# Guzzle Upgrade Guide

## 6.0 to 7.0

In order to take advantage of the new features of PHP, Guzzle dropped the support of PHP 5. The minimum supported PHP version is now PHP 7.2. Type hints and return types for functions and methods have been added wherever possible.

Please make sure:

* You are calling a function or a method with the correct type.
* If you extend a class of Guzzle; update all signatures on methods you override.

#### Other backwards compatibility breaking changes

* Class GuzzleHttp\UriTemplate is removed.
* Class GuzzleHttp\Exception\SeekException is removed.
* Classes GuzzleHttp\Exception\BadResponseException, GuzzleHttp\Exception\ClientException, GuzzleHttp\Exception\ServerException can no longer be initialized with an empty Response as argument.
* Class GuzzleHttp\Exception\ConnectException now extends GuzzleHttp\Exception\TransferException instead of GuzzleHttp\Exception\RequestException.
* Function GuzzleHttp\Exception\ConnectException::getResponse() is removed.
* Function GuzzleHttp\Exception\ConnectException::hasResponse() is removed.
* Constant GuzzleHttp\ClientInterface::VERSION is removed. Added GuzzleHttp\ClientInterface::MAJOR\_VERSION instead.
* Function GuzzleHttp\Exception\RequestException::getResponseBodySummary is removed. Use \GuzzleHttp\Psr7\get\_message\_body\_summary as an alternative.
* Function GuzzleHttp\Cookie\CookieJar::getCookieValue is removed.
* Request option exceptions is removed. Please use http\_errors.
* Request option save\_to is removed. Please use sink.
* Pool option pool\_size is removed. Please use concurrency.
* We now look for environment variables in the $\_SERVER super global, due to thread safety issues with getenv. We continue to fallback to getenv in CLI environments, for maximum compatibility.
* The get, head, put, post, patch, delete, getAsync, headAsync, putAsync, postAsync, patchAsync, and deleteAsync methods are now implemented as genuine methods on GuzzleHttp\Client, with strong typing. The original \_\_call implementation remains unchanged for now, for maximum backwards compatibility, but won't be invoked under normal operation.
* The log middleware will log the errors with level error instead of notice
* Support for international domain names (IDN) is now disabled by default, and enabling it requires installing ext-intl, linked against a modern version of the C library (ICU 4.6 or higher).

#### Native functions calls

All internal native functions calls of Guzzle are now prefixed with a slash. This change makes it impossible for method overloading by other libraries or applications. Example:

// Before:

curl\_version();

// After:

\curl\_version();

For the full diff you can check [here](https://github.com/guzzle/guzzle/compare/6.5.4..master).

## 5.0 to 6.0

Guzzle now uses [PSR-7](https://www.php-fig.org/psr/psr-7/) for HTTP messages. Due to the fact that these messages are immutable, this prompted a refactoring of Guzzle to use a middleware based system rather than an event system. Any HTTP message interaction (e.g., GuzzleHttp\Message\Request) need to be updated to work with the new immutable PSR-7 request and response objects. Any event listeners or subscribers need to be updated to become middleware functions that wrap handlers (or are injected into a GuzzleHttp\HandlerStack).

* Removed GuzzleHttp\BatchResults
* Removed GuzzleHttp\Collection
* Removed GuzzleHttp\HasDataTrait
* Removed GuzzleHttp\ToArrayInterface
* The guzzlehttp/streams dependency has been removed. Stream functionality is now present in the GuzzleHttp\Psr7 namespace provided by the guzzlehttp/psr7 package.
* Guzzle no longer uses ReactPHP promises and now uses the guzzlehttp/promises library. We use a custom promise library for three significant reasons:
  1. React promises (at the time of writing this) are recursive. Promise chaining and promise resolution will eventually blow the stack. Guzzle promises are not recursive as they use a sort of trampolining technique. Note: there has been movement in the React project to modify promises to no longer utilize recursion.
  2. Guzzle needs to have the ability to synchronously block on a promise to wait for a result. Guzzle promises allows this functionality (and does not require the use of recursion).
  3. Because we need to be able to wait on a result, doing so using React promises requires wrapping react promises with RingPHP futures. This overhead is no longer needed, reducing stack sizes, reducing complexity, and improving performance.
* GuzzleHttp\Mimetypes has been moved to a function in GuzzleHttp\Psr7\mimetype\_from\_extension and GuzzleHttp\Psr7\mimetype\_from\_filename.
* GuzzleHttp\Query and GuzzleHttp\QueryParser have been removed. Query strings must now be passed into request objects as strings, or provided to the query request option when creating requests with clients. The query option uses PHP's http\_build\_query to convert an array to a string. If you need a different serialization technique, you will need to pass the query string in as a string. There are a couple helper functions that will make working with query strings easier: GuzzleHttp\Psr7\parse\_query and GuzzleHttp\Psr7\build\_query.
* Guzzle no longer has a dependency on RingPHP. Due to the use of a middleware system based on PSR-7, using RingPHP and it's middleware system as well adds more complexity than the benefits it provides. All HTTP handlers that were present in RingPHP have been modified to work directly with PSR-7 messages and placed in the GuzzleHttp\Handler namespace. This significantly reduces complexity in Guzzle, removes a dependency, and improves performance. RingPHP will be maintained for Guzzle 5 support, but will no longer be a part of Guzzle 6.
* As Guzzle now uses a middleware based systems the event system and RingPHP integration has been removed. Note: while the event system has been removed, it is possible to add your own type of event system that is powered by the middleware system.
  + Removed the Event namespace.
  + Removed the Subscriber namespace.
  + Removed Transaction class
  + Removed RequestFsm
  + Removed RingBridge
  + GuzzleHttp\Subscriber\Cookie is now provided by GuzzleHttp\Middleware::cookies
  + GuzzleHttp\Subscriber\HttpError is now provided by GuzzleHttp\Middleware::httpError
  + GuzzleHttp\Subscriber\History is now provided by GuzzleHttp\Middleware::history
  + GuzzleHttp\Subscriber\Mock is now provided by GuzzleHttp\Handler\MockHandler
  + GuzzleHttp\Subscriber\Prepare is now provided by GuzzleHttp\PrepareBodyMiddleware
  + GuzzleHttp\Subscriber\Redirect is now provided by GuzzleHttp\RedirectMiddleware
* Guzzle now uses Psr\Http\Message\UriInterface (implements in GuzzleHttp\Psr7\Uri) for URI support. GuzzleHttp\Url is now gone.
* Static functions in GuzzleHttp\Utils have been moved to namespaced functions under the GuzzleHttp namespace. This requires either a Composer based autoloader or you to include functions.php.
* GuzzleHttp\ClientInterface::getDefaultOption has been renamed to GuzzleHttp\ClientInterface::getConfig.
* GuzzleHttp\ClientInterface::setDefaultOption has been removed.
* The json and xml methods of response objects has been removed. With the migration to strictly adhering to PSR-7 as the interface for Guzzle messages, adding methods to message interfaces would actually require Guzzle messages to extend from PSR-7 messages rather then work with them directly.

