Plaso Documentation

Release 20180312

The Plaso Project Authors

Mar 18, 2018

Contents

Plaso (Plaso Langar Að Safna Öllu) is a computer forensic tool for timeline generation and analysis.

The project's code is available from https://github.com/log2timeline/plaso, and user documentation is available at https://github.com/log2timeline/plaso/wiki/ and http://plaso.kiddaland.com.

Plaso is licensed under the Apache license version 2.

Project Contents:

CHAPTER 1

plaso

1.1 plaso package

1.1.1 Subpackages

plaso.analysis package

Submodules

plaso.analysis.browser_search module

A plugin that extracts browser history from events.

class plaso.analysis.browser_search.**BrowserSearchPlugin**

Bases: *plaso.analysis.interface.AnalysisPlugin*

Analyze browser search entries from events.

CompileReport (*mediator*)

Compiles an analysis report.

Parameters **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.

Returns analysis report.

Return type *AnalysisReport*

ENABLE_IN_EXTRACTION = False

ExamineEvent (*mediator*, *event*)

Analyzes an event.

Parameters

- **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.
- **event** (EventObject) – event to examine.

NAME = u'browser_search'

class plaso.analysis.browser_search.**SEARCH_OBJECT**(*time*, *source*, *engine*, *search_term*)
Bases: tuple

__getnewargs__()
Return self as a plain tuple. Used by copy and pickle.

__getstate__()
Exclude the OrderedDict from pickling

__repr__()
Return a nicely formatted representation string

engine
Alias for field number 2

search_term
Alias for field number 3

source
Alias for field number 1

time
Alias for field number 0

plaso.analysis.chrome_extension module

A plugin that gather extension IDs from Chrome history browser.

class plaso.analysis.chrome_extension.**ChromeExtensionPlugin**
Bases: *plaso.analysis.interface.AnalysisPlugin*

Convert Chrome extension IDs into names, requires Internet connection.

CompileReport(*mediator*)
Compiles an analysis report.

Parameters **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.

Returns analysis report.

Return type *AnalysisReport*

ENABLE_IN_EXTRACTION = True

ExamineEvent(*mediator*, *event*)
Analyzes an event.

Parameters

- **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.
- **event** (EventObject) – event to examine.

NAME = u'chrome_extension'

plaso.analysis.definitions module

This file contains the definitions for analysis plugins.

plaso.analysis.file_hashes module

A plugin to generate a list of unique hashes and paths.

class plaso.analysis.file_hashes.**FileHashesPlugin**
Bases: *plaso.analysis.interface.AnalysisPlugin*

A plugin for generating a list of file paths and corresponding hashes.

CompileReport (*mediator*)
Compiles an analysis report.

Parameters **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.

Returns report.

Return type *AnalysisReport*

ENABLE_IN_EXTRACTION = True

ExamineEvent (*mediator*, *event*)
Analyzes an event and creates extracts hashes as required.

Parameters

- **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.
- **event** (EventObject) – event to examine.

NAME = u'file_hashes'

plaso.analysis.interface module

This file contains the interface for analysis plugins.

class plaso.analysis.interface.**AnalysisPlugin**
Bases: object

Class that defines the analysis plugin interface.

CompileReport (*mediator*)
Compiles a report of the analysis.

After the plugin has received every copy of an event to analyze this function will be called so that the report can be assembled.

Parameters **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.

Returns report.

Return type *AnalysisReport*

ENABLE_IN_EXTRACTION = False

ExamineEvent(*mediator*, *event*)
Analyzes an event object.

Parameters

- **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.
- **event** (EventObject) – event.

NAME = u'analysis_plugin'

URLS = []

plugin_name
str – name of the plugin.

class plaso.analysis.interface.**HTTPHashAnalyzer**(*hash_queue*, *hash_analysis_queue*, ***kwargs*)
Bases: *plaso.analysis.interface.HashAnalyzer*

Interface for hash analysis plugins that use HTTP(S)

Analyze(*hashes*)
Analyzes a list of hashes.

Parameters **hashes** (*list[str]*) – hashes to look up.

Returns analysis results.

Return type list[*HashAnalysis*]

MakeRequestAndDecodeJSON(*url*, *method*, ***kwargs*)
Make a HTTP request and decode the results as JSON.

Parameters

- **url** (*str*) – URL to make a request to.
- **method** (*str*) – HTTP method to used to make the request. GET and POST are supported.
- **kwargs** – parameters to the requests .get() or post() methods, depending on the value of the method parameter.

Returns body of the HTTP response, decoded from JSON.

