AlexaPy

Release 1.25.3

unknown

2021-11-23

CONTENTS:

CHAPTER

ONE

ALEXAPY

Python Package for controlling Alexa devices (echo dot, etc) programmatically. This was originally designed for alexa_media_player a custom_component for Home Assistant.

NOTE: Alexa has no official API; therefore, this library may stop working at any time without warning.

CHAPTER

TWO

CREDITS

Originally inspired by this blog (GitHub). Additional scaffolding from simplisafe-python

CHAPTER

THREE

CONTRIBUTING

1. Check for open features/bugs or initiate a discussion on one.
2. Fork the repository.
3. Install the dev environment: `make init`.
4. Enter the virtual environment: `pipenv shell`
5. Code your new feature or bug fix.
6. Write a test that covers your new functionality.
7. Update `README.md` with any new documentation.
8. Run tests and ensure 100% code coverage for your contribution: `make coverage`
9. Ensure you have no linting errors: `make lint`
10. Ensure you have typed your code correctly: `make typing`
11. Add yourself to `AUTHORS.md`.
12. Submit a pull request!

CHAPTER

FOUR

LICENSE

Apache-2.0. By providing a contribution, you agree the contribution is licensed under Apache-2.0.

CHAPTER

FIVE

API REFERENCE

See the docs .

CHAPTER

SIX

API REFERENCE

6.1 alexapy

6.1.1 Submodules

alexapy.aiohttp

Submodules

alexapy.aiohttp.abc

alexapy.aiohttp.base_protocol

alexapy.aiohttp.client

Functions

- `request()`: Constructs and sends a request. Returns response object.

Classes

- `ClientRequest`: Undocumented.
- `ClientResponse`: Undocumented.
- `Fingerprint`: Undocumented.
- `RequestInfo`: Undocumented.
- `BaseConnector`: Base connector class.
- `TCPConnector`: TCP connector.
- `UnixConnector`: Unix socket connector.
- `NamedPipeConnector`: Named pipe connector.
- `ClientWebSocketResponse`: Undocumented.
- `ClientSession`: First-class interface for making HTTP requests.
- `ClientTimeout`: Undocumented.

Exceptions

- `ClientConnectionError`: Base class for client socket errors.
- `ClientConnectorCertificateError`: Response certificate error.
- `ClientConnectorError`: Client connector error.
- `ClientConnectorSSLError`: Response ssl error.
- `ClientError`: Base class for client connection errors.
- `ClientHttpProxyError`: HTTP proxy error.
- `ClientOSError`: OSError error.
- `ClientPayloadError`: Response payload error.
- `ClientProxyConnectionError`: Proxy connection error.
- `ClientResponseError`: Connection error during reading response.
- `ClientSSLError`: Base error for ssl.*Errors.
- `ContentTypeError`: ContentType found is not valid.
- `InvalidURL`: Invalid URL.
- `ServerConnectionError`: Server connection errors.
- `ServerDisconnectedError`: Server disconnected.
- `ServerFingerprintMismatch`: SSL certificate does not match expected fingerprint.
- `ServerTimeoutError`: Server timeout error.
- `TooManyRedirects`: Client was redirected too many times.
- `WSServerHandshakeError`: websocket server handshake error.

`alexapy.aiohttp.client_exceptions`

Exceptions

- `ClientError`: Base class for client connection errors.
- `ClientConnectionError`: Base class for client socket errors.
- `ClientOSError`: OSError error.
- `ClientConnectorError`: Client connector error.
- `ClientProxyConnectionError`: Proxy connection error.
- `ClientSSLError`: Base error for ssl.*Errors.
- `ClientConnectorSSLError`: Response ssl error.
- `ClientConnectorCertificateError`: Response certificate error.
- `ServerConnectionError`: Server connection errors.
- `ServerTimeoutError`: Server timeout error.
- `ServerDisconnectedError`: Server disconnected.
- `ServerFingerprintMismatch`: SSL certificate does not match expected fingerprint.

