scrapy-zyte-api

Release 0.18.2

Zyte Group Ltd

May 02, 2024

SETUP

Scrapy plugin for seamless Zyte API integration.

After the *initial setup*, you can use Zyte API automatically, either *globally* or *per request*, or *manually per request*.

CHAPTER

ONE

INITIAL SETUP

Learn how to get scrapy-zyte-api installed and configured on an existing Scrapy project.

Tip: Zyte's web scraping tutorial covers scrapy-zyte-api setup as well.

1.1 Requirements

You need at least:

- A Zyte API subscription (there's a free trial).
- Python 3.8+
- Scrapy 2.0.1+

scrapy-poet integration requires higher versions:

- Scrapy 2.6+

1.2 Installation

For a basic installation:

```
pip install scrapy-zyte-api
```

For *scrapy-poet integration*:

```
pip install scrapy-zyte-api[provider]
```

1.3 Configuration

To configure scrapy-zyte-api, *set your API key* and either *enable the add-on* (Scrapy 2.10) or *configure all components separately*.

Warning: *Changing reactors may require code changes*.

1.3.1 Setting your API key

Add your Zyte API key, and add it to your project `settings.py`:

```
ZYTE_API_KEY = "YOUR_API_KEY"
```

Alternatively, you can set your API key in the `ZYTE_API_KEY` environment variable instead.

1.3.2 Enabling the add-on

If you are using Scrapy 2.10 or higher, you can set up scrapy-zyte-api integration using the following add-on with any priority:

Listing 1: settings.py

```
ADDONS = {
    "scrapy_zyte_api.Addon": 500,
}
```

Note: The addon enables *transparent mode* by default.

1.3.3 Enabling all components separately

If *enabling the add-on* is not an option, you can set up scrapy-zyte-api integration as follows:

Listing 2: settings.py

```
DOWNLOAD_HANDLERS = {
    "http": "scrapy_zyte_api.ScrapyZyteAPIDownloadHandler",
    "https": "scrapy_zyte_api.ScrapyZyteAPIDownloadHandler",
}
DOWNLOADER_MIDDLEWARES = {
    "scrapy_zyte_api.ScrapyZyteAPIDownloaderMiddleware": 1000,
}
SPIDER_MIDDLEWARES = {
    "scrapy_zyte_api.ScrapyZyteAPISpiderMiddleware": 100,
}
REQUEST_FINGERPRINTER_CLASS = "scrapy_zyte_api.ScrapyZyteAPIRequestFingerprinter"
TWISTED_REACTOR = "twisted.internet.asyncioreactor.AsyncioSelectorReactor"
```

By default, scrapy-zyte-api doesn't change the spider behavior. To switch your spider to use Zyte API for all requests, set the following setting as well:

Listing 3: settings.py

```
ZYTE_API_TRANSPARENT_MODE = True
```

For *scrapy-poet integration*, add the following provider to the `SCRAPY_POET_PROVIDERS` setting:

Listing 4: settings.py

```
SCRAPY_POET_PROVIDERS = {
    "scrapy_zyte_api.providers.ZyteApiProvider": 1100,
}
```

If you already had a custom value for REQUEST_FINGERPRINTER_CLASS, set that value on *ZYTE_API_FALLBACK_REQUEST_FINGERPRINTER_CLASS* instead.

Listing 5: settings.py

```
ZYTE_API_FALLBACK_REQUEST_FINGERPRINTER_CLASS = "myproject.CustomRequestFingerprinter"
```

1.4 Changing reactors may require code changes

If your TWISTED_REACTOR setting was not set to `"twisted.internet.asyncioreactor.AsyncioSelectorReactor"` before, you will be changing the Twisted reactor that your Scrapy project uses, and your existing code may need changes, such as:

- Handling a pre-installed reactor.

  Some Twisted imports install the default, non-asyncio Twisted reactor as a side effect. Once a reactor is installed, it cannot be changed for the whole run time.

- Awaiting on Deferreds.

  Note that you might be using Deferreds without realizing it through some Scrapy functions and methods. For example, when you yield the return value of `self.crawler.engine.download()` from a spider callback, you are yielding a Deferred.

CHAPTER

TWO

TRANSPARENT MODE

Set *ZYTE_API_TRANSPARENT_MODE* to `True` to handle requests as follows:

- By default, requests are sent with *automatic request parameters*.
- Requests with *zyte_api* set to a `dict` are sent with *manual request parameters*.
- Requests with *zyte_api_automap* set to `False` are *not* sent through Zyte API.

For example:

```
import scrapy


class SampleQuotesSpider(scrapy.Spider):
    name = "sample_quotes"
    start_urls = ["https://quotes.toscrape.com/"]

    custom_settings = {
        "ZYTE_API_TRANSPARENT_MODE": True,
    }

    def parse(self, response):
        print(response.text)
        # "<html>...</html>"
```

CHAPTER

THREE

MANUAL REQUEST PARAMETERS

To send a Scrapy request through Zyte API with manually-defined Zyte API request parameters, define your parameters in the *zyte_api* key in `Request.meta` as a `dict`.

The only exception is the url parameter, which should not be defined as a Zyte API parameter. The value from `Request.url` is used automatically.

For example:

```
import scrapy


class SampleQuotesSpider(scrapy.Spider):
    name = "sample_quotes"

    def start_requests(self):
        yield scrapy.Request(
            url="https://quotes.toscrape.com/",
            meta={
                "zyte_api": {
                    "browserHtml": True,
                }
            },
        )

    def parse(self, response):
        print(response.text)
        # "<html>...</html>"
```

Note that response headers are necessary for raw response decoding. When defining parameters manually and requesting httpResponseBody, remember to also request httpResponseHeaders:

```
import scrapy


class SampleQuotesSpider(scrapy.Spider):
    name = "sample_quotes"

    def start_requests(self):
        yield scrapy.Request(
            url="https://quotes.toscrape.com/",
            meta={
                "zyte_api": {
```

(continues on next page)

(continued from previous page)

```
                    "httpResponseBody": True,
                    "httpResponseHeaders": True,
                }
            },
        )

    def parse(self, response):
        print(response.text)
        # "<html>...</html>"
```

To learn more about Zyte API parameters, see the upstream usage and API reference pages.

CHAPTER

FOUR

AUTOMATIC REQUEST PARAMETERS

To send a Scrapy request through Zyte API letting Zyte API request parameters be automatically chosen based on the parameters of that Scrapy request, set the *zyte_api_automap* key in `Request.meta` to `True`.

For example:

```
import scrapy


class SampleQuotesSpider(scrapy.Spider):
    name = "sample_quotes"

    def start_requests(self):
        yield scrapy.Request(
            url="https://quotes.toscrape.com/",
            meta={
                "zyte_api_automap": True,
            },
        )

    def parse(self, response):
        print(response.text)
        # "<html>...</html>"
```

In *transparent mode*, *zyte_api_automap* is `True` by default.