Return type dict[str, object]

Raises

- ConnectionError – If it is not possible to connect to the given URL, or it the request returns a HTTP error.
- ValueError – If an invalid HTTP method is specified.

class plaso.analysis.interface.**HashAnalysis**(*subject_hash*, *hash_information*)
Bases: object

Analysis information about a hash.

hash_information
object – object containing information about the hash.

subject_hash
str – hash that was analyzed.

class plaso.analysis.interface.**HashAnalyzer**(*hash_queue*, *hash_analysis_queue*, *hashes_per_batch=1*, *lookup_hash=u'sha256'*, *wait_after_analysis=0*)

Bases: threading.Thread

Class that defines the interfaces for hash analyzer threads.

This interface should be implemented once for each hash analysis plugin.

analyses_performed
int – number of analysis batches completed by this analyzer.

hashes_per_batch
int – maximum number of hashes to analyze at once.

lookup_hash
str – name of the hash attribute to look up.

seconds_spent_analyzing
int – number of seconds this analyzer has spent performing analysis (as opposed to waiting on queues, etc.)

wait_after_analysis
int – number of seconds the analyzer will sleep for after analyzing a batch of hashes.

Analyze(*hashes*)
Analyzes a list of hashes.

Parameters **hashes** (*list[str]*) – list of hashes to look up.

Returns list of results of analyzing the hashes.

Return type list[*HashAnalysis*]

EMPTY_QUEUE_WAIT_TIME = 4

SUPPORTED_HASHES = []

SetLookupHash(*lookup_hash*)
Sets the hash to query.

Parameters **lookup_hash** (*str*) – name of the hash attribute to look up.

Raises ValueError – if the lookup hash is not supported.

SignalAbort()
Instructs this analyzer to stop running.

run()
The method called by the threading library to start the thread.

class plaso.analysis.interface.**HashTaggingAnalysisPlugin**(*analyzer_class*)

Bases: *plaso.analysis.interface.AnalysisPlugin*

An interface for plugins that tag events based on the source file hash.

An implementation of this class should be paired with an implementation of the HashAnalyzer interface.

hash_analysis_queue
Queue.queue – queue that contains the results of analysis of file hashes.

hash_queue
Queue.queue – queue that contains file hashes.

CompileReport(*mediator*)
Compiles an analysis report.

Parameters **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.

Returns report.

Return type *AnalysisReport*

DATA_TYPES = []

DEFAULT_QUEUE_TIMEOUT = 4

EstimateTimeRemaining()
Estimates how long until all hashes have been analyzed.

Returns estimated number of seconds until all hashes have been analyzed.

Return type int

ExamineEvent(*mediator*, *event*)
Evaluates whether an event contains the right data for a hash lookup.

Parameters

- **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.
- **event** (EventObject) – event.

GenerateLabels(*hash_information*)
Generates a list of strings to tag events with.

Parameters **hash_information** (*object*) – object that mediates the result of the analysis of a hash, as returned by the Analyze() method of the analyzer class associated with this plugin.

Returns list of labels to apply to events.

Return type list[str]

SECONDS_BETWEEN_STATUS_LOG_MESSAGES = 30

SetLookupHash(*lookup_hash*)
Sets the hash to query.

Parameters **lookup_hash** (*str*) – name of the hash attribute to look up.

plaso.analysis.manager module

This file contains the analysis plugin manager class.

class plaso.analysis.manager.**AnalysisPluginManager**
Bases: object

Analysis plugin manager.

classmethod DeregisterPlugin(*plugin_class*)
Deregisters an analysis plugin class.

The analysis plugin classes are identified by their lower case name.

Parameters **plugin_class** (*type*) – class of the analysis plugin.

Raises KeyError – if an analysis plugin class is not set for the corresponding name.

classmethod GetAllPluginInformation(*show_all=True*)
Retrieves a list of the registered analysis plugins.

Parameters **show_all** (*Optional[bool]*) – True if all analysis plugin names should be listed.

Returns

the name, docstring and type string of each analysis plugin in alphabetical order.

Return type list[tuple[str, str, str]]

classmethod GetPluginNames()
Retrieves the analysis plugin names.

Returns analysis plugin names.

Return type list[str]

classmethod GetPluginObjects(*plugin_names*)
Retrieves the plugin objects.

Parameters **plugin_names** (*list[str]*) – names of plugins that should be retrieved.

Returns analysis plugins per name.

Return type dict[str, *AnalysisPlugin*]

classmethod GetPlugins()
Retrieves the registered analysis plugin classes.

Yields *tuple* –

contains:

str: name of the plugin type: plugin class

classmethod RegisterPlugin(*plugin_class*)
Registers an analysis plugin class.

Then analysis plugin classes are identified based on their lower case name.

Parameters **plugin_class** (*type*) – class of the analysis plugin.

Raises KeyError – if an analysis plugin class is already set for the corresponding name.

classmethod RegisterPlugins(*plugin_classes*)
Registers analysis plugin classes.

The analysis plugin classes are identified based on their lower case name.