- `ClientResponseError`: Connection error during reading response.
- `ClientHttpProxyError`: HTTP proxy error.
- `WSServerHandshakeError`: websocket server handshake error.
- `ContentTypeError`: ContentType found is not valid.
- `ClientPayloadError`: Response payload error.
- `InvalidURL`: Invalid URL.

alexapy.aiohttp.client_proto

alexapy.aiohttp.client_reqrep

Classes

- `ClientRequest`: Undocumented.
- `ClientResponse`: Undocumented.
- `RequestInfo`: Undocumented.
- `Fingerprint`: Undocumented.

alexapy.aiohttp.client_ws

alexapy.aiohttp.connector

Classes

- `BaseConnector`: Base connector class.
- `TCPConnector`: TCP connector.
- `UnixConnector`: Unix socket connector.
- `NamedPipeConnector`: Named pipe connector.

alexapy.aiohttp.cookiejar

Classes

- `CookieJar`: Implements cookie storage adhering to RFC 6265.
- `DummyCookieJar`: Implements a dummy cookie storage.

alexapy.aiohttp.formdata

Classes

- `FormData`: Helper class for multipart/form-data and

alexapy.aiohttp.frozenlist

alexapy.aiohttp.hdrs

HTTP Headers constants.

alexapy.aiohttp.helpers

Classes

- `BasicAuth`: Http basic authentication helper.
- `ChainMapProxy`: Abstract base class for generic types.

alexapy.aiohttp.http

Functions

- `ws_ext_gen()`: Undocumented.
- `ws_ext_parse()`: Undocumented.

Classes

- `StreamWriter`: Abstract stream writer.
- `HttpVersion`: HttpVersion(major, minor)
- `HeadersParser`: Undocumented.
- `HttpParser`: Helper class that provides a standard way to create an ABC using
- `HttpRequestParser`: Read request status line. Exception .http_exceptions.BadStatusLine
- `HttpResponseParser`: Read response status line and headers.
- `RawRequestMessage`: RawRequestMessage(method, path, version, headers, raw_headers, should_close, compression, upgrade, chunked, url)
- `RawResponseMessage`: RawResponseMessage(version, code, reason, headers, raw_headers, should_close, compression, upgrade, chunked)
- `WebSocketReader`: Undocumented.
- `WebSocketWriter`: Undocumented.
- `WSMessage`: _WSMessageBase(type, data, extra)
- `WSMsgType`: An enumeration.

- `WSCloseCode`: An enumeration.

Exceptions

- `HttpProcessingError`: HTTP error.
- `WebSocketError`: WebSocket protocol parser error.

Variables

- `RESPONSES`
- `SERVER_SOFTWARE`
- `HttpVersion10`
- `HttpVersion11`
- `WS_CLOSED_MESSAGE`
- `WS_CLOSING_MESSAGE`
- `WS_KEY`

alexapy.aiohttp.http_exceptions

Low-level http related exceptions.

Exceptions

- *HttpProcessingError*: HTTP error.

exception alexapy.aiohttp.http_exceptions.**HttpProcessingError**(*, *code: Optional[int] = None*, *message: str = ''*, *headers: Optional[multidict._multidict.CIMultiDict] = None*)

HTTP error.

Shortcut for raising HTTP errors with custom code, message and headers.

code: HTTP Error code. message: (optional) Error message. headers: (optional) Headers to be sent in response, a list of pairs

Inheritance

alexapy.aiohttp.http_parser

Classes

- `HeadersParser`: Undocumented.
- `HttpParser`: Helper class that provides a standard way to create an ABC using
- `HttpRequestParser`: Read request status line. Exception .http_exceptions.BadStatusLine
- `HttpResponseParser`: Read response status line and headers.
- `RawRequestMessage`: RawRequestMessage(method, path, version, headers, raw_headers, should_close, compression, upgrade, chunked, url)
- `RawResponseMessage`: RawResponseMessage(version, code, reason, headers, raw_headers, should_close, compression, upgrade, chunked)

alexapy.aiohttp.http_websocket

Classes

- `WebSocketReader`: Undocumented.
- `WebSocketWriter`: Undocumented.
- `WSMessage`: _WSMessageBase(type, data, extra)
- `WSMsgType`: An enumeration.
- `WSCloseCode`: An enumeration.