## Migrating to middleware

The change to PSR-7 unfortunately required significant refactoring to Guzzle due to the fact that PSR-7 messages are immutable. Guzzle 5 relied on an event system from plugins. The event system relied on mutability of HTTP messages and side effects in order to work. With immutable messages, you have to change your workflow to become more about either returning a value (e.g., functional middlewares) or setting a value on an object. Guzzle v6 has chosen the functional middleware approach.

Instead of using the event system to listen for things like the before event, you now create a stack based middleware function that intercepts a request on the way in and the promise of the response on the way out. This is a much simpler and more predictable approach than the event system and works nicely with PSR-7 middleware. Due to the use of promises, the middleware system is also asynchronous.

v5:

use GuzzleHttp\Event\BeforeEvent;

$client = new GuzzleHttp\Client();

// Get the emitter and listen to the before event.

$client->getEmitter()->on('before', function (BeforeEvent $e) {

// Guzzle v5 events relied on mutation

$e->getRequest()->setHeader('X-Foo', 'Bar');

});

v6:

In v6, you can modify the request before it is sent using the mapRequest middleware. The idiomatic way in v6 to modify the request/response lifecycle is to setup a handler middleware stack up front and inject the handler into a client.

use GuzzleHttp\Middleware;

// Create a handler stack that has all of the default middlewares attached

$handler = GuzzleHttp\HandlerStack::create();

// Push the handler onto the handler stack

$handler->push(Middleware::mapRequest(function (RequestInterface $request) {

// Notice that we have to return a request object

return $request->withHeader('X-Foo', 'Bar');

}));

// Inject the handler into the client

$client = new GuzzleHttp\Client(['handler' => $handler]);

## POST Requests

This version added the [form\_params](https://docs.guzzlephp.org/en/latest/request-options.html#form_params) and multipart request options. form\_params is an associative array of strings or array of strings and is used to serialize an application/x-www-form-urlencoded POST request. The [multipart](https://docs.guzzlephp.org/en/latest/request-options.html#multipart) option is now used to send a multipart/form-data POST request.

GuzzleHttp\Post\PostFile has been removed. Use the multipart option to add POST files to a multipart/form-data request.

The body option no longer accepts an array to send POST requests. Please use multipart or form\_params instead.

The base\_url option has been renamed to base\_uri.

## 4.x to 5.0

## Rewritten Adapter Layer

Guzzle now uses [RingPHP](https://ringphp.readthedocs.org/en/latest) to send HTTP requests. The adapter option in a GuzzleHttp\Client constructor is still supported, but it has now been renamed to handler. Instead of passing a GuzzleHttp\Adapter\AdapterInterface, you must now pass a PHP callable that follows the RingPHP specification.

## Removed Fluent Interfaces

[Fluent interfaces were removed](https://ocramius.github.io/blog/fluent-interfaces-are-evil/) from the following classes:

* GuzzleHttp\Collection
* GuzzleHttp\Url
* GuzzleHttp\Query
* GuzzleHttp\Post\PostBody
* GuzzleHttp\Cookie\SetCookie

## Removed functions.php

Removed "functions.php", so that Guzzle is truly PSR-4 compliant. The following functions can be used as replacements.

* GuzzleHttp\json\_decode -> GuzzleHttp\Utils::jsonDecode
* GuzzleHttp\get\_path -> GuzzleHttp\Utils::getPath
* GuzzleHttp\Utils::setPath -> GuzzleHttp\set\_path
* GuzzleHttp\Pool::batch -> GuzzleHttp\batch. This function is, however, deprecated in favor of using GuzzleHttp\Pool::batch().

The "procedural" global client has been removed with no replacement (e.g., GuzzleHttp\get(), GuzzleHttp\post(), etc.). Use a GuzzleHttp\Client object as a replacement.

## throwImmediately has been removed

The concept of "throwImmediately" has been removed from exceptions and error events. This control mechanism was used to stop a transfer of concurrent requests from completing. This can now be handled by throwing the exception or by cancelling a pool of requests or each outstanding future request individually.

## headers event has been removed

Removed the "headers" event. This event was only useful for changing the body a response once the headers of the response were known. You can implement a similar behavior in a number of ways. One example might be to use a FnStream that has access to the transaction being sent. For example, when the first byte is written, you could check if the response headers match your expectations, and if so, change the actual stream body that is being written to.

## Updates to HTTP Messages

Removed the asArray parameter from GuzzleHttp\Message\MessageInterface::getHeader. If you want to get a header value as an array, then use the newly added getHeaderAsArray() method of MessageInterface. This change makes the Guzzle interfaces compatible with the PSR-7 interfaces.

## 3.x to 4.0

## Overarching changes:

* Now requires PHP 5.4 or greater.
* No longer requires cURL to send requests.
* Guzzle no longer wraps every exception it throws. Only exceptions that are recoverable are now wrapped by Guzzle.
* Various namespaces have been removed or renamed.
* No longer requiring the Symfony EventDispatcher. A custom event dispatcher based on the Symfony EventDispatcher is now utilized in GuzzleHttp\Event\EmitterInterface (resulting in significant speed and functionality improvements).