See *Request mapping* to learn how exactly request parameters are mapped when using automatic request parameters.

4.1 Changing parameters

You may set *zyte_api_automap* in `Request.meta` to a `dict` of Zyte API parameters to add, modify, or remove (by setting to `False`) automatic request parameters. This also works in *transparent mode*.

Enabling browserHtml, screenshot, or an automatic extraction property, unsets httpResponseBody and httpResponseHeaders, and makes `Request.headers` become requestHeaders instead of customHttpRequestHeaders. For example, the following Scrapy request:

```
Request(
    url="https://quotes.toscrape.com",
    headers={"Referer": "https://example.com/"},
    meta={"zyte_api_automap": {"browserHtml": True}},
)
```

Results in a request to the Zyte API data extraction endpoint with the following parameters:

```
{
    "browserHtml": true,
    "experimental": {
        "responseCookies": true
    },
    "requestHeaders": {"referer": "https://example.com/"},
    "url": "https://quotes.toscrape.com"
}
```

See also: *Unsupported scenarios*.

CHAPTER

FIVE

DEFAULT PARAMETERS

Often the same configuration needs to be used for all Zyte API requests. For example, all requests may need to set the same geolocation, or the spider only uses browserHtml requests.

The following settings allow you to define Zyte API parameters to be included in all requests:

- *ZYTE_API_AUTOMAP_PARAMS*, for *transparent mode* and *automatic request parameters*.
- *ZYTE_API_DEFAULT_PARAMS*, for *manual request parameters*.

For example, if you set *ZYTE_API_DEFAULT_PARAMS* to `{"geolocation": "US"}` and *zyte_api* to `{"browserHtml": True}`, `{"url": "...", "geolocation": "US", "browserHtml": True}` is sent to Zyte API.

CHAPTER

SIX

RETRIES

To make error handling easier, scrapy-zyte-api lets you *handle successful Zyte API responses as usual*, but *implements a more advanced retry mechanism for rate-limiting and unsuccessful responses*.

6.1 Retrying successful Zyte API responses

When a successful Zyte API response is received, a Scrapy response object is built based on the upstream website response (see *Response mapping*), and passed to your downloader middlewares and spider callback.

Usually, these responses do not need to be retried. If they do, you can retry them using Scrapy's built-in retry middleware (`RetryMiddleware`) or its `get_retry_request()` function.

6.2 Retrying non-successful Zyte API responses

When a rate-limiting or an unsuccessful Zyte API response is received, no Scrapy response object is built. Instead, a *retry policy* is followed, and if the policy retries are exhausted, a `zyte_api.RequestError` exception is raised.

That `zyte_api.RequestError` exception is passed to the `process_exception` method of your downloader middlewares and to your spider errback if you defined one for the request. And you could have `RetryMiddleware` retry that request by adding `zyte_api.RequestError` to the `RETRY_EXCEPTIONS` setting. But you are better off *relying on the default retry policy or defining a custom retry policy* instead.

6.3 Retry policy

Retry policies are a feature of the Python Zyte API client library, which scrapy-zyte-api uses underneath. See the upstream retry policy documentation to learn about the default retry policy and how to create a custom retry policy, including ready-to-use examples.

In scrapy-zyte-api, use the *`ZYTE_API_RETRY_POLICY`* setting or the *`zyte_api_retry_policy`* `Request.meta` key to point to a custom retry policy or to its import path, to override the default retry policy:

Listing 1: settings.py

```
ZYTE_API_RETRY_POLICY = "project.retry_policies.CUSTOM_RETRY_POLICY"
```

CHAPTER

SEVEN

SCRAPY-POET INTEGRATION

If during the *initial setup* you followed the required steps for scrapy-poet integration, you can request *supported page inputs* in your page objects:

```
@attrs.define
class ProductPage(BasePage):
    response: BrowserResponse
    product: Product


class ZyteApiSpider(scrapy.Spider):
    ...

    def parse_page(self, response: DummyResponse, page: ProductPage):
        ...
```

Or request them directly in the callback:

```
class ZyteApiSpider(scrapy.Spider):
    ...

    def parse_page(self,
                   response: DummyResponse,
                   browser_response: BrowserResponse,
                   product: Product,
                   ):
        ...
```

7.1 Dependency annotations

`ZyteApiProvider` understands and makes use of some dependency annotations.

Note: Dependency annotations require Python 3.9+.

7.1.1 Item annotations

Item dependencies such as `zyte_common_items.Product` can be annotated directly. The only currently supported annotation is *scrapy_zyte_api.ExtractFrom*:

```
from typing import Annotated

from scrapy_zyte_api import ExtractFrom


@attrs.define
class MyPageObject(BasePage):
    product: Annotated[Product, ExtractFrom.httpResponseBody]
```

The provider will set the extraction options based on the annotations, so for this code `extractFrom` will be set to `httpResponseBody` in `productOptions`.

7.1.2 Geolocation

You can specify the geolocation field by adding a *scrapy_zyte_api.Geolocation* dependency and annotating it with a country code:

```
from typing import Annotated

from scrapy_zyte_api import Geolocation


@attrs.define
class MyPageObject(BasePage):
    product: Product
    geolocation: Annotated[Geolocation, "DE"]
```

7.1.3 Browser actions

You can specify browser actions by adding a *scrapy_zyte_api.Actions* dependency and annotating it with actions passed to the *scrapy_zyte_api.actions()* function:

```
from typing import Annotated

from scrapy_zyte_api import Actions, actions


@attrs.define
class MyPageObject(BasePage):
    product: Product
    actions: Annotated[
        Actions,
        actions(
            [
                {
                    "action": "click",
                    "selector": {"type": "css", "value": "button#openDescription"},
```

(continues on next page)

(continued from previous page)

```
                    "delay": 0,
                    "button": "left",
                    "onError": "return",
                },
                {"action": "waitForTimeout", "timeout": 5, "onError": "return"},
            ]
        ),
    ]
```

You can access the results of these actions in the *Actions.results* attribute of the dependency in the resulting page object:

```
def validate_input(self):
    for action_result in self.actions.result:
        if action_result["status"] != "success":
            return Product(is_valid=False)
    return None
```

7.2 Custom parameters

scrapy-poet integration ignores both *manual* and *automatic* Zyte API parameters.

Whenever you can, use *inputs* and *dependency annotations* to get additional Zyte API parameters into Zyte API requests made by the scrapy-poet integration.

If that is not possible, you can add Zyte API parameters to requests made by the scrapy-poet integration with the *zyte_api_provider* request metadata key or the *ZYTE_API_PROVIDER_PARAMS* setting.