Exceptions

- `WebSocketError`: WebSocket protocol parser error.

Variables

- `WS_CLOSED_MESSAGE`
- `WS_CLOSING_MESSAGE`
- `WS_KEY`

alexapy.aiohttp.http_writer

Classes

- `StreamWriter`: Abstract stream writer.
- `HttpVersion`: HttpVersion(major, minor)

Variables

- `HttpVersion10`
- `HttpVersion11`

alexapy.aiohttp.locks

alexapy.aiohttp.log

alexapy.aiohttp.multipart

Functions

- `parse_content_disposition()`: Undocumented.
- `content_disposition_filename()`: Undocumented.

Classes

- `MultipartReader`: Multipart body reader.
- `MultipartWriter`: Multipart body writer.
- `BodyPartReader`: Multipart reader for single body part.

Exceptions

- `BadContentDispositionHeader`: Base class for warnings about dubious runtime behavior.
- `BadContentDispositionParam`: Base class for warnings about dubious runtime behavior.

alexapy.aiohttp.payload

Functions

- `get_payload()`: Undocumented.

Classes

- `payload_type`: Undocumented.
- `Payload`: Helper class that provides a standard way to create an ABC using
- `BytesPayload`: Helper class that provides a standard way to create an ABC using
- `StringPayload`: Helper class that provides a standard way to create an ABC using
- `IOBasePayload`: Helper class that provides a standard way to create an ABC using
- `BytesIOPayload`: Helper class that provides a standard way to create an ABC using
- `BufferedReaderPayload`: Helper class that provides a standard way to create an ABC using
- `TextIOPayload`: Helper class that provides a standard way to create an ABC using
- `StringIOPayload`: Helper class that provides a standard way to create an ABC using
- `JsonPayload`: Helper class that provides a standard way to create an ABC using
- `AsyncIterablePayload`: Helper class that provides a standard way to create an ABC using

Variables

- `PAYLOAD_REGISTRY`

alexapy.aiohttp.payload_streamer

Classes

- `streamer`: Undocumented.

alexapy.aiohttp.pytest_plugin

alexapy.aiohttp.resolver

Classes

- `ThreadedResolver`: Use Executor for synchronous getaddrinfo() calls, which defaults to
- `AsyncResolver`: Use the *aiodns* package to make asynchronous DNS lookups
- `DefaultResolver`: Use Executor for synchronous getaddrinfo() calls, which defaults to

alexapy.aiohttp.signals

Classes

- `Signal`: Coroutine-based signal implementation.

alexapy.aiohttp.streams

Classes

- `StreamReader`: An enhancement of asyncio.StreamReader.
- `DataQueue`: DataQueue is a general-purpose blocking queue with one reader.
- `FlowControlDataQueue`: FlowControlDataQueue resumes and pauses an underlying stream.

Exceptions

- `EofStream`: eof stream indication.

Variables

- `EMPTY_PAYLOAD`

alexapy.aiohttp.tcp_helpers

Helper methods to tune a TCP connection

Functions

- *tcp_keepalive()*: Undocumented.
- *tcp_nodelay()*: Undocumented.
- tcp_cork(): Undocumented.