Changes per Guzzle 3.x namespace are described below.

## Batch

The Guzzle\Batch namespace has been removed. This is best left to third-parties to implement on top of Guzzle's core HTTP library.

## Cache

The Guzzle\Cache namespace has been removed. (Todo: No suitable replacement has been implemented yet, but hoping to utilize a PSR cache interface).

## Common

* Removed all of the wrapped exceptions. It's better to use the standard PHP library for unrecoverable exceptions.
* FromConfigInterface has been removed.
* Guzzle\Common\Version has been removed. The VERSION constant can be found at GuzzleHttp\ClientInterface::VERSION.

### Collection

* getAll has been removed. Use toArray to convert a collection to an array.
* inject has been removed.
* keySearch has been removed.
* getPath no longer supports wildcard expressions. Use something better like JMESPath for this.
* setPath now supports appending to an existing array via the [] notation.

### Events

Guzzle no longer requires Symfony's EventDispatcher component. Guzzle now uses GuzzleHttp\Event\Emitter.

* Symfony\Component\EventDispatcher\EventDispatcherInterface is replaced by GuzzleHttp\Event\EmitterInterface.
* Symfony\Component\EventDispatcher\EventDispatcher is replaced by GuzzleHttp\Event\Emitter.
* Symfony\Component\EventDispatcher\Event is replaced by GuzzleHttp\Event\Event, and Guzzle now has an EventInterface in GuzzleHttp\Event\EventInterface.
* AbstractHasDispatcher has moved to a trait, HasEmitterTrait, and HasDispatcherInterface has moved to HasEmitterInterface. Retrieving the event emitter of a request, client, etc. now uses the getEmitter method rather than the getDispatcher method.

#### Emitter

* Use the once() method to add a listener that automatically removes itself the first time it is invoked.
* Use the listeners() method to retrieve a list of event listeners rather than the getListeners() method.
* Use emit() instead of dispatch() to emit an event from an emitter.
* Use attach() instead of addSubscriber() and detach() instead of removeSubscriber().

$mock = new Mock();

// 3.x

$request->getEventDispatcher()->addSubscriber($mock);

$request->getEventDispatcher()->removeSubscriber($mock);

// 4.x

$request->getEmitter()->attach($mock);

$request->getEmitter()->detach($mock);

Use the on() method to add a listener rather than the addListener() method.

// 3.x

$request->getEventDispatcher()->addListener('foo', function (Event $event) { /\* ... \*/ } );

// 4.x

$request->getEmitter()->on('foo', function (Event $event, $name) { /\* ... \*/ } );

## Http

### General changes

* The cacert.pem certificate has been moved to src/cacert.pem.
* Added the concept of adapters that are used to transfer requests over the wire.
* Simplified the event system.
* Sending requests in parallel is still possible, but batching is no longer a concept of the HTTP layer. Instead, you must use the complete and error events to asynchronously manage parallel request transfers.
* Guzzle\Http\Url has moved to GuzzleHttp\Url.
* Guzzle\Http\QueryString has moved to GuzzleHttp\Query.
* QueryAggregators have been rewritten so that they are simply callable functions.
* GuzzleHttp\StaticClient has been removed. Use the functions provided in functions.php for an easy to use static client instance.
* Exceptions in GuzzleHttp\Exception have been updated to all extend from GuzzleHttp\Exception\TransferException.

### Client

Calling methods like get(), post(), head(), etc. no longer create and return a request, but rather creates a request, sends the request, and returns the response.

// 3.0

$request = $client->get('/');

$response = $request->send();

// 4.0

$response = $client->get('/');

// or, to mirror the previous behavior

$request = $client->createRequest('GET', '/');

$response = $client->send($request);

GuzzleHttp\ClientInterface has changed.

* The send method no longer accepts more than one request. Use sendAll to send multiple requests in parallel.
* setUserAgent() has been removed. Use a default request option instead. You could, for example, do something like: $client->setConfig('defaults/headers/User-Agent', 'Foo/Bar ' . $client::getDefaultUserAgent()).
* setSslVerification() has been removed. Use default request options instead, like $client->setConfig('defaults/verify', true).

GuzzleHttp\Client has changed.

* The constructor now accepts only an associative array. You can include a base\_url string or array to use a URI template as the base URL of a client. You can also specify a defaults key that is an associative array of default request options. You can pass an adapter to use a custom adapter, batch\_adapter to use a custom adapter for sending requests in parallel, or a message\_factory to change the factory used to create HTTP requests and responses.
* The client no longer emits a client.create\_request event.
* Creating requests with a client no longer automatically utilize a URI template. You must pass an array into a creational method (e.g., createRequest, get, put, etc.) in order to expand a URI template.

### Messages

Messages no longer have references to their counterparts (i.e., a request no longer has a reference to it's response, and a response no loger has a reference to its request). This association is now managed through a GuzzleHttp\Adapter\TransactionInterface object. You can get references to these transaction objects using request events that are emitted over the lifecycle of a request.

#### Requests with a body

* GuzzleHttp\Message\EntityEnclosingRequest and GuzzleHttp\Message\EntityEnclosingRequestInterface have been removed. The separation between requests that contain a body and requests that do not contain a body has been removed, and now GuzzleHttp\Message\RequestInterface handles both use cases.
* Any method that previously accepts a GuzzleHttp\Response object now accept a GuzzleHttp\Message\ResponseInterface.
* GuzzleHttp\Message\RequestFactoryInterface has been renamed to GuzzleHttp\Message\MessageFactoryInterface. This interface is used to create both requests and responses and is implemented in GuzzleHttp\Message\MessageFactory.
* POST field and file methods have been removed from the request object. You must now use the methods made available to GuzzleHttp\Post\PostBodyInterface to control the format of a POST body. Requests that are created using a standard GuzzleHttp\Message\MessageFactoryInterface will automatically use a GuzzleHttp\Post\PostBody body if the body was passed as an array or if the method is POST and no body is provided.