When *zyte_api_provider* or *ZYTE_API_PROVIDER_PARAMS* include one of the Zyte API extraction option parameters (e.g. `productOptions` for `product`), but the final Zyte API request does not include the corresponding extraction type, the unused options are automatically removed. So, it is safe to use *ZYTE_API_PROVIDER_PARAMS* to set the default options for various extraction types:

Listing 1: setting.py

```
ZYTE_API_PROVIDER_PARAMS = {
    "productOptions": {"extractFrom": "httpResponseBody"},
    "productNavigationOptions": {"extractFrom": "httpResponseBody"},
}
```

When both *zyte_api_provider* and *ZYTE_API_PROVIDER_PARAMS* are defined, they are combined, with *zyte_api_provider* taking precedence in case of conflict.

CHAPTER

EIGHT

STATS

Stats from python-zyte-api are exposed as Scrapy stats with the `scrapy-zyte-api` prefix.

For example, `scrapy-zyte-api/status_codes/<status code>` stats indicate the status code of Zyte API responses (e.g. `429` for rate limiting or `520` for temporary download errors).

Note: The actual status code that is received from the target website, i.e. the statusCode response field of a Zyte API successful response, is accounted for in the `downloader/response_status_count/<status code>` stat, as with any other Scrapy response.

CHAPTER

NINE

REQUEST FINGERPRINTING

The request fingerprinter class of scrapy-zyte-api ensures that Scrapy 2.7 and later generate unique request fingerprints for Zyte API requests *based on some of their parameters*.

For example, a request for browserHtml and a request for screenshot with the same target URL are considered different requests. Similarly, requests with the same target URL but different actions are also considered different requests.

Use *ZYTE_API_FALLBACK_REQUEST_FINGERPRINTER_CLASS* to define a custom request fingerprinting for requests that do not go through Zyte API.

9.1 Request fingerprinting before Scrapy 2.7

If you have a Scrapy version older than Scrapy 2.7, Zyte API parameters are not taken into account for request fingerprinting. This can cause some Scrapy components, like the filter of duplicate requests or the HTTP cache extension, to interpret 2 different requests as being the same.

To avoid most issues, use *automatic request parameters*, either through *transparent mode* or setting *zyte_api_automap* to `True` in `Request.meta`, and then use `Request` attributes instead of `Request.meta` as much as possible. Unlike `Request.meta`, `Request` attributes do affect request fingerprints in Scrapy versions older than Scrapy 2.7.

For requests that must have the same `Request` attributes but should still be considered different, such as browser-based requests with different URL fragments, you can set `dont_filter=True` when creating your request to prevent the duplicate filter of Scrapy to filter any of them out. For example:

```
yield Request(
    "https://toscrape.com#1",
    meta={"zyte_api_automap": {"browserHtml": True}},
    dont_filter=True,
)
yield Request(
    "https://toscrape.com#2",
    meta={"zyte_api_automap": {"browserHtml": True}},
    dont_filter=True,
)
```

Note, however, that for other Scrapy components, like the HTTP cache extensions, these 2 requests would still be considered identical.

CHAPTER

TEN

USING A PROXY

If you need a proxy to access Zyte API (e.g. a corporate proxy), configure the `HTTP_PROXY` and `HTTPS_PROXY` environment variables accordingly, and set the *ZYTE_API_USE_ENV_PROXY* setting to `True`.

CHAPTER

ELEVEN

REQUEST MAPPING

When you enable automatic request parameter mapping, be it through *transparent mode* or *for a specific request*, some Zyte API parameters are *chosen automatically for you*, and you can then *change them further* if you wish.

11.1 Automatic mapping

- `Request.url` becomes url, same as in *requests with manual parameters*.
- If `Request.method` is something other than `"GET"`, it becomes httpRequestMethod.
- `Request.body` becomes httpRequestBody.
- `Request.headers` become customHttpRequestHeaders for HTTP requests and requestHeaders for browser requests. See *Header mapping* and *Unsupported scenarios* for details.
- If *`ZYTE_API_EXPERIMENTAL_COOKIES_ENABLED`* is `True`, `COOKIES_ENABLED` is `True` (default), and `Request.meta` does not set `dont_merge_cookies` to `True`:
  - experimental.responseCookies becomes `True`.
  - Cookies from the `cookiejar` become experimental.requestCookies.

    All cookies from the cookie jar are set, regardless of their cookie domain. This is because Zyte API requests may involve requests to different domains (e.g. when following cross-domain redirects, or during browser rendering).

    See also: *`ZYTE_API_MAX_COOKIES`*, *`ZYTE_API_COOKIE_MIDDLEWARE`*.
- httpResponseBody and httpResponseHeaders are set to `True`.

  This is subject to change without prior notice in future versions of scrapy-zyte-api, so please account for the following:

  - If you are requesting a binary resource, such as a PDF file or an image file, set httpResponseBody to `True` explicitly in your requests:

    ```
    Request(
        url="https://toscrape.com/img/zyte.png",
        meta={
            "zyte_api_automap": {"httpResponseBody": True},
        },
    )
    ```

    In the future, we may stop setting httpResponseBody to `True` by default, and instead use a different, new Zyte API parameter that only works for non-binary responses (e.g. HMTL, JSON, plain text).

- If you need to access response headers, be it through *response.headers* or through *response.raw_api_response["httpResponseHeaders"]*, set httpResponseHeaders to `True` explicitly in your requests:

  ```
  Request(
      url="https://toscrape.com/",
      meta={
          "zyte_api_automap": {"httpResponseHeaders": True},
      },
  )
  ```

  At the moment scrapy-zyte-api requests response headers because some response headers are necessary to properly decode the response body as text. In the future, Zyte API may be able to handle this decoding automatically, so scrapy-zyte-api would stop setting httpResponseHeaders to `True` by default.

For example, the following Scrapy request:

```
Request(
    method="POST",
    url="https://httpbin.org/anything",
    headers={"Content-Type": "application/json"},
    body=b'{"foo": "bar"}',
    cookies={"a": "b"},
)
```

Results in a request to the Zyte API data extraction endpoint with the following parameters:

```
{
    "customHttpRequestHeaders": [
        {
            "name": "Content-Type",
            "value": "application/json"
        }
    ],
    "experimental": {
        "requestCookies": [
            {
                "name": "a",
                "value": "b",
                "domain": ""
            }
        ],
        "responseCookies": true
    },
    "httpResponseBody": true,
    "httpResponseHeaders": true,
    "httpRequestBody": "eyJmb28iOiAiYmFyIn0=",
    "httpRequestMethod": "POST",
    "url": "https://httpbin.org/anything"
}
```

11.2 Header mapping

When mapping headers, some headers are dropped based on the values of the *ZYTE_API_SKIP_HEADERS* and *ZYTE_API_BROWSER_HEADERS* settings. Their default values cause the drop of headers not supported by Zyte API.

