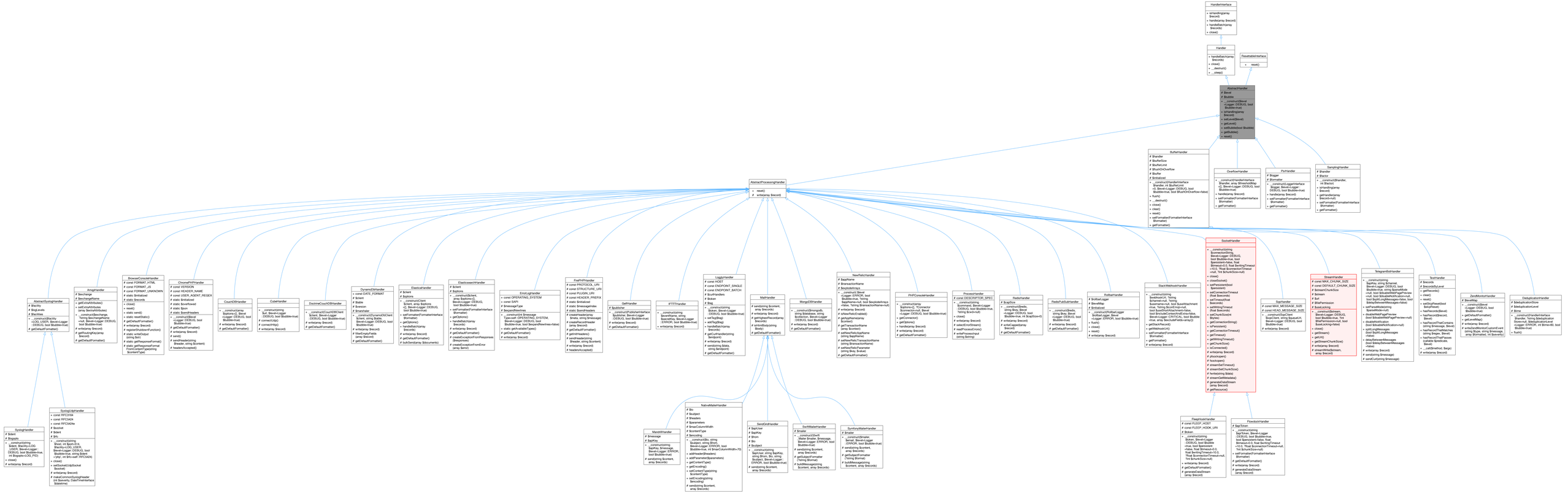
HandlerInterface
+ isHandling(array $record)
+ handle(array $record)
+ handleBatch(array $records)
+ close()
Handler
+ handleBatch(array $records)
+ close()
+ __destruct()
+ __sleep()
ResettableInterface
+ reset()
AbstractHandler
$level
$bubble
+ __construct($level =Logger::DEBUG, bool $bubble=true)
+ isHandling(array $record)
+ setLevel($level)
+ getLevel()
+ setBubble(bool $bubble)
+ getBubble()
+ reset()
AbstractProcessingHandler
+ reset()
# write(array $record)
BufferHandler
$handler
$bufferSize
$bufferLimit
$flushOnOverflow
$buffer
$initialized
+ __construct(HandlerInterface $handler, int $bufferLimit =0, $level=Logger::DEBUG, bool $bubble=true, bool $flushOnOverflow=false)
+ flush()
+ __destruct()
+ close()
+ clear()
+ reset()
+ setFormatter(FormatterInterface $formatter)
+ getFormatter()
OverflowHandler
+ __construct(HandlerInterface $handler, int|array $thresholds =0, $level=Logger::DEBUG, bool $bubble=true)
+ handle(array $record)
+ setFormatter(FormatterInterface $formatter)
+ getFormatter()
PsrHandler
$logger
$formatter
+ __construct(LoggerInterface $logger, $level=Logger::DEBUG, bool $bubble=true)
+ handle(array $record)
+ setFormatter(FormatterInterface $formatter)
+ getFormatter()
SamplingHandler
$handler
$factor
+ __construct(Handler, int $factor)
+ isHandling(array $record)
+ handle(array $record=null)
+ setFormatter(FormatterInterface $formatter)
+ getFormatter()
AbstractSyslogHandler
$facility
$logLevels
$facilities
+ __construct($facility =LOG_USER, $level=Logger::DEBUG, bool $bubble=true)
+ getDefaultFormatter()
AmqpHandler
$exchange
$exchangeName
+ getExtraAttributes()
+ setExtraAttributes(array $extraAttributes)
+ __construct(Exchange, ?string $exchangeName =null, $level=Logger::DEBUG, bool $bubble=true)
# writeArray($record)
# getFullqueryParam($record)
# getDefaultFormatter()
BrowserConsoleHandler
const FORMAT_HTML
const FORMAT_JS
const FORMAT_UNKNOWN
static $initialized
static $records
+ close()
+ reset()
+ static send()
+ static resetStatic()
# getDefaultFormatter()
# writeArray($record)
# registerShutdownFunction()
# static writeOutput($str)
# static getResponseFormat()
# static getResponseFormatFromContentTypes(array $contentTypes)
ChromePhpHandler
const VERSION
const HEADER_NAME
const USER_AGENT_REGEX
static $initialized
static $overflowed
static $json
static $sendHeaders
+ __construct($level =Logger::DEBUG, bool $bubble=true)
+ getDefaultFormatter()
+ writeArray($record)
+ send()
+ sendHeaders(array $header, string $content)
+ headersAccepted()
CouchDBHandler
+ __construct(array $options=[], $level =Logger::DEBUG, bool $bubble=true)
+ close()
# writeArray($record)
# getDefaultFormatter()
CubeHandler
+ __construct(string $url, $level=Logger::DEBUG, bool $bubble=true)
+ connectHttp()
+ close()
# writeArray($record)
DoctrineCouchDBHandler
+ __construct(CouchDBClient $client, $level=Logger::DEBUG, bool $bubble=true)
# writeArray($record)
# getDefaultFormatter()
DynamoDbHandler