Parameters **plugin_classes** (*list[type]*) – classes of the analysis plugin.

Raises KeyError – if an analysis plugin class is already set for the corresponding name.

plaso.analysis.mediator module

The analysis plugin mediator object.

class plaso.analysis.mediator.**AnalysisMediator**(*storage_writer*, *knowledge_base*, *data_location=None*)

Bases: object

Analysis plugin mediator.

last_activity_timestamp
int – timestamp received that indicates the last time activity was observed. The last activity timestamp is updated when the mediator produces an attribute container, such as an event tag. This timestamp is used by the multi processing worker process to indicate the last time the worker was known to be active. This information is then used by the foreman to detect workers that are not responding (stalled).

number_of_produced_analysis_reports
int – number of produced analysis reports.

number_of_produced_event_tags
int – number of produced event tags.

GetDisplayNameForPathSpec(*path_spec*)
Retrieves the display name for a path specification.

Parameters **path_spec** (*dfvfs.PathSpec*) – path specification.

Returns human readable version of the path specification.

Return type str

GetUsernameForPath(*path*)
Retrieves a username for a specific path.

This is determining if a specific path is within a user's directory and returning the username of the user if so.

Parameters **path** (*str*) – path.

Returns

username or None if the path does not appear to be within a user's directory.

Return type str

ProduceAnalysisReport(*plugin*)
Produces an analysis report.

Parameters **plugin** (AnalysisPlugin) – plugin.

ProduceEventTag(*event_tag*)
Produces an event tag.

Parameters **event_tag** (EventTag) – event tag.

SignalAbort()
Signals the analysis plugins to abort.

abort
bool – True if the analysis should be aborted.

data_location
str – path to the data files.

operating_system
str – operating system or None if not set.

plaso.analysis.nsrlsvr module

Analysis plugin to look up files in nsrlsvr and tag events.

class plaso.analysis.nsrlsvr.**NsrlsvrAnalysisPlugin**
Bases: *plaso.analysis.interface.HashTaggingAnalysisPlugin*

Analysis plugin for looking up hashes in nsrlsvr.

DATA_TYPES = [u'fs:stat', u'fs:stat:ntfs']

GenerateLabels(*hash_information*)
Generates a list of strings that will be used in the event tag.

Parameters **hash_information** (*bool*) – whether the analyzer received a response from nsrlsvr indicating that the hash was present in its loaded NSRL set.

Returns strings describing the results from nsrlsvr.

Return type list[str]

NAME = u'nsrlsvr'

SetHost(*host*)
Sets the address or hostname of the server running nsrlsvr.

Parameters **host** (*str*) – IP address or hostname to query.

SetLabel(*label*)
Sets the tagging label.

Parameters **label** (*str*) – label to apply to events extracted from files that are present in nsrlsvr.

SetPort(*port*)
Sets the port where nsrlsvr is listening.

Parameters **port** (*int*) – port to query.

TestConnection()
Tests the connection to nsrlsvr.

Returns True if nsrlsvr instance is reachable.

Return type bool

URLS = [u'https://rjhansen.github.io/nsrlsvr/']

class plaso.analysis.nsrlsvr.**NsrlsvrAnalyzer**(*hash_queue*, *hash_analysis_queue*, ***kwargs*)
Bases: *plaso.analysis.interface.HashAnalyzer*

Analyzes file hashes by consulting an nsrlsvr instance.

analyses_performed
int – number of analysis batches completed by this analyzer.

hashes_per_batch
int – maximum number of hashes to analyze at once.

seconds_spent_analyzing
int – number of seconds this analyzer has spent performing analysis (as opposed to waiting on queues, etc.)

wait_after_analysis
int – number of seconds the analyzer will sleep for after analyzing a batch of hashes.

Analyze(*hashes*)
Looks up hashes in nsrlsvr.

Parameters **hashes** (*list[str]*) – hash values to look up.

Returns analysis results, or an empty list on error.

Return type list[*HashAnalysis*]

SUPPORTED_HASHES = [u'md5', u'sha1']

SetHost(*host*)
Sets the address or hostname of the server running nsrlsvr.

Parameters **host** (*str*) – IP address or hostname to query.

SetPort(*port*)
Sets the port where nsrlsvr is listening.

Parameters **port** (*int*) – port to query.

TestConnection()
Tests the connection to nsrlsvr.

Checks if a connection can be set up and queries the server for the MD5 of an empty file and expects a response. The value of the response is not checked.

Returns True if nsrlsvr instance is reachable.

Return type bool

plaso.analysis.sessionize module

A plugin to tag events according to rules in a tag file.

class plaso.analysis.sessionize.**SessionizeAnalysisPlugin**
Bases: *plaso.analysis.interface.AnalysisPlugin*

Analysis plugin that