Even if not defined in *ZYTE_API_SKIP_HEADERS*, additional headers may be dropped from HTTP requests (customHttpRequestHeaders):

- The `Accept` and `Accept-Language` headers are dropped if their values are not user-defined, i.e. they come from the default global value (setting `priority` of 0) of the `DEFAULT_REQUEST_HEADERS` setting.
- The `Accept-Encoding` header is dropped if its value is not user-defined, i.e. it was set by the `HttpCompressionMiddleware`.
- The `User-Agent` header is dropped if its value is not user-defined, i.e. it comes from the default global value (setting `priority` of 0) of the `USER_AGENT` setting.

To force the mapping of these headers, define the corresponding setting (if any), set them in the `DEFAULT_REQUEST_HEADERS` setting, or set them in `Request.headers` from a spider callback. They will be mapped even if defined with their default value.

Headers will also be mapped if set to a non-default value elsewhere, e.g. in a custom downloader middleware, as long as it is done before the scrapy-zyte-api downloader middleware, which is responsible for the mapping, processes the request. Here "before" means a lower value than `1000` in the `DOWNLOADER_MIDDLEWARES` setting.

Similarly, you can add any of those headers to the *ZYTE_API_SKIP_HEADERS* setting to prevent their mapping.

Also note that Scrapy sets the `Referer` header by default in all requests that come from spider callbacks. To unset the header on a given request, set the header value to `None` on that request. To unset it from all requests, set the `REFERER_ENABLED` setting to `False`. To unset it only from Zyte API requests, add it to the *ZYTE_API_SKIP_HEADERS* setting and remove it from the *ZYTE_API_BROWSER_HEADERS* setting.

11.3 Unsupported scenarios

To maximize support for potential future changes in Zyte API, automatic request parameter mapping allows some parameter values and parameter combinations that Zyte API does not currently support, and may never support:

- `Request.method` becomes httpRequestMethod even for unsupported httpRequestMethod values, and even if httpResponseBody is unset.
- You can set customHttpRequestHeaders or requestHeaders to `True` to force their mapping from `Request.headers` in scenarios where they would not be mapped otherwise.

  Conversely, you can set customHttpRequestHeaders or requestHeaders to `False` to prevent their mapping from `Request.headers`.
- `Request.body` becomes httpRequestBody even if httpResponseBody is unset.
- You can set httpResponseBody to `False` (which unsets the parameter), and not set other outputs (browserHtml, screenshot, product…) to `True`. In this case, `Request.headers` is mapped as requestHeaders.
- You can set httpResponseBody to `True` or use automatic extraction from httpResponseBody, and also set browserHtml or screenshot to `True` or use automatic extraction from browserHtml. In this case, `Request.headers` is mapped both as customHttpRequestHeaders and as requestHeaders, and browserHtml is used as *response.body*.

CHAPTER

TWELVE

RESPONSE MAPPING

12.1 Parameters

Zyte API response parameters are mapped into *response class* attributes where possible:

- url becomes *response.url*.
- statusCode becomes *response.status*.
- httpResponseHeaders and experimental.responseCookies become *response.headers*.
- experimental.responseCookies is also mapped into the request cookiejar.
- browserHtml and httpResponseBody are mapped into both *response.text* and *response.body*.

  If none of these parameters were present, e.g. if the only requested output was screenshot, *response.text* and *response.body* would be empty.

  If a future version of Zyte API supported requesting both outputs on the same request, and both parameters were present, browserHtml would be the one mapped into *response.text* and *response.body*.

Both *response classes* have a *response.raw_api_response* attribute that contains a dict with the complete, raw response from Zyte API, where you can find all Zyte API response parameters, including those that are not mapped into other response class attributes.

For example, for a request for httpResponseBody and httpResponseHeaders, you would get:

```
def parse(self, response):
    print(response.url)
    # "https://quotes.toscrape.com/"
    print(response.status)
    # 200
    print(response.headers)
    # {b"Content-Type": [b"text/html"], ...}
    print(response.text)
    # "<html>...</html>"
    print(response.body)
    # b"<html>...</html>"
    print(response.raw_api_response)
    # {
    #     "url": "https://quotes.toscrape.com/",
    #     "statusCode": 200,
    #     "httpResponseBody": "PGh0bWw+4oCmPC9odG1sPg==",
    #     "httpResponseHeaders": [...],
    # }
```

For a request for screenshot, on the other hand, the response would look as follows:

```
def parse(self, response):
    print(response.url)
    # "https://quotes.toscrape.com/"
    print(response.status)
    # 200
    print(response.headers)
    # {}
    print(response.text)
    # ""
    print(response.body)
    # b""
    print(response.raw_api_response)
    # {
    #     "url": "https://quotes.toscrape.com/",
    #     "statusCode": 200,
    #     "screenshot": "iVBORw0KGgoAAAANSUh...",
    # }
    from base64 import b64decode

    print(b64decode(response.raw_api_response["screenshot"]))
    # b'\x89PNG\r\n\x1a\n\x00\x00\x00\r...'
```

12.2 Classes

Zyte API responses are mapped with one of the following classes:

- *ZyteAPITextResponse* is used to map text responses, i.e. responses with browserHtml or responses with both httpResponseBody and httpResponseHeaders with a text body (e.g. plain text, HTML, JSON).
- *ZyteAPIResponse* is used to map any other response.

class scrapy_zyte_api.responses.**ZyteAPIResponse**(**args: Any*, ***kwargs: Any*)

Bases: ZyteAPIMixin, Response

url

status

headers

body: bytes

raw_api_response

Contains the raw API response from Zyte API.

For the full list of parameters, see Zyte API reference documentation.

class scrapy_zyte_api.responses.**ZyteAPITextResponse**(**args: Any*, ***kwargs: Any*)

Bases: ZyteAPIMixin, HtmlResponse

url

status

headers

body: bytes

text: str

raw_api_response

Contains the raw API response from Zyte API.

For the full list of parameters, see Zyte API reference documentation.

CHAPTER

THIRTEEN

SETTINGS

Settings for scrapy-zyte-api.

13.1 ZYTE_API_AUTOMAP_PARAMS

Default: `{}`

`dict` of parameters to be combined with *automatic request parameters*.

These parameters are merged with *`zyte_api_automap`* parameters. *`zyte_api_automap`* parameters take precedence.

This setting has no effect on requests with *manual request parameters*.

When using *transparent mode*, be careful of which parameters you define in this setting. In transparent mode, all Scrapy requests go through Zyte API, even requests that Scrapy sends automatically, such as those for `robots.txt` files when `ROBOTSTXT_OBEY` is `True`, or those for sitemaps when using `SitemapSpider`. Certain parameters, like browserHtml or screenshot, are not meant to be used for every single request.