+ const DATE_FORMAT
# $client
# $table
# $version
# $marshaler
+ __construct(DynamoDbClient $client, string $table, $level=Logger::DEBUG, bool $bubble=true)
# writeArray($record)
# formatEntry(array $record)
# getDefaultFormatter()
ElasticaHandler
$client
$options
+ __construct(Client $client, array $options=[], $level=Logger::DEBUG, bool $bubble=true)
+ setFormatter(FormatterInterface $formatter)
+ getOptions()
+ handleBatch(array $records)
# writeArray($record)
# getDefaultFormatter()
# bulkSend(array $documents)
ElasticsearchHandler
$client
$options
+ __construct(Client, array $options=[], $level=Logger::DEBUG, bool $bubble=true)
+ setFormatter(FormatterInterface $formatter)
+ getOptions()
+ handleBatch(array $records)
# writeArray($record)
# getDefaultFormatter()
# createException(FormatException, $response)
# createException(FormatException, $error, $errno)
ErrorLogHandler
+ const OPERATING_SYSTEM
+ const SAPI
# $messageType
# $sequenceNewlines
+ __construct(int $messageType=self::OPERATING_SYSTEM, $level=Logger::DEBUG, bool $bubble=true, bool $sequenceNewlines=false)
+ static getAvailableTypes()
# getDefaultFormatter()
# writeArray($record)
FirePHPHandler
const PROTOCOL_URI
const STRUCTURE_URI
const PLUGIN_URI
const HEADER_PREFIX
static $initialized
static $messageIndex
static $sendHeaders
createHeaders(array $data, string $messageIndex)
createRecordHeader(array $record)
getDefaultFormatter()
getInfoHeaders()
sendHeader($name, $value)
sendHeaders(array $headers, string $content)
writeArray($record)
headersAccepted()
GelfHandler
$publisher
+ __construct(PublisherInterface $publisher, $level=Logger::DEBUG, bool $bubble=true)
# writeArray($record)
# getDefaultFormatter()
IFTTTHandler
+ __construct(string $eventName, string $secretKey, $level=Logger::ERROR, bool $bubble=true)
+ writeArray($record)
LogglyHandler
const HOST
const ENDPOINT_SINGLE
const ENDPOINT_BATCH
$curlHandlers
$token
$tag
+ __construct(string $token, $level=Logger::DEBUG, bool $bubble=true)
+ setTag($tag)
+ addTag($tag)
+ handleBatch(array $records)
# getCurlHandler(string $endpoint)
# writeArray($record)
# sendToCurl(string $data, string $endpoint)
# getDefaultFormatter()
NewRelicHandler
$appName
$transactionName
$explodeArrays
+ __construct($level =Logger::DEBUG, bool $bubble=true, ?string $appName=null, bool $explodeArrays =false, ?string $transactionName=null)
# writeArray($record)
# shouldFetch(value)
# getKeyName(array $context)
# getTransactionName(array $context)
# setNewRelicAppName (string $appName)
# setNewRelicTransactionName (string $transactionName)
# setNewRelicParameter (string $key, $value)
# getDefaultFormatter()
PhpConsoleHandler
+ __construct(array $options=[], ?Connector $connector=null, $level =Logger::DEBUG, bool $bubble=true)
+ getConnector()
+ getOptions()
+ handle(array $record)
# writeArray($record)
# getDefaultFormatter()
ProcessHandler
const DESCRIPTOR_SPEC
+ __construct(string $command, $level=Logger::DEBUG, bool $bubble=true, ?string $cwd=null)