alexapy.aiohttp.tcp_helpers.**tcp_keepalive**(*transport: asyncio.transports.Transport*) → None

alexapy.aiohttp.tcp_helpers.**tcp_nodelay**(*transport: asyncio.transports.Transport*, *value: bool*) → None

alexapy.aiohttp.test_utils

alexapy.aiohttp.tracing

Classes

- TraceConfig: First-class used to trace requests launched via ClientSession
- TraceRequestStartParams: Parameters sent by the *on_request_start* signal
- TraceRequestEndParams: Parameters sent by the *on_request_end* signal
- TraceRequestExceptionParams: Parameters sent by the *on_request_exception* signal
- TraceConnectionQueuedStartParams: Parameters sent by the *on_connection_queued_start* signal
- TraceConnectionQueuedEndParams: Parameters sent by the *on_connection_queued_end* signal
- TraceConnectionCreateStartParams: Parameters sent by the *on_connection_create_start* signal
- TraceConnectionCreateEndParams: Parameters sent by the *on_connection_create_end* signal
- TraceConnectionReuseconnParams: Parameters sent by the *on_connection_reuseconn* signal
- TraceDnsResolveHostStartParams: Parameters sent by the *on_dns_resolvehost_start* signal
- TraceDnsResolveHostEndParams: Parameters sent by the *on_dns_resolvehost_end* signal
- TraceDnsCacheHitParams: Parameters sent by the *on_dns_cache_hit* signal
- TraceDnsCacheMissParams: Parameters sent by the *on_dns_cache_miss* signal
- TraceRequestRedirectParams: Parameters sent by the *on_request_redirect* signal
- TraceRequestChunkSentParams: Parameters sent by the *on_request_chunk_sent* signal
- TraceResponseChunkReceivedParams: Parameters sent by the *on_response_chunk_received* signal

alexapy.aiohttp.typedefs

alexapy.aiohttp.web

Functions

- `middleware()`: Undocumented.
- `normalize_path_middleware()`: Middleware factory which produces a middleware that normalizes
- `json_response()`: Undocumented.
- `delete()`: Undocumented.
- `get()`: Undocumented.
- `head()`: Undocumented.
- `options()`: Undocumented.
- `patch()`: Undocumented.
- `post()`: Undocumented.
- `put()`: Undocumented.
- `route()`: Undocumented.
- `static()`: Undocumented.
- `view()`: Undocumented.
- `run_app()`: Run an app locally

Classes

- `Application`: Abstract base class for generic types.
- `FileResponse`: A response object can be used to send files.
- `RequestHandler`: HTTP protocol implementation.
- `BaseRequest`: Abstract base class for generic types.
- `FileField`: Undocumented.
- `Request`: Abstract base class for generic types.
- `ContentCoding`: An enumeration.
- `Response`: Undocumented.
- `StreamResponse`: Undocumented.
- `AbstractRouteDef`: Helper class that provides a standard way to create an ABC using
- `RouteDef`: Helper class that provides a standard way to create an ABC using
- `RouteTableDef`: Route definition table
- `StaticDef`: Helper class that provides a standard way to create an ABC using
- `AppRunner`: Web Application runner
- `BaseRunner`: Helper class that provides a standard way to create an ABC using

- `BaseSite`: Helper class that provides a standard way to create an ABC using
- `GracefulExit`: Request to exit from the interpreter.
- `ServerRunner`: Low-level web server runner
- `SockSite`: Helper class that provides a standard way to create an ABC using
- `TCPSite`: Helper class that provides a standard way to create an ABC using
- `UnixSite`: Helper class that provides a standard way to create an ABC using
- `NamedPipeSite`: Helper class that provides a standard way to create an ABC using
- `Server`: Undocumented.
- `AbstractResource`: Abstract base class for generic types.
- `AbstractRoute`: Helper class that provides a standard way to create an ABC using
- `DynamicResource`: Abstract base class for generic types.
- `PlainResource`: Abstract base class for generic types.
- `Resource`: Abstract base class for generic types.
- `ResourceRoute`: A route with resource
- `StaticResource`: Abstract base class for generic types.
- `UrlDispatcher`: Helper class that provides a standard way to create an ABC using
- `UrlMappingMatchInfo`: dict() -> new empty dictionary
- `View`: Abstract class based view.
- `WebSocketReady`: Undocumented.
- `WebSocketResponse`: Undocumented.
- `WSMsgType`: An enumeration.