$request = $client->createRequest('POST', '/');

$request->getBody()->setField('foo', 'bar');

$request->getBody()->addFile(new PostFile('file\_key', fopen('/path/to/content', 'r')));

#### Headers

* GuzzleHttp\Message\Header has been removed. Header values are now simply represented by an array of values or as a string. Header values are returned as a string by default when retrieving a header value from a message. You can pass an optional argument of true to retrieve a header value as an array of strings instead of a single concatenated string.
* GuzzleHttp\PostFile and GuzzleHttp\PostFileInterface have been moved to GuzzleHttp\Post. This interface has been simplified and now allows the addition of arbitrary headers.
* Custom headers like GuzzleHttp\Message\Header\Link have been removed. Most of the custom headers are now handled separately in specific subscribers/plugins, and GuzzleHttp\Message\HeaderValues::parseParams() has been updated to properly handle headers that contain parameters (like the Link header).

#### Responses

* GuzzleHttp\Message\Response::getInfo() and GuzzleHttp\Message\Response::setInfo() have been removed. Use the event system to retrieve this type of information.
* GuzzleHttp\Message\Response::getRawHeaders() has been removed.
* GuzzleHttp\Message\Response::getMessage() has been removed.
* GuzzleHttp\Message\Response::calculateAge() and other cache specific methods have moved to the CacheSubscriber.
* Header specific helper functions like getContentMd5() have been removed. Just use getHeader('Content-MD5') instead.
* GuzzleHttp\Message\Response::setRequest() and GuzzleHttp\Message\Response::getRequest() have been removed. Use the event system to work with request and response objects as a transaction.
* GuzzleHttp\Message\Response::getRedirectCount() has been removed. Use the Redirect subscriber instead.
* GuzzleHttp\Message\Response::isSuccessful() and other related methods have been removed. Use getStatusCode() instead.

#### Streaming responses

Streaming requests can now be created by a client directly, returning a GuzzleHttp\Message\ResponseInterface object that contains a body stream referencing an open PHP HTTP stream.

// 3.0

use Guzzle\Stream\PhpStreamRequestFactory;

$request = $client->get('/');

$factory = new PhpStreamRequestFactory();

$stream = $factory->fromRequest($request);

$data = $stream->read(1024);

// 4.0

$response = $client->get('/', ['stream' => true]);

// Read some data off of the stream in the response body

$data = $response->getBody()->read(1024);

#### Redirects

The configureRedirects() method has been removed in favor of a allow\_redirects request option.

// Standard redirects with a default of a max of 5 redirects

$request = $client->createRequest('GET', '/', ['allow\_redirects' => true]);

// Strict redirects with a custom number of redirects

$request = $client->createRequest('GET', '/', [

'allow\_redirects' => ['max' => 5, 'strict' => true]

]);

#### EntityBody

EntityBody interfaces and classes have been removed or moved to GuzzleHttp\Stream. All classes and interfaces that once required GuzzleHttp\EntityBodyInterface now require GuzzleHttp\Stream\StreamInterface. Creating a new body for a request no longer uses GuzzleHttp\EntityBody::factory but now uses GuzzleHttp\Stream\Stream::factory or even better: GuzzleHttp\Stream\create().

* Guzzle\Http\EntityBodyInterface is now GuzzleHttp\Stream\StreamInterface
* Guzzle\Http\EntityBody is now GuzzleHttp\Stream\Stream
* Guzzle\Http\CachingEntityBody is now GuzzleHttp\Stream\CachingStream
* Guzzle\Http\ReadLimitEntityBody is now GuzzleHttp\Stream\LimitStream
* Guzzle\Http\IoEmittyinEntityBody has been removed.

#### Request lifecycle events

Requests previously submitted a large number of requests. The number of events emitted over the lifecycle of a request has been significantly reduced to make it easier to understand how to extend the behavior of a request. All events emitted during the lifecycle of a request now emit a custom GuzzleHttp\Event\EventInterface object that contains context providing methods and a way in which to modify the transaction at that specific point in time (e.g., intercept the request and set a response on the transaction).

* request.before\_send has been renamed to before and now emits a GuzzleHttp\Event\BeforeEvent
* request.complete has been renamed to complete and now emits a GuzzleHttp\Event\CompleteEvent.
* request.sent has been removed. Use complete.
* request.success has been removed. Use complete.
* error is now an event that emits a GuzzleHttp\Event\ErrorEvent.
* request.exception has been removed. Use error.
* request.receive.status\_line has been removed.
* curl.callback.progress has been removed. Use a custom StreamInterface to maintain a status update.
* curl.callback.write has been removed. Use a custom StreamInterface to intercept writes.
* curl.callback.read has been removed. Use a custom StreamInterface to intercept reads.

headers is a new event that is emitted after the response headers of a request have been received before the body of the response is downloaded. This event emits a GuzzleHttp\Event\HeadersEvent.

You can intercept a request and inject a response using the intercept() event of a GuzzleHttp\Event\BeforeEvent, GuzzleHttp\Event\CompleteEvent, and GuzzleHttp\Event\ErrorEvent event.

See: <https://docs.guzzlephp.org/en/latest/events.html>

## Inflection

The Guzzle\Inflection namespace has been removed. This is not a core concern of Guzzle.

## Iterator

The Guzzle\Iterator namespace has been removed.

* Guzzle\Iterator\AppendIterator, Guzzle\Iterator\ChunkedIterator, and Guzzle\Iterator\MethodProxyIterator are nice, but not a core requirement of Guzzle itself.
* Guzzle\Iterator\FilterIterator is no longer needed because an equivalent class is shipped with PHP 5.4.
* Guzzle\Iterator\MapIterator is not really needed when using PHP 5.5 because it's easier to just wrap an iterator in a generator that maps values.

For a replacement of these iterators, see <https://github.com/nikic/iter>

## Log

The LogPlugin has moved to <https://github.com/guzzle/log-subscriber>. The Guzzle\Log namespace has been removed. Guzzle now relies on Psr\Log\LoggerInterface for all logging. The MessageFormatter class has been moved to GuzzleHttp\Subscriber\Log\Formatter.