If *`zyte_api_default_params`* in `Request.meta` is set to `False`, this setting is ignored for that request.

See *Default parameters*.

13.2 ZYTE_API_BROWSER_HEADERS

Default: `{"Referer": "referer"}`

Determines headers that *can* be mapped as requestHeaders.

It is a `dict`, where keys are header names and values are the key that represents them in requestHeaders.

13.3 ZYTE_API_COOKIE_MIDDLEWARE

Default: `scrapy.downloadermiddlewares.cookies.CookiesMiddleware`

If you are using a custom downloader middleware to handle request cookie jars, you can point this setting to its import path to make scrapy-zyte-api work with it.

Your cookie downloader middleware must have a `jars` property with the same signature as in the built-in Scrapy downloader middleware for cookie handling.

13.4 ZYTE_API_DEFAULT_PARAMS

Default: `{}`

`dict` of parameters to be combined with *manual request parameters*.

You may set `zyte_api` to an empty `dict` to only use the parameters defined here for that request.

These parameters are merged with `zyte_api` parameters. `zyte_api` parameters take precedence.

This setting has no effect on requests with *automatic request parameters*.

If `zyte_api_default_params` in `Request.meta` is set to `False`, this setting is ignored for that request.

See *Default parameters*.

13.5 ZYTE_API_ENABLED

Default: `True`

Can be set to `False` to disable scrapy-zyte-api.

13.6 ZYTE_API_EXPERIMENTAL_COOKIES_ENABLED

Default: `False`

See *Automatic mapping*.

13.7 ZYTE_API_FALLBACK_REQUEST_FINGERPRINTER_CLASS

Default: `scrapy_poet.ScrapyPoetRequestFingerprinter` if scrapy-poet is installed, else `scrapy.utils.request.RequestFingerprinter`

Request fingerprinter to for requests that do not go through Zyte API. See *Request fingerprinting*.

13.8 ZYTE_API_KEY

Default: `None`

Your Zyte API key.

You can alternatively define an environment variable with the same name.

Tip: On Scrapy Cloud, this setting is defined automatically.

13.9 ZYTE_API_LOG_REQUESTS

Default: `False`

Set this to `True` and `LOG_LEVEL` to `"DEBUG"` to enable the logging of debug messages that indicate the JSON object sent on every Zyte API request.

For example:

```
Sending Zyte API extract request: {"url": "https://example.com", "httpResponseBody":␣
↪true}
```

See also: *ZYTE_API_LOG_REQUESTS_TRUNCATE*.

13.10 ZYTE_API_LOG_REQUESTS_TRUNCATE

Default: `64`

Determines the maximum length of any string value in the JSON object logged when *ZYTE_API_LOG_REQUESTS* is enabled, excluding object keys.

To disable truncation, set this to `0`.

13.11 ZYTE_API_MAX_COOKIES

Default: `100`

If the cookies to be set during *request mapping* exceed this limit, a warning is logged, and only as many cookies as the limit allows are set for the target request.

To silence this warning, set experimental.requestCookies manually, e.g. to an empty `dict`.

Alternatively, if experimental.requestCookies starts supporting more than 100 cookies, update this setting accordingly.

13.12 ZYTE_API_MAX_REQUESTS

Default: `None`

When set to an integer value > 0, the spider will close when the number of Zyte API requests reaches it.

Note that requests with error responses that cannot be retried or exceed their retry limit also count here.

13.13 ZYTE_API_PROVIDER_PARAMS

Default: `{}`

Defines additional request parameters to use in Zyte API requests sent by the *scrapy-poet integration*.

For example:

Listing 1: settings.py

```
ZYTE_API_PROVIDER_PARAMS = {
    "requestCookies": [
        {"name": "a", "value": "b", "domain": "example.com"},
    ],
}
```

13.14 ZYTE_API_RETRY_POLICY

Default: `"zyte_api.aio.retry.zyte_api_retrying"`

Determines the retry policy for Zyte API requests.

It must be a string with the import path of a `tenacity.AsyncRetrying` subclass.

Note: Settings must be `picklable`, and retry policies are not, so you cannot assign a retry policy class directly to this setting, you must use their import path as a string instead.

See *Retries*.

13.15 ZYTE_API_SKIP_HEADERS

Default: `["Cookie"]`

Determines headers that must *not* be mapped as customHttpRequestHeaders.

13.16 ZYTE_API_TRANSPARENT_MODE

Default: `False`

See *Transparent mode*.

13.17 ZYTE_API_USE_ENV_PROXY

Default: `False`

Set to `True` to make Zyte API requests respect system proxy settings. See *Using a proxy*.

CHAPTER

FOURTEEN

REQUEST.META KEYS

Keys that can be defined in `Request.meta` for scrapy-zyte-api.

14.1 zyte_api

Default: `False`

See *Manual request parameters*.

14.2 zyte_api_automap

Default: *ZYTE_API_TRANSPARENT_MODE* (`False`)

See *Automatic request parameters*.

14.3 zyte_api_default_params

Default: `True`

If set to `False`, the values of *ZYTE_API_AUTOMAP_PARAMS* and *ZYTE_API_DEFAULT_PARAMS* are ignored for this request.

14.4 zyte_api_provider

Default: `{}`

Sets Zyte API parameters to include into requests made by the *scrapy-poet integration*.

For example:

```
Request(
    "https://example.com",
    meta={
        "zyte_api_provider": {
            "requestCookies": [
                {"name": "a", "value": "b", "domain": "example.com"},
            ],
```

(continues on next page)

(continued from previous page)

```
        }
    },
)
```

See also *ZYTE_API_PROVIDER_PARAMS*.

14.5 zyte_api_retry_policy

Default: *ZYTE_API_RETRY_POLICY* (`zyte_api.aio.retry.zyte_api_retrying`)

Determines the retry policy for Zyte API requests used to fulfill this request.

It must be a `tenacity.AsyncRetrying` subclass or its import path as a string.

Note: If you need your request to be serializable, e.g. to use `HttpCacheMiddleware`, you must specify the import path of your retry policy class as a string, because retry policies are not serializable.

See *Retries*.

CHAPTER

FIFTEEN

INPUTS

scrapy-poet integration, if enabled during the *initial setup*, allows obtaining the following inputs from web-poet and zyte-common-items through Zyte API:

- `web_poet.BrowserHtml`
- `web_poet.BrowserResponse`
- `web_poet.AnyResponse`

  This re-uses either `web_poet.BrowserResponse` *(takes priority)* or `web_poet.HttpResponse` if they're available.