+ close()
# writeArray($record)
# selectErrorStream()
# readProcessErrors()
# writeProcessinput(string $string)
RedisHandler
$redis
$prefix
+ __construct(Redis, string $key, $level =Logger::DEBUG, bool $bubble=true, int $capSize=0)
# writeArray($record)
# writeCappedArray($record)
# getDefaultFormatter
RedisPubSubHandler
+ __construct(Redis, string $key, $level =Logger::DEBUG, bool $bubble=true, int $capSize=0)
# writeArray($record)
# getDefaultFormatter()
RollbarHandler
$rollbarLogger
$levelMap
$initialized
+ __construct(RollbarLogger $rollbarLogger, $level =Logger::ERROR, bool $bubble=true)
+ flush()
+ close()
+ reset()
# writeArray($record)
SlackWebhookHandler
+ __construct(string $webhookURL, ?string $channel=null, ?string $username=null, ?string $useAttachment =true, ?string $iconEmoji=null, bool $useShortAttachment=false, bool $includeContextAndExtra=false, $level=Logger::CRITICAL, bool $bubble =true, array $excludeFields=array())
+ getSlackRecord()
+ getWebhookURL()
+ setFormatter(FormatterInterface $formatter)
+ getFormatter()
# writeArray($record)
SocketHandler
+ __construct(string $connectionString, $level=Logger::DEBUG, bool $bubble=true, bool $persistent=false, float $timeout=0.0, float $writingTimeout =10.0, ?int $chunkSize=null, ?int $writeBuffer=null, ?int $buffersize=null)
+ close()
+ closeSocket()
+ setPersistent(bool $persistent)
+ setConnectionTimeout (float $seconds)
+ setTimeout(float $seconds)
+ setWriteTimeout (float $seconds)
+ setChunkSize($bytes)
+ getConnectionString()
+ isPersistent()
+ getConnectionTimeout()
+ getTimeout()
+ getWriteTimeout()
+ getChunkSize()
+ isConnected()
# writeArray($record)
# pfsockopen()
# fsockopen()
# streamSetTimeout()
# streamSetChunkSize()
# fwrite(string $data)
# streamGetMetadata()
# generateDataStream (array $record)
# getResource()
SqsHandler
const MAX_MESSAGE_SIZE
const HEAD_MESSAGE_SIZE
+ __construct(SqsClient $sqsClient, string $queueUrl, $level=Logger::DEBUG, bool $bubble=true)
# writeArray($record)
StreamHandler
const MAX_CHUNK_SIZE
const DEFAULT_CHUNK_SIZE
$streamChunkSize
$stream
$url
$filePermission
$useLocking
+ __construct($stream, $level=Logger::DEBUG, bool $bubble=true, ?int $filePermission=null, bool $useLocking=false)
+ close()
+ getStream()
+ getUrl()
+ getStreamChunkSize()
# writeArray($record)
# streamWrite($stream, array $record)
TelegramBotHandler
+ __construct(string $apiKey, string $channel, $level=Logger::DEBUG, bool $bubble=true, string $parseMode =null, bool $disableWebPagePreview =null, bool $disableNotification=null, bool $splitLongMessages=false, bool $delayBetweenMessages=false)
+ setParseMode(string $parseMode=null)
+ disableWebPagePreview (bool $disableWebPagePreview=null)
+ disableNotification (bool $disableNotification=null)
+ splitLongMessages (bool $splitLongMessages =false)
+ delayBetweenMessages (bool $delayBetweenMessages =false)
# writeArray($record)
# send(string $message)
# sendContent(string $message)
TestHandler
$records
$recordsByLevel
$error
+ reset()