## Parser

The Guzzle\Parser namespace has been removed. This was previously used to make it possible to plug in custom parsers for cookies, messages, URI templates, and URLs; however, this level of complexity is not needed in Guzzle so it has been removed.

* Cookie: Cookie parsing logic has been moved to GuzzleHttp\Cookie\SetCookie::fromString.
* Message: Message parsing logic for both requests and responses has been moved to GuzzleHttp\Message\MessageFactory::fromMessage. Message parsing is only used in debugging or deserializing messages, so it doesn't make sense for Guzzle as a library to add this level of complexity to parsing messages.
* UriTemplate: URI template parsing has been moved to GuzzleHttp\UriTemplate. The Guzzle library will automatically use the PECL URI template library if it is installed.
* Url: URL parsing is now performed in GuzzleHttp\Url::fromString (previously it was Guzzle\Http\Url::factory()). If custom URL parsing is necessary, then developers are free to subclass GuzzleHttp\Url.

## Plugin

The Guzzle\Plugin namespace has been renamed to GuzzleHttp\Subscriber. Several plugins are shipping with the core Guzzle library under this namespace.

* GuzzleHttp\Subscriber\Cookie: Replaces the old CookiePlugin. Cookie jar code has moved to GuzzleHttp\Cookie.
* GuzzleHttp\Subscriber\History: Replaces the old HistoryPlugin.
* GuzzleHttp\Subscriber\HttpError: Throws errors when a bad HTTP response is received.
* GuzzleHttp\Subscriber\Mock: Replaces the old MockPlugin.
* GuzzleHttp\Subscriber\Prepare: Prepares the body of a request just before sending. This subscriber is attached to all requests by default.
* GuzzleHttp\Subscriber\Redirect: Replaces the RedirectPlugin.

The following plugins have been removed (third-parties are free to re-implement these if needed):

* GuzzleHttp\Plugin\Async has been removed.
* GuzzleHttp\Plugin\CurlAuth has been removed.
* GuzzleHttp\Plugin\ErrorResponse\ErrorResponsePlugin has been removed. This functionality should instead be implemented with event listeners that occur after normal response parsing occurs in the guzzle/command package.

The following plugins are not part of the core Guzzle package, but are provided in separate repositories:

* Guzzle\Http\Plugin\BackoffPlugin has been rewritten to be much simpler to build custom retry policies using simple functions rather than various chained classes. See: <https://github.com/guzzle/retry-subscriber>
* Guzzle\Http\Plugin\Cache\CachePlugin has moved to <https://github.com/guzzle/cache-subscriber>
* Guzzle\Http\Plugin\Log\LogPlugin has moved to <https://github.com/guzzle/log-subscriber>
* Guzzle\Http\Plugin\Md5\Md5Plugin has moved to <https://github.com/guzzle/message-integrity-subscriber>
* Guzzle\Http\Plugin\Mock\MockPlugin has moved to GuzzleHttp\Subscriber\MockSubscriber.
* Guzzle\Http\Plugin\Oauth\OauthPlugin has moved to <https://github.com/guzzle/oauth-subscriber>

## Service

The service description layer of Guzzle has moved into two separate packages:

* <https://github.com/guzzle/command> Provides a high level abstraction over web services by representing web service operations using commands.
* <https://github.com/guzzle/guzzle-services> Provides an implementation of guzzle/command that provides request serialization and response parsing using Guzzle service descriptions.

## Stream

Stream have moved to a separate package available at <https://github.com/guzzle/streams>.

Guzzle\Stream\StreamInterface has been given a large update to cleanly take on the responsibilities of Guzzle\Http\EntityBody and Guzzle\Http\EntityBodyInterface now that they have been removed. The number of methods implemented by the StreamInterface has been drastically reduced to allow developers to more easily extend and decorate stream behavior.

## Removed methods from StreamInterface

* getStream and setStream have been removed to better encapsulate streams.
* getMetadata and setMetadata have been removed in favor of GuzzleHttp\Stream\MetadataStreamInterface.
* getWrapper, getWrapperData, getStreamType, and getUri have all been removed. This data is accessible when using streams that implement GuzzleHttp\Stream\MetadataStreamInterface.
* rewind has been removed. Use seek(0) for a similar behavior.

## Renamed methods

* detachStream has been renamed to detach.
* feof has been renamed to eof.
* ftell has been renamed to tell.
* readLine has moved from an instance method to a static class method of GuzzleHttp\Stream\Stream.

## Metadata streams

GuzzleHttp\Stream\MetadataStreamInterface has been added to denote streams that contain additional metadata accessible via getMetadata(). GuzzleHttp\Stream\StreamInterface::getMetadata and GuzzleHttp\Stream\StreamInterface::setMetadata have been removed.

## StreamRequestFactory

The entire concept of the StreamRequestFactory has been removed. The way this was used in Guzzle 3 broke the actual interface of sending streaming requests (instead of getting back a Response, you got a StreamInterface). Streaming PHP requests are now implemented through the GuzzleHttp\Adapter\StreamAdapter.

## 3.6 to 3.7

### Deprecations

* You can now enable E\_USER\_DEPRECATED warnings to see if you are using any deprecated methods.:

\Guzzle\Common\Version::$emitWarnings = true;

The following APIs and options have been marked as deprecated:

* Marked Guzzle\Http\Message\Request::isResponseBodyRepeatable() as deprecated. Use $request->getResponseBody()->isRepeatable() instead.
* Marked Guzzle\Http\Message\Request::canCache() as deprecated. Use Guzzle\Plugin\Cache\DefaultCanCacheStrategy->canCacheRequest() instead.
* Marked Guzzle\Http\Message\Request::canCache() as deprecated. Use Guzzle\Plugin\Cache\DefaultCanCacheStrategy->canCacheRequest() instead.
* Marked Guzzle\Http\Message\Request::setIsRedirect() as deprecated. Use the HistoryPlugin instead.
* Marked Guzzle\Http\Message\Request::isRedirect() as deprecated. Use the HistoryPlugin instead.
* Marked Guzzle\Cache\CacheAdapterFactory::factory() as deprecated
* Marked Guzzle\Service\Client::enableMagicMethods() as deprecated. Magic methods can no longer be disabled on a Guzzle\Service\Client.
* Marked Guzzle\Parser\Url\UrlParser as deprecated. Just use PHP's parse\_url() and percent encode your UTF-8.
* Marked Guzzle\Common\Collection::inject() as deprecated.
* Marked Guzzle\Plugin\CurlAuth\CurlAuthPlugin as deprecated. Use $client->getConfig()->setPath('request.options/auth', array('user', 'pass', 'Basic|Digest|NTLM|Any')); or $client->setDefaultOption('auth', array('user', 'pass', 'Basic|Digest|NTLM|Any'));