Exceptions

- `CleanupError`: Unspecified run-time error.
- `HTTPAccepted`: Base class for exceptions with status codes in the 200s.
- `HTTPBadGateway`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPBadRequest`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPClientError`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPConflict`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPCreated`: Base class for exceptions with status codes in the 200s.
- `HTTPError`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPException`: Common base class for all non-exit exceptions.
- `HTTPExpectationFailed`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPFailedDependency`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPForbidden`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPFound`: Base class for exceptions with status codes in the 300s.

- `HTTPGatewayTimeout`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPGone`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPInsufficientStorage`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPInternalServerError`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPLengthRequired`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPMethodNotAllowed`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPMisdirectedRequest`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPMovedPermanently`: Base class for exceptions with status codes in the 300s.
- `HTTPMultipleChoices`: Base class for exceptions with status codes in the 300s.
- `HTTPNetworkAuthenticationRequired`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPNoContent`: Base class for exceptions with status codes in the 200s.
- `HTTPNonAuthoritativeInformation`: Base class for exceptions with status codes in the 200s.
- `HTTPNotAcceptable`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPNotExtended`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPNotFound`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPNotImplemented`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPNotModified`: Base class for exceptions with status codes in the 300s.
- `HTTPOk`: Base class for exceptions with status codes in the 200s.
- `HTTPPartialContent`: Base class for exceptions with status codes in the 200s.
- `HTTPPaymentRequired`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPPermanentRedirect`: Base class for exceptions with status codes in the 300s.
- `HTTPPreconditionFailed`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPPreconditionRequired`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPProxyAuthenticationRequired`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPRedirection`: Base class for exceptions with status codes in the 300s.
- `HTTPRequestEntityTooLarge`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPRequestHeaderFieldsTooLarge`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPRequestRangeNotSatisfiable`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPRequestTimeout`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPRequestURITooLong`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPResetContent`: Base class for exceptions with status codes in the 200s.
- `HTTPSeeOther`: Base class for exceptions with status codes in the 300s.
- `HTTPServerError`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPServiceUnavailable`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPSuccessful`: Base class for exceptions with status codes in the 200s.

- `HTTPTemporaryRedirect`: Base class for exceptions with status codes in the 300s.
- `HTTPTooManyRequests`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPUnauthorized`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPUnavailableForLegalReasons`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPUnprocessableEntity`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPUnsupportedMediaType`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPUpgradeRequired`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPUseProxy`: Base class for exceptions with status codes in the 300s.
- `HTTPVariantAlsoNegotiates`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPVersionNotSupported`: Base class for exceptions with status codes in the 400s and 500s.
- `PayloadAccessError`: Payload was accessed after response was sent.
- `RequestPayloadError`: Payload parsing error.

alexapy.aiohttp.web_app

Classes

- `Application`: Abstract base class for generic types.

Exceptions

- `CleanupError`: Unspecified run-time error.

alexapy.aiohttp.web_exceptions

Exceptions

- `HTTPException`: Common base class for all non-exit exceptions.
- `HTTPError`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPRedirection`: Base class for exceptions with status codes in the 300s.
- `HTTPSuccessful`: Base class for exceptions with status codes in the 200s.
- `HTTPOk`: Base class for exceptions with status codes in the 200s.
- `HTTPCreated`: Base class for exceptions with status codes in the 200s.
- `HTTPAccepted`: Base class for exceptions with status codes in the 200s.
- `HTTPNonAuthoritativeInformation`: Base class for exceptions with status codes in the 200s.
- `HTTPNoContent`: Base class for exceptions with status codes in the 200s.
- `HTTPResetContent`: Base class for exceptions with status codes in the 200s.
- `HTTPPartialContent`: Base class for exceptions with status codes in the 200s.
- `HTTPMultipleChoices`: Base class for exceptions with status codes in the 300s.