  If neither is available, it would use `web_poet.HttpResponse` requested from Zyte API. However, if other item inputs (e.g. `zyte_common_items.Product`) are present, it would request `web_poet.BrowserResponse` from Zyte API unless an extraction source is provided.
- `zyte_common_items.Article`
- `zyte_common_items.ArticleList`
- `zyte_common_items.ArticleNavigation`
- `zyte_common_items.JobPosting`
- `zyte_common_items.Product`
- `zyte_common_items.ProductList`
- `zyte_common_items.ProductNavigation`

Additional inputs and input annotations are also provided:

15.1 Built-in inputs

class scrapy_zyte_api.**Actions**(*results: List[_ActionResult] | None*)

A page input that specifies browser actions and contains their results.

The actions must be *specified with an annotation* using `actions()`.

results: List[_ActionResult] | None

Results of actions.

class scrapy_zyte_api.**Geolocation**

A page input that forces a given geolocation for all other page inputs.

The target geolocation must be *specified with an annotation*.

class scrapy_zyte_api.**Screenshot**(*body: bytes*)

A container for holding the screenshot of a webpage.

body: bytes

Body.

15.2 Built-in input annotations

enum scrapy_zyte_api.**ExtractFrom**(*value*)

Annotation to specify the extraction source of an automatic extraction *input*, such as `Product` or `Article`.

See *Dependency annotations*.

Member Type
str

Valid values are as follows:

httpResponseBody: str = <ExtractFrom.httpResponseBody: 'httpResponseBody'>

browserHtml: str = <ExtractFrom.browserHtml: 'browserHtml'>

scrapy_zyte_api.**actions**(*value: Iterable[Action]*)

Convert an iterable of `Action` dicts into a hashable value.

CHAPTER

SIXTEEN

REQUEST FINGERPRINTING PARAMETERS

The request fingerprinter class of scrapy-zyte-api generates request fingerprints for Zyte API requests based on the following Zyte API parameters:

- url (`canonicalized`)

  For URLs that include a URL fragment, like `https://example.com#foo`, URL canonicalization keeps the URL fragment if browserHtml or screenshot are enabled, or if extractFrom is set to `browserHtml`.

- Request attribute parameters (httpRequestBody, httpRequestText, httpRequestMethod), except headers

  Equivalent httpRequestBody and httpRequestText values generate the same signature.

- Output parameters (browserHtml, httpResponseBody, httpResponseHeaders, responseCookies, screenshot, and automatic extraction outputs like product)

- Rendering option parameters (actions, device, javascript, screenshotOptions, viewport, and automatic extraction options like productOptions)

- geolocation

- echoData

The following Zyte API parameters are *not* taken into account for request fingerprinting:

- Request header parameters (customHttpRequestHeaders, requestHeaders)

- Request cookie parameters (cookieManagement, requestCookies)

- Session handling parameters (sessionContext, sessionContextParameters)

- jobId

- Experimental parameters (experimental.*)

CHAPTER

SEVENTEEN

CHANGES

17.1 0.18.2 (2024-04-25)

- The `Accept`, `Accept-Encoding`, `Accept-Language`, and `User-Agent` headers are now dropped automatically during *header mapping* unless they have user-defined values. This fix can improve success rates on some websites when using HTTP requests.

17.2 0.18.1 (2024-04-19)

- `extractFrom` in *`zyte_api_provider`* or *`ZYTE_API_PROVIDER_PARAMS`* overrides *`ExtractFrom`* annotations.

17.3 0.18.0 (2024-04-17)

- Updated requirement versions:
  - zyte-api >= 0.5.1
- A new *`zyte_api_provider`* request metadata key offers the same functionality as the *`ZYTE_API_PROVIDER_PARAMS`* setting on a per-request basis.
- Fixed support for nested dicts, tuples and lists when defining *browser actions*.

17.4 0.17.3 (2024-03-18)

- `scrapy_zyte_api.Addon` now adds `scrapy_zyte_api.providers.ZyteApiProvider` to the `SCRAPY_POET_PROVIDERS` scrapy-poet setting if scrapy-poet is installed.

17.5 0.17.2 (2024-03-14)

- Added a *scrapy_zyte_api.Actions* dependency.

17.6 0.17.1 (2024-03-11)

- Added a *scrapy_zyte_api.Screenshot* dependency.

17.7 0.17.0 (2024-03-05)

- Added support for Python 3.12.
- Updated requirement versions:
  - scrapy-poet >= 0.22.0
  - web-poet >= 0.17.0
- Added a Scrapy add-on, `scrapy_zyte_api.Addon`, which simplifies configuring Scrapy projects to work with `scrapy-zyte-api`.
- CI improvements.

17.8 0.16.1 (2024-02-23)

- Fix `"extractFrom": "httpResponseBody"` causing both customHttpRequestHeaders and requestHeaders, which are incompatible with each other, to be set when using automatic request mapping.

17.9 0.16.0 (2024-02-08)

- Removed support for Python 3.7.
- Updated requirement versions:
  - scrapy-poet >= 0.21.0
  - web-poet >= 0.16.0
- Added support for `web_poet.AnyResponse` dependency.
- Added support to specify the country code via `typing.Annotated` and *scrapy_zyte_api.Geolocation* dependency *(supported only on Python 3.9+)*.
- Improved tests.

17.10 0.15.0 (2024-01-31)

- Updated requirement versions:
  - scrapy-poet >= 0.20.1
- Dependency injection *through scrapy-poet* is now taken into account for request fingerprinting.

  Now, when scrapy-poet is installed, the default value of the *ZYTE_API_FALLBACK_REQUEST_FINGERPRINTER_CLASS* setting is `scrapy_poet.ScrapyPoetRequestFingerprinter`, and a warning will be issued if a custom value is not a subclass of `ScrapyPoetRequestFingerprinter`.
- Zyte Smart Proxy Manager special headers will now be dropped automatically when using *transparent mode* or *automatic request parameters*. Where possible, they will be replaced with equivalent Zyte API parameters. In all cases, a warning will be issued.
- Covered the configuration of `scrapy_zyte_api.ScrapyZyteAPISpiderMiddleware` in the *setup documentation*.

  `ScrapyZyteAPISpiderMiddleware` was added in scrapy-zyte-api 0.13.0, and is required to automatically close spiders when all start requests fail because they are pointing to domains forbidden by Zyte API.

17.11 0.14.1 (2024-01-17)

- The assignment of a custom download slot to requests that use Zyte API now also happens in the spider middleware, not only in the downloader middleware.

  This way requests get a download slot assigned before they reach the scheduler, making Zyte API requests work as expected with `scrapy.pqueues.DownloaderAwarePriorityQueue`.

  Note: New requests created from downloader middlewares do not get their download slot assigned before they reach the scheduler. So, unless they reuse the metadata from a requests that did get a download slot assigned (e.g. retries, redirects), they will continue not to work as expected with `DownloaderAwarePriorityQueue`.