3.7 introduces request.options as a parameter for a client configuration and as an optional argument to all creational request methods. When paired with a client's configuration settings, these options allow you to specify default settings for various aspects of a request. Because these options make other previous configuration options redundant, several configuration options and methods of a client and AbstractCommand have been deprecated.

* Marked Guzzle\Service\Client::getDefaultHeaders() as deprecated. Use $client->getDefaultOption('headers').
* Marked Guzzle\Service\Client::setDefaultHeaders() as deprecated. Use $client->setDefaultOption('headers/{header\_name}', 'value').
* Marked 'request.params' for Guzzle\Http\Client as deprecated. Use $client->setDefaultOption('params/{param\_name}', 'value')
* Marked 'command.headers', 'command.response\_body' and 'command.on\_complete' as deprecated for AbstractCommand. These will work through Guzzle 4.0

$command = $client->getCommand('foo', array(

'command.headers' => array('Test' => '123'),

'command.response\_body' => '/path/to/file'

));

// Should be changed to:

$command = $client->getCommand('foo', array(

'command.request\_options' => array(

'headers' => array('Test' => '123'),

'save\_as' => '/path/to/file'

)

));

### Interface changes

Additions and changes (you will need to update any implementations or subclasses you may have created):

* Added an $options argument to the end of the following methods of Guzzle\Http\ClientInterface: createRequest, head, delete, put, patch, post, options, prepareRequest
* Added an $options argument to the end of Guzzle\Http\Message\Request\RequestFactoryInterface::createRequest()
* Added an applyOptions() method to Guzzle\Http\Message\Request\RequestFactoryInterface
* Changed Guzzle\Http\ClientInterface::get($uri = null, $headers = null, $body = null) to Guzzle\Http\ClientInterface::get($uri = null, $headers = null, $options = array()). You can still pass in a resource, string, or EntityBody into the $options parameter to specify the download location of the response.
* Changed Guzzle\Common\Collection::\_\_construct($data) to no longer accepts a null value for $data but a default array()
* Added Guzzle\Stream\StreamInterface::isRepeatable
* Made Guzzle\Http\Client::expandTemplate and getUriTemplate protected methods.

The following methods were removed from interfaces. All of these methods are still available in the concrete classes that implement them, but you should update your code to use alternative methods:

* Removed Guzzle\Http\ClientInterface::setDefaultHeaders(). Use $client->getConfig()->setPath('request.options/headers/{header\_name}', 'value'). or $client->getConfig()->setPath('request.options/headers', array('header\_name' => 'value'))or$client->setDefaultOption('headers/{header\_name}', 'value'). or $client->setDefaultOption('headers', array('header\_name' => 'value'))`.
* Removed Guzzle\Http\ClientInterface::getDefaultHeaders(). Use $client->getConfig()->getPath('request.options/headers')`.
* Removed Guzzle\Http\ClientInterface::expandTemplate(). This is an implementation detail.
* Removed Guzzle\Http\ClientInterface::setRequestFactory(). This is an implementation detail.
* Removed Guzzle\Http\ClientInterface::getCurlMulti(). This is a very specific implementation detail.
* Removed Guzzle\Http\Message\RequestInterface::canCache. Use the CachePlugin.
* Removed Guzzle\Http\Message\RequestInterface::setIsRedirect. Use the HistoryPlugin.
* Removed Guzzle\Http\Message\RequestInterface::isRedirect. Use the HistoryPlugin.

### Cache plugin breaking changes

* CacheKeyProviderInterface and DefaultCacheKeyProvider are no longer used. All of this logic is handled in a CacheStorageInterface. These two objects and interface will be removed in a future version.
* Always setting X-cache headers on cached responses
* Default cache TTLs are now handled by the CacheStorageInterface of a CachePlugin
* CacheStorageInterface::cache($key, Response $response, $ttl = null) has changed to cache(RequestInterface $request, Response $response);
* CacheStorageInterface::fetch($key) has changed to fetch(RequestInterface $request);
* CacheStorageInterface::delete($key) has changed to delete(RequestInterface $request);
* Added CacheStorageInterface::purge($url)
* DefaultRevalidation::\_\_construct(CacheKeyProviderInterface $cacheKey, CacheStorageInterface $cache, CachePlugin $plugin) has changed to DefaultRevalidation::\_\_construct(CacheStorageInterface $cache, CanCacheStrategyInterface $canCache = null)
* Added RevalidationInterface::shouldRevalidate(RequestInterface $request, Response $response)

## 3.5 to 3.6

* Mixed casing of headers are now forced to be a single consistent casing across all values for that header.
* Messages internally use a HeaderCollection object to delegate handling case-insensitive header resolution
* Removed the whole changedHeader() function system of messages because all header changes now go through addHeader(). For example, setHeader() first removes the header using unset on a HeaderCollection and then calls addHeader(). Keeping the Host header and URL host in sync is now handled by overriding the addHeader method in Request.
* Specific header implementations can be created for complex headers. When a message creates a header, it uses a HeaderFactory which can map specific headers to specific header classes. There is now a Link header and CacheControl header implementation.
* Moved getLinks() from Response to just be used on a Link header object.

If you previously relied on Guzzle\Http\Message\Header::raw(), then you will need to update your code to use the HeaderInterface (e.g. toArray(), getAll(), etc.).