- `HTTPMovedPermanently`: Base class for exceptions with status codes in the 300s.
- `HTTPFound`: Base class for exceptions with status codes in the 300s.
- `HTTPSeeOther`: Base class for exceptions with status codes in the 300s.
- `HTTPNotModified`: Base class for exceptions with status codes in the 300s.
- `HTTPUseProxy`: Base class for exceptions with status codes in the 300s.
- `HTTPTemporaryRedirect`: Base class for exceptions with status codes in the 300s.
- `HTTPPermanentRedirect`: Base class for exceptions with status codes in the 300s.
- `HTTPClientError`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPBadRequest`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPUnauthorized`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPPaymentRequired`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPForbidden`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPNotFound`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPMethodNotAllowed`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPNotAcceptable`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPProxyAuthenticationRequired`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPRequestTimeout`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPConflict`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPGone`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPLengthRequired`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPPreconditionFailed`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPRequestEntityTooLarge`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPRequestURITooLong`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPUnsupportedMediaType`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPRequestRangeNotSatisfiable`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPExpectationFailed`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPMisdirectedRequest`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPUnprocessableEntity`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPFailedDependency`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPUpgradeRequired`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPPreconditionRequired`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPTooManyRequests`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPRequestHeaderFieldsTooLarge`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPUnavailableForLegalReasons`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPServerError`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPInternalServerError`: Base class for exceptions with status codes in the 400s and 500s.

- `HTTPNotImplemented`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPBadGateway`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPServiceUnavailable`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPGatewayTimeout`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPVersionNotSupported`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPVariantAlsoNegotiates`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPInsufficientStorage`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPNotExtended`: Base class for exceptions with status codes in the 400s and 500s.
- `HTTPNetworkAuthenticationRequired`: Base class for exceptions with status codes in the 400s and 500s.

alexapy.aiohttp.web_fileresponse

Classes

- `FileResponse`: A response object can be used to send files.

alexapy.aiohttp.web_log

alexapy.aiohttp.web_middlewares

Functions

- `middleware()`: Undocumented.
- `normalize_path_middleware()`: Middleware factory which produces a middleware that normalizes

alexapy.aiohttp.web_protocol

Classes

- `RequestHandler`: HTTP protocol implementation.

Exceptions

- `RequestPayloadError`: Payload parsing error.
- `PayloadAccessError`: Payload was accessed after response was sent.

alexapy.aiohttp.web_request

Classes

- `BaseRequest`: Abstract base class for generic types.
- `FileField`: Undocumented.
- `Request`: Abstract base class for generic types.

alexapy.aiohttp.web_response

Functions

- `json_response()`: Undocumented.

Classes

- `ContentCoding`: An enumeration.
- `StreamResponse`: Undocumented.
- `Response`: Undocumented.

alexapy.aiohttp.web_routedef

Functions

- `head()`: Undocumented.
- `options()`: Undocumented.
- `get()`: Undocumented.
- `post()`: Undocumented.
- `patch()`: Undocumented.
- `put()`: Undocumented.
- `delete()`: Undocumented.
- `route()`: Undocumented.
- `view()`: Undocumented.
- `static()`: Undocumented.

Classes

- `AbstractRouteDef`: Helper class that provides a standard way to create an ABC using
- `RouteDef`: Helper class that provides a standard way to create an ABC using
- `StaticDef`: Helper class that provides a standard way to create an ABC using
- `RouteTableDef`: Route definition table

alexapy.aiohttp.web_runner

Classes

- `BaseSite`: Helper class that provides a standard way to create an ABC using
- `TCPSite`: Helper class that provides a standard way to create an ABC using
- `UnixSite`: Helper class that provides a standard way to create an ABC using
- `NamedPipeSite`: Helper class that provides a standard way to create an ABC using
- `SockSite`: Helper class that provides a standard way to create an ABC using
- `BaseRunner`: Helper class that provides a standard way to create an ABC using
- `AppRunner`: Web Application runner
- `ServerRunner`: Low-level web server runner
- `GracefulExit`: Request to exit from the interpreter.