17.12 0.14.0 (2024-01-15)

- Updated requirement versions:
  - andi >= 0.6.0
  - scrapy-poet >= 0.19.0
  - zyte-common-items >= 0.8.0
- Added support for `zyte_common_items.JobPosting` to the scrapy-poet provider.

17.13 0.13.0 (2023-12-13)

- Updated requirement versions:
  - andi >= 0.5.0
  - scrapy-poet >= 0.18.0
  - web-poet >= 0.15.1
  - zyte-api >= 0.4.8
- The spider is now closed and the finish reason is set to `"zyte_api_bad_key"` or `"zyte_api_suspended_account"` when receiving "Authentication Key Not Found" or "Account Suspended" responses from Zyte API.
- The spider is now closed and the finish reason is set to `"failed_forbidden_domain"` when all start requests fail because they are pointing to domains forbidden by Zyte API.
- The spider is now closed and the finish reason is set to `"plugin_conflict"` if both scrapy-zyte-smartproxy and the transparent mode of scrapy-zyte-api are enabled.
- The `extractFrom` extraction option can now be requested by annotating the dependency with a `scrapy_zyte_api.ExtractFrom` member (e.g. `product: typing.Annotated[Product, ExtractFrom.httpResponseBody]`).
- The `Set-Cookie` header is now removed from the response if the cookies were returned by Zyte API (as `"experimental.responseCookies"`).
- The request fingerprinting was improved by refining which parts of the request affect the fingerprint.
- Zyte API Request IDs are now included in the error logs.
- Split README.rst into multiple documentation files and publish them on ReadTheDocs.
- Improve the documentation for the `ZYTE_API_MAX_REQUESTS` setting.
- Test and CI improvements.

17.14 0.12.2 (2023-10-19)

- Unused `<data type>Options` (e.g. `productOptions`) are now dropped from `ZYTE_API_PROVIDER_PARAMS` when sending the Zyte API request
- When logging Zyte API requests, truncation now uses "…" instead of Unicode ellipsis.

17.15 0.12.1 (2023-09-29)

- The new `_ZYTE_API_USER_AGENT` setting allows customizing the user agent string reported to Zyte API.

  Note that this setting is only meant for libraries and frameworks built on top of scrapy-zyte-api, to report themselves to Zyte API, for client software tracking and monitoring purposes. The value of this setting is *not* the `User-Agent` header sent to upstream websites when using Zyte API.

17.16 0.12.0 (2023-09-26)

- A new `ZYTE_API_PROVIDER_PARAMS` setting allows setting Zyte API parameters, like `geolocation`, to be included in all Zyte API requests by the scrapy-poet provider.
- A new `scrapy-zyte-api/request_args/<parameter>` stat, counts the number of requests containing a given Zyte API request parameter. For example, `scrapy-zyte-api/request_args/url` counts the number of Zyte API requests with the URL parameter set (which should be all of them).

  Experimental is treated as a namespace, and its parameters are the ones counted, i.e. there is no `scrapy-zyte-api/request_args/experimental` stat, but there are stats like `scrapy-zyte-api/request_args/experimental.responseCookies`.

17.17 0.11.1 (2023-08-25)

- scrapy-zyte-api 0.11.0 accidentally increased the minimum required version of scrapy-poet from 0.10.0 to 0.11.0. We have reverted that change and implemented measures to prevent similar accidents in the future.
- Automatic parameter mapping no longer warns about dropping the `Accept-Encoding` header when the header value matches the Scrapy default.
- The README now mentions additional changes that may be necessary when switching Twisted reactors on existing projects.
- The README now explains how status codes, from Zyte API or from wrapped responses, are reflected in Scrapy stats.

17.18 0.11.0 (2023-08-07)

- Added a `ZYTE_API_MAX_REQUESTS` setting to limit the number of successful Zyte API requests that a spider can send. Reaching the limit stops the spider.
- Setting `requestCookies` to `[]` in the `zyte_api_automap` request metadata field now triggers a warning.

17.19 0.10.0 (2023-07-14)

- Added more data types to the scrapy-poet provider:
  - `zyte_common_items.ProductList`
  - `zyte_common_items.ProductNavigation`
  - `zyte_common_items.Article`
  - `zyte_common_items.ArticleList`
  - `zyte_common_items.ArticleNavigation`
- Moved the new dependencies added in 0.9.0 and needed only for the scrapy-poet provider (`scrapy-poet`, `web-poet`, `zyte-common-items`) into the new optional feature `[provider]`.
- Improved result caching in the scrapy-poet provider.
- Added a new setting, `ZYTE_API_USE_ENV_PROXY`, which can be set to `True` to access Zyte API using a proxy configured in the local environment.

- Fixed getting the Scrapy Cloud job ID.
- Improved the documentation.
- Improved the CI configuration.

17.20 0.9.0 (2023-06-13)

- New and updated requirements:
  - packaging >= 20.0
  - scrapy-poet >= 0.9.0
  - web-poet >= 0.13.0
  - zyte-common-items
- Added a scrapy-poet provider for Zyte API. Currently supported data types:
  - `web_poet.BrowserHtml`
  - `web_poet.BrowserResponse`
  - `zyte_common_items.Product`
- Added a `zyte_api_default_params` request meta key which allows users to ignore the `ZYTE_API_DEFAULT_PARAMS` setting for individual requests.
- CI fixes.

17.21 0.8.4 (2023-05-26)

- Fixed an exception raised by the downloader middleware when cookies were enabled.

17.22 0.8.3 (2023-05-17)

- Made Python 3.11 support official.
- Added support for the upcoming automatic extraction feature of Zyte API.
- Included a descriptive message in the exception that triggers when the download handler cannot be initialized.
- Clarified that `LOG_LEVEL` must be `DEBUG` for `ZYTE_API_LOG_REQUESTS` messages to be visible.

17.23 0.8.2 (2023-05-02)

- Fixed the handling of response cookies without a domain.
- CI fixes

17.24 0.8.1 (2023-04-13)

- Fixed an `AssertionError` when cookies are disabled.
- Added links to the README to improve navigation from GitHub.
- Added a license file (BSD-3-Clause).

17.25 0.8.0 (2023-03-28)

- Added experimental cookie support:
  - The `experimental.responseCookies` response parameter is now mapped to the response headers as `Set-Cookie` headers, as well as added to the cookiejar of the request.
  - A new boolean setting, `ZYTE_API_EXPERIMENTAL_COOKIES_ENABLED`, can be set to `True` to enable automatic mapping of cookies from a request cookiejar into the `experimental.requestCookies` Zyte API parameter.
- `ZyteAPITextResponse` is now a subclass of `HtmlResponse`, so that the `open_in_browser` function of Scrapy uses the `.html` extension for Zyte API responses.