### Interface changes

* Removed from interface: Guzzle\Http\ClientInterface::setUriTemplate
* Removed from interface: Guzzle\Http\ClientInterface::setCurlMulti()
* Removed Guzzle\Http\Message\Request::receivedRequestHeader() and implemented this functionality in Guzzle\Http\Curl\RequestMediator
* Removed the optional $asString parameter from MessageInterface::getHeader(). Just cast the header to a string.
* Removed the optional $tryChunkedTransfer option from Guzzle\Http\Message\EntityEnclosingRequestInterface
* Removed the $asObjects argument from Guzzle\Http\Message\MessageInterface::getHeaders()

### Removed deprecated functions

* Removed Guzzle\Parser\ParserRegister::get(). Use getParser()
* Removed Guzzle\Parser\ParserRegister::set(). Use registerParser().

### Deprecations

* The ability to case-insensitively search for header values
* Guzzle\Http\Message\Header::hasExactHeader
* Guzzle\Http\Message\Header::raw. Use getAll()
* Deprecated cache control specific methods on Guzzle\Http\Message\AbstractMessage. Use the CacheControl header object instead.

### Other changes

* All response header helper functions return a string rather than mixing Header objects and strings inconsistently
* Removed cURL blacklist support. This is no longer necessary now that Expect, Accept, etc. are managed by Guzzle directly via interfaces
* Removed the injecting of a request object onto a response object. The methods to get and set a request still exist but are a no-op until removed.
* Most classes that used to require a Guzzle\Service\Command\CommandInterface typehint now request a Guzzle\Service\Command\ArrayCommandInterface.
* Added Guzzle\Http\Message\RequestInterface::startResponse() to the RequestInterface to handle injecting a response on a request while the request is still being transferred
* Guzzle\Service\Command\CommandInterface now extends from ToArrayInterface and ArrayAccess

## 3.3 to 3.4

Base URLs of a client now follow the rules of <https://datatracker.ietf.org/doc/html/rfc3986#section-5.2.2> when merging URLs.

## 3.2 to 3.3

### Response::getEtag() quote stripping removed

Guzzle\Http\Message\Response::getEtag() no longer strips quotes around the ETag response header

### Removed Guzzle\Http\Utils

The Guzzle\Http\Utils class was removed. This class was only used for testing.

### Stream wrapper and type

Guzzle\Stream\Stream::getWrapper() and Guzzle\Stream\Stream::getStreamType() are no longer converted to lowercase.

### curl.emit\_io became emit\_io

Emitting IO events from a RequestMediator is now a parameter that must be set in a request's curl options using the 'emit\_io' key. This was previously set under a request's parameters using 'curl.emit\_io'

## 3.1 to 3.2

### CurlMulti is no longer reused globally

Before 3.2, the same CurlMulti object was reused globally for each client. This can cause issue where plugins added to a single client can pollute requests dispatched from other clients.

If you still wish to reuse the same CurlMulti object with each client, then you can add a listener to the ServiceBuilder's service\_builder.create\_client event to inject a custom CurlMulti object into each client as it is created.

$multi = new Guzzle\Http\Curl\CurlMulti();

$builder = Guzzle\Service\Builder\ServiceBuilder::factory('/path/to/config.json');

$builder->addListener('service\_builder.create\_client', function ($event) use ($multi) {

$event['client']->setCurlMulti($multi);

}

});

### No default path

URLs no longer have a default path value of '/' if no path was specified.

Before:

$request = $client->get('http://www.foo.com');

echo $request->getUrl();

// >> http://www.foo.com/

After:

$request = $client->get('http://www.foo.com');

echo $request->getUrl();

// >> http://www.foo.com

### Less verbose BadResponseException

The exception message for Guzzle\Http\Exception\BadResponseException no longer contains the full HTTP request and response information. You can, however, get access to the request and response object by calling getRequest() or getResponse() on the exception object.

### Query parameter aggregation

Multi-valued query parameters are no longer aggregated using a callback function. Guzzle\Http\Query now has a setAggregator() method that accepts a Guzzle\Http\QueryAggregator\QueryAggregatorInterface object. This object is responsible for handling the aggregation of multi-valued query string variables into a flattened hash.

## 2.8 to 3.x

### Guzzle\Service\Inspector

Change \Guzzle\Service\Inspector::fromConfig to \Guzzle\Common\Collection::fromConfig

**Before**

use Guzzle\Service\Inspector;

class YourClient extends \Guzzle\Service\Client

{

public static function factory($config = array())

{

$default = array();

$required = array('base\_url', 'username', 'api\_key');

$config = Inspector::fromConfig($config, $default, $required);

$client = new self(

$config->get('base\_url'),

$config->get('username'),

$config->get('api\_key')

);

$client->setConfig($config);

$client->setDescription(ServiceDescription::factory(\_\_DIR\_\_ . DIRECTORY\_SEPARATOR . 'client.json'));

return $client;

}

**After**

use Guzzle\Common\Collection;

class YourClient extends \Guzzle\Service\Client

{

public static function factory($config = array())

{

$default = array();

$required = array('base\_url', 'username', 'api\_key');

$config = Collection::fromConfig($config, $default, $required);

$client = new self(

$config->get('base\_url'),

$config->get('username'),

$config->get('api\_key')

);

$client->setConfig($config);

$client->setDescription(ServiceDescription::factory(\_\_DIR\_\_ . DIRECTORY\_SEPARATOR . 'client.json'));

return $client;

}

### Convert XML Service Descriptions to JSON

**Before**

<?xml version="1.0" encoding="UTF-8"?>

<client>

<commands>

<!-- Groups -->

<command name="list\_groups" method="GET" uri="groups.json">

<doc>Get a list of groups</doc>

</command>

<command name="search\_groups" method="GET" uri='search.json?query="{{query}} type:group"'>

<doc>Uses a search query to get a list of groups</doc>

<param name="query" type="string" required="true" />

</command>

<command name="create\_group" method="POST" uri="groups.json">

<doc>Create a group</doc>

<param name="data" type="array" location="body" filters="json\_encode" doc="Group JSON"/>