alexapy.aiohttp.web_server

Classes

- `Server`: Undocumented.

alexapy.aiohttp.web_urldispatcher

Classes

- `UrlDispatcher`: Helper class that provides a standard way to create an ABC using
- `UrlMappingMatchInfo`: dict() -> new empty dictionary
- `AbstractResource`: Abstract base class for generic types.
- `Resource`: Abstract base class for generic types.
- `PlainResource`: Abstract base class for generic types.
- `DynamicResource`: Abstract base class for generic types.
- `AbstractRoute`: Helper class that provides a standard way to create an ABC using
- `ResourceRoute`: A route with resource
- `StaticResource`: Abstract base class for generic types.

- `View`: Abstract class based view.

alexapy.aiohttp.web_ws

Classes

- `WebSocketResponse`: Undocumented.
- `WebSocketReady`: Undocumented.
- `WSMsgType`: An enumeration.

Functions

- `request()`: Constructs and sends a request. Returns response object.
- `content_disposition_filename()`: Undocumented.
- `parse_content_disposition()`: Undocumented.
- `get_payload()`: Undocumented.

Classes

- `BaseConnector`: Base connector class.
- `ClientResponse`: Undocumented.
- `ClientRequest`: Undocumented.
- `ClientSession`: First-class interface for making HTTP requests.
- `ClientTimeout`: Undocumented.
- `ClientWebSocketResponse`: Undocumented.
- `Fingerprint`: Undocumented.
- `RequestInfo`: Undocumented.
- `TCPConnector`: TCP connector.
- `UnixConnector`: Unix socket connector.
- `NamedPipeConnector`: Named pipe connector.
- `CookieJar`: Implements cookie storage adhering to RFC 6265.
- `DummyCookieJar`: Implements a dummy cookie storage.
- `FormData`: Helper class for multipart/form-data and
- `BasicAuth`: Http basic authentication helper.
- `ChainMapProxy`: Abstract base class for generic types.
- `HttpVersion`: HttpVersion(major, minor)
- `WSMsgType`: An enumeration.
- `WSCloseCode`: An enumeration.
- `WSMessage`: _WSMessageBase(type, data, extra)

- `BodyPartReader`: Multipart reader for single body part.
- `MultipartReader`: Multipart body reader.
- `MultipartWriter`: Multipart body writer.
- `AsyncIterablePayload`: Helper class that provides a standard way to create an ABC using
- `BufferedReaderPayload`: Helper class that provides a standard way to create an ABC using
- `BytesIOPayload`: Helper class that provides a standard way to create an ABC using
- `BytesPayload`: Helper class that provides a standard way to create an ABC using
- `IOBasePayload`: Helper class that provides a standard way to create an ABC using
- `JsonPayload`: Helper class that provides a standard way to create an ABC using
- `Payload`: Helper class that provides a standard way to create an ABC using
- `StringIOPayload`: Helper class that provides a standard way to create an ABC using
- `StringPayload`: Helper class that provides a standard way to create an ABC using
- `TextIOPayload`: Helper class that provides a standard way to create an ABC using
- `payload_type`: Undocumented.
- `streamer`: Undocumented.
- `AsyncResolver`: Use the *aiodns* package to make asynchronous DNS lookups
- `DefaultResolver`: Use Executor for synchronous getaddrinfo() calls, which defaults to
- `ThreadedResolver`: Use Executor for synchronous getaddrinfo() calls, which defaults to
- `Signal`: Coroutine-based signal implementation.
- `DataQueue`: DataQueue is a general-purpose blocking queue with one reader.
- `FlowControlDataQueue`: FlowControlDataQueue resumes and pauses an underlying stream.
- `StreamReader`: An enhancement of asyncio.StreamReader.
- `TraceConfig`: First-class used to trace requests launched via ClientSession
- `TraceConnectionCreateEndParams`: Parameters sent by the *on_connection_create_end* signal
- `TraceConnectionCreateStartParams`: Parameters sent by the *on_connection_create_start* signal
- `TraceConnectionQueuedEndParams`: Parameters sent by the *on_connection_queued_end* signal
- `TraceConnectionQueuedStartParams`: Parameters sent by the *on_connection_queued_start* signal
- `TraceConnectionReuseconnParams`: Parameters sent by the *on_connection_reuseconn* signal
- `TraceDnsCacheHitParams`: Parameters sent by the *on_dns_cache_hit* signal
- `TraceDnsCacheMissParams`: Parameters sent by the *on_dns_cache_miss* signal
- `TraceDnsResolveHostEndParams`: Parameters sent by the *on_dns_resolvehost_end* signal
- `TraceDnsResolveHostStartParams`: Parameters sent by the *on_dns_resolvehost_start* signal
- `TraceRequestChunkSentParams`: Parameters sent by the *on_request_chunk_sent* signal
- `TraceRequestEndParams`: Parameters sent by the *on_request_end* signal
- `TraceRequestExceptionParams`: Parameters sent by the *on_request_exception* signal
- `TraceRequestRedirectParams`: Parameters sent by the *on_request_redirect* signal

