` value indicates that the line doesn't exist in the `from` revision. ▪ `to` (integer): The first byte of the line in the `to` revision. A `null` value indicates that the line doesn't exist in the `to` revision.