<param name="Content-Type" location="header" static="application/json"/>

</command>

<command name="delete\_group" method="DELETE" uri="groups/{{id}}.json">

<doc>Delete a group by ID</doc>

<param name="id" type="integer" required="true"/>

</command>

<command name="get\_group" method="GET" uri="groups/{{id}}.json">

<param name="id" type="integer" required="true"/>

</command>

<command name="update\_group" method="PUT" uri="groups/{{id}}.json">

<doc>Update a group</doc>

<param name="id" type="integer" required="true"/>

<param name="data" type="array" location="body" filters="json\_encode" doc="Group JSON"/>

<param name="Content-Type" location="header" static="application/json"/>

</command>

</commands>

</client>

**After**

{

"name": "Zendesk REST API v2",

"apiVersion": "2012-12-31",

"description":"Provides access to Zendesk views, groups, tickets, ticket fields, and users",

"operations": {

"list\_groups": {

"httpMethod":"GET",

"uri": "groups.json",

"summary": "Get a list of groups"

},

"search\_groups":{

"httpMethod":"GET",

"uri": "search.json?query=\"{query} type:group\"",

"summary": "Uses a search query to get a list of groups",

"parameters":{

"query":{

"location": "uri",

"description":"Zendesk Search Query",

"type": "string",

"required": true

}

}

},

"create\_group": {

"httpMethod":"POST",

"uri": "groups.json",

"summary": "Create a group",

"parameters":{

"data": {

"type": "array",

"location": "body",

"description":"Group JSON",

"filters": "json\_encode",

"required": true

},

"Content-Type":{

"type": "string",

"location":"header",

"static": "application/json"

}

}

},

"delete\_group": {

"httpMethod":"DELETE",

"uri": "groups/{id}.json",

"summary": "Delete a group",

"parameters":{

"id":{

"location": "uri",

"description":"Group to delete by ID",

"type": "integer",

"required": true

}

}

},

"get\_group": {

"httpMethod":"GET",

"uri": "groups/{id}.json",

"summary": "Get a ticket",

"parameters":{

"id":{

"location": "uri",

"description":"Group to get by ID",

"type": "integer",

"required": true

}

}

},

"update\_group": {

"httpMethod":"PUT",

"uri": "groups/{id}.json",

"summary": "Update a group",

"parameters":{

"id": {

"location": "uri",

"description":"Group to update by ID",

"type": "integer",

"required": true

},

"data": {

"type": "array",

"location": "body",

"description":"Group JSON",

"filters": "json\_encode",

"required": true

},

"Content-Type":{

"type": "string",

"location":"header",

"static": "application/json"

}

}

}

}

### Guzzle\Service\Description\ServiceDescription

Commands are now called Operations

**Before**

use Guzzle\Service\Description\ServiceDescription;

$sd = new ServiceDescription();

$sd->getCommands(); // @returns ApiCommandInterface[]

$sd->hasCommand($name);

$sd->getCommand($name); // @returns ApiCommandInterface|null

$sd->addCommand($command); // @param ApiCommandInterface $command

**After**

use Guzzle\Service\Description\ServiceDescription;

$sd = new ServiceDescription();

$sd->getOperations(); // @returns OperationInterface[]

$sd->hasOperation($name);

$sd->getOperation($name); // @returns OperationInterface|null

$sd->addOperation($operation); // @param OperationInterface $operation

### Guzzle\Common\Inflection\Inflector

Namespace is now Guzzle\Inflection\Inflector

### Guzzle\Http\Plugin

Namespace is now Guzzle\Plugin. Many other changes occur within this namespace and are detailed in their own sections below.

### Guzzle\Http\Plugin\LogPlugin and Guzzle\Common\Log

Now Guzzle\Plugin\Log\LogPlugin and Guzzle\Log respectively.

**Before**

use Guzzle\Common\Log\ClosureLogAdapter;

use Guzzle\Http\Plugin\LogPlugin;

/\*\* @var \Guzzle\Http\Client \*/

$client;

// $verbosity is an integer indicating desired message verbosity level

$client->addSubscriber(new LogPlugin(new ClosureLogAdapter(function($m) { echo $m; }, $verbosity = LogPlugin::LOG\_VERBOSE);

**After**

use Guzzle\Log\ClosureLogAdapter;

use Guzzle\Log\MessageFormatter;

use Guzzle\Plugin\Log\LogPlugin;

/\*\* @var \Guzzle\Http\Client \*/

$client;

// $format is a string indicating desired message format -- @see MessageFormatter

$client->addSubscriber(new LogPlugin(new ClosureLogAdapter(function($m) { echo $m; }, $format = MessageFormatter::DEBUG\_FORMAT);

### Guzzle\Http\Plugin\CurlAuthPlugin

Now Guzzle\Plugin\CurlAuth\CurlAuthPlugin.

### Guzzle\Http\Plugin\ExponentialBackoffPlugin

Now Guzzle\Plugin\Backoff\BackoffPlugin, and other changes.

**Before**

use Guzzle\Http\Plugin\ExponentialBackoffPlugin;

$backoffPlugin = new ExponentialBackoffPlugin($maxRetries, array\_merge(

ExponentialBackoffPlugin::getDefaultFailureCodes(), array(429)

));

$client->addSubscriber($backoffPlugin);

**After**

use Guzzle\Plugin\Backoff\BackoffPlugin;

use Guzzle\Plugin\Backoff\HttpBackoffStrategy;

// Use convenient factory method instead -- see implementation for ideas of what

// you can do with chaining backoff strategies

$backoffPlugin = BackoffPlugin::getExponentialBackoff($maxRetries, array\_merge(

HttpBackoffStrategy::getDefaultFailureCodes(), array(429)

));

$client->addSubscriber($backoffPlugin);

### Known Issues

#### [BUG] Accept-Encoding header behavior changed unintentionally.

(See #217) (Fixed in 09daeb8c666fb44499a0646d655a8ae36456575e)

In version 2.8 setting the Accept-Encoding header would set the CURLOPT\_ENCODING option, which permitted cURL to properly handle gzip/deflate compressed responses from the server. In versions affected by this bug this does not happen. See issue #217 for a workaround, or use a version containing the fix.