- `TraceRequestStartParams`: Parameters sent by the *on_request_start* signal
- `TraceResponseChunkReceivedParams`: Parameters sent by the *on_response_chunk_received* signal

Exceptions

- `ClientConnectionError`: Base class for client socket errors.
- `ClientConnectorCertificateError`: Response certificate error.
- `ClientConnectorError`: Client connector error.
- `ClientConnectorSSLError`: Response ssl error.
- `ClientError`: Base class for client connection errors.
- `ClientHttpProxyError`: HTTP proxy error.
- `ClientOSError`: OSError error.
- `ClientPayloadError`: Response payload error.
- `ClientProxyConnectionError`: Proxy connection error.
- `ClientResponseError`: Connection error during reading response.
- `ClientSSLError`: Base error for ssl.*Errors.
- `ContentTypeError`: ContentType found is not valid.
- `InvalidURL`: Invalid URL.
- `ServerConnectionError`: Server connection errors.
- `ServerDisconnectedError`: Server disconnected.
- `ServerFingerprintMismatch`: SSL certificate does not match expected fingerprint.
- `ServerTimeoutError`: Server timeout error.
- `TooManyRedirects`: Client was redirected too many times.
- `WSServerHandshakeError`: websocket server handshake error.
- `WebSocketError`: WebSocket protocol parser error.
- `BadContentDispositionHeader`: Base class for warnings about dubious runtime behavior.
- `BadContentDispositionParam`: Base class for warnings about dubious runtime behavior.
- `EofStream`: eof stream indication.

Variables

- *hdrs*
- `HttpVersion10`
- `HttpVersion11`
- `PAYLOAD_REGISTRY`
- `EMPTY_PAYLOAD`

alexapy.alexaapi

alexapy.alexalogin

alexapy.alexaproxy

alexapy.alexawebsocket

alexapy.const

alexapy.errors

alexapy.helpers

6.1.2 Functions

- `hide_email()`: Obfuscate email.
- `hide_serial()`: Obfuscate serial.
- `obfuscate()`: Obfuscate email, password, and other known sensitive keys.

6.1.3 Classes

- `AlexaLogin`: Class to handle login connection to Alexa. This class will not reconnect.
- `AlexaAPI`: Class for accessing a specific Alexa device using rest API.
- `AlexaProxy`: Class to handle proxy login connections to Alexa.
- `WebsocketEchoClient`: WebSocket Client Class for Echo Devices.

6.1.4 Exceptions

- `AlexapyConnectionError`: Define an error related to invalid requests.
- `AlexapyLoginCloseRequested`: Define an error related to requesting access to API after requested close.
- `AlexapyLoginError`: Define an error related to no longer being logged in.
- `AlexapyPyotpInvalidKey`: Define an error related to invalid 2FA key.

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

INDEX