  While not ideal, this is much better than the previous behavior, where the `.html` extension was *never* used for Zyte API responses.
- `ScrapyZyteAPIDownloaderMiddleware` now also supports non-string slot IDs.

17.26 0.7.1 (2023-01-25)

- It is now possible to log the parameters of requests sent.
- Stats for HTTP and HTTPS traffic used to be kept separate, and only one of those sets of stats would be reported. This is fixed now.
- Fixed some code examples and references in the README.

17.27 0.7.0 (2022-12-09)

When upgrading, you should set the following in your Scrapy settings:

```
DOWNLOADER_MIDDLEWARES = {
    "scrapy_zyte_api.ScrapyZyteAPIDownloaderMiddleware": 1000,
}
# only applicable for Scrapy 2.7+
REQUEST_FINGERPRINTER_CLASS = "scrapy_zyte_api.ScrapyZyteAPIRequestFingerprinter"
```

- Fixes the issue where scrapy-zyte-api is slow when Scrapy Cloud has Autothrottle Addon enabled. The new `ScrapyZyteAPIDownloaderMiddleware` fixes this.
- It now supports Scrapy 2.7's new `REQUEST_FINGERPRINTER_CLASS` which ensures that Zyte API requests are properly fingerprinted. This addresses the issue where Scrapy marks POST requests as duplicate if they point to the same URL despite having different request bodies. As a workaround, users were marking their requests with `dont_filter=True` to prevent such dupe filtering.

For users having `scrapy >= 2.7`, you can simply update your Scrapy settings to have `REQUEST_FINGERPRINTER_CLASS = "scrapy_zyte_api.ScrapyZyteAPIRequestFingerprinter"`.

If your Scrapy project performs other requests aside from Zyte API, you can set `ZYTE_API_FALLBACK_REQUEST_FINGERPRINTER_CLASS = "custom.RequestFingerprinter"` to allow custom fingerprinting. By default, the default Scrapy request fingerprinter is used for non-Zyte API requests.

For users having `scrapy < 2.7`, check the following link to see different ways on handling the duplicate request issue: https://github.com/scrapy-plugins/scrapy-zyte-api#request-fingerprinting-before-scrapy-27.

More information about the request fingerprinting topic can be found in https://github.com/scrapy-plugins/scrapy-zyte-api#request-fingerprinting.

- Various improvements to docs and tests.

17.28 0.6.0 (2022-10-20)

- Add a `ZYTE_API_TRANSPARENT_MODE` setting, `False` by default, which can be set to `True` to make all requests use Zyte API by default, with request parameters being automatically mapped to Zyte API parameters.
- Add a Request meta key, `zyte_api_automap`, that can be used to enable automatic request parameter mapping for specific requests, or to modify the outcome of automatic request parameter mapping for specific requests.
- Add a `ZYTE_API_AUTOMAP_PARAMS` setting, which is a counterpart for `ZYTE_API_DEFAULT_PARAMS` that applies to requests where automatic request parameter mapping is enabled.
- Add the `ZYTE_API_SKIP_HEADERS` and `ZYTE_API_BROWSER_HEADERS` settings to control the automatic mapping of request headers.
- Add a `ZYTE_API_ENABLED` setting, `True` by default, which can be used to disable this plugin.
- Document how Zyte API responses are mapped to Scrapy response subclasses.

17.29 0.5.1 (2022-09-20)

- Raise the minimum dependency of Zyte API's Python API to `zyte-api>=0.4.0`. This changes all the requests to Zyte API to have have `Accept-Encoding: br` and automatically decompress brotli responses.
- Rename "Zyte Data API" to simply "Zyte API" in the README.
- Lower the minimum Scrapy version from `2.6.0` to `2.0.1`.

17.30 0.5.0 (2022-08-25)

- Zyte Data API error responses (after retries) are no longer ignored, and instead raise a `zyte_api.aio.errors.RequestError` exception, which allows user-side handling of errors and provides better feedback for debugging.
- Allowed retry policies to be specified as import path strings, which is required for the `ZYTE_API_RETRY_POLICY` setting, and allows requests with the `zyte_api_retry_policy` request.meta key to remain serializable.
- Fixed the naming of stats for some error types.
- Updated the output examples on the README.

17.31 0.4.2 (2022-08-03)

- Cleaned up Scrapy stats names: fixed an issue with //, renamed `scrapy-zyte-api/api_error_types/..` to `scrapy-zyte-api/error_types/..`, added `scrapy-zyte-api/error_types/<empty>` for cases error type is unknown;
- Added error type to the error log messages
- Testing improvements

17.32 0.4.1 (2022-08-02)

Fixed incorrect 0.4.0 release.

17.33 0.4.0 (2022-08-02)

- Requires a more recent Python client library zyte-api 0.3.0.
- Stats from zyte-api are now copied into Scrapy stats. The `scrapy-zyte-api/request_count` stat has been renamed to `scrapy-zyte-api/processed` accordingly.

17.34 0.3.0 (2022-07-22)

- `CONCURRENT_REQUESTS` Scrapy setting is properly supported; in previous releases max concurrency of Zyte API requests was limited to 15.
- The retry policy for Zyte API requests can be overridden, using either `ZYTE_API_RETRY_POLICY` setting or `zyte_api_retry_policy` request.meta key.
- Proper response.status is set when Zyte API returns `statusCode` field.
- URL of the Zyte API server can be set using `ZYTE_API_URL` Scrapy setting. This feature is currently used in tests.
- The minimum required Scrapy version (2.6.0) is now enforced in setup.py.
- Test and documentation improvements.

17.35 0.2.0 (2022-05-31)

- Remove the `Content-Decoding` header when returning the responses. This prevents Scrapy from decompressing already decompressed contents done by Zyte Data API. Otherwise, this leads to errors inside Scrapy's `HttpCompressionMiddleware`.
- Introduce `ZyteAPIResponse` and `ZyteAPITextResponse` which are subclasses of `scrapy.http.Response` and `scrapy.http.TextResponse` respectively. These new response classes hold the raw Zyte Data API response in the `raw_api_response` attribute.
- Introduce a new setting named `ZYTE_API_DEFAULT_PARAMS`.
  - At the moment, this only applies to Zyte API enabled `scrapy.Request` (which is declared by having the `zyte_api` parameter in the Request meta having valid parameters, set to `True`, or `{}`).

- Specify in the **README** to set `dont_filter=True` when using the same URL but with different `zyte_api` parameters in the Request meta. This is a current workaround since Scrapy will tag them as duplicate requests and will result in duplication filtering.
- Various documentation improvements.

17.36 0.1.0 (2022-02-03)

- Initial release

INDEX