+ serialize()
+ hasRecords(int|string $level)
+ hasRecord($record, $level)
+ hasRecordThat(callable $predicate, $level)
+ hasRecordThatMatches (string $regex, $level)
+ hasRecordThatContains (string $message, $level)
+ hasRecordThatPasses (callable $predicate, $level)
# writeArray($record)
ZendMonitorHandler
$sevMap
+ __construct($level =Logger::DEBUG, bool $bubble=true)
# getDefaultFormatter()
+ getLevelMap()
# writeArray($record)
# writeZendMonitorCustomEvent (string $type, string $message, array $formatted, int $severity)
DeduplicationHandler
$deduplicationStore
$deduplicationLevel
$time
+ __construct(HandlerInterface $handler, ?string $deduplication Store=null, $deduplicationLevel =Logger::ERROR, int $time=60, bool $bubble=true)
+ flush()
NativeMailerHandler
$to
$subject
$headers
$parameters
$maxColumnWidth
$contentType
$encoding
+ __construct($to, string $subject, string $from, $level=Logger::ERROR, bool $bubble=true, int $maxColumnWidth=70)
+ addHeader($headers)
+ addParameter($parameters)
+ getContentType()
+ getEncoding()
+ setContentTypeUsing($contentType)
+ setEncodingUsing($encoding)
# sendUsing($content, array $records)
MandrillHandler
$message
$apiKey
+ __construct(string $apiKey, $message, $level=Logger::ERROR, bool $bubble=true)
# sendUsing($content, array $records)
MailHandler
sendUsing($content, array $records)
writeArray($record)
getHeaderFrom(array $records)
setMimeBody(string $content)
getDefaultFormatter()
MongoDBHandler
+ __construct($mongodb, string $collection, $level=Logger::DEBUG, bool $bubble=true)
# writeArray($record)
# getDefaultFormatter()
SendGridHandler
$apiUser
$apiKey
$from
$to
$subject
+ __construct(string $apiUser, string $apiKey, string $from, $to, string $subject, $level=Logger::ERROR, bool $bubble=true)
# sendUsing($content, array $records)
SwiftMailerHandler
$mailer
+ __construct(Swift_Mailer $mailer, $message, $level=Logger::ERROR, bool $bubble=true)
# sendUsing($content, array $records)
# getSubjectFormatter (?string $format)
# buildMessage(string $content, array $records)
SymfonyMailerHandler
$mailer
+ __construct(Mailer $mailer, $message, $level=Logger::ERROR, bool $bubble=true)
# sendUsing($content, array $records)
# getSubjectFormatter (?string $format)
# buildMessage(string $content, array $records)
FlowdockHandler
$apiTokens
+ __construct(string $apiToken, $level=Logger::DEBUG, bool $bubble=true, bool $overSSL=false, float $timeout=0.0, float $writingTimeout =10.0, ?bool $connectionTimeout=null, ?int $chunkSize=null)
+ setFormatter(FormatterInterface $formatter)
+ getFormatter()
# getDefaultFormatter()
# writeArray($record)
# getSentCatsStream (array $record)
FlapjackHandler
const FLEEP_HOST
const FLEEP_HOOK_URI
$token
+ __construct(string $token, $level=Logger::DEBUG, bool $bubble =true, bool $ignoreerror =true, float $timeout=0.0, float $writingTimeout=10.0, ?float $connectionTimeout=null, ?int $chunkSize=null)
# writeArray($record)
# getDefaultFormatter()
# generateDataStream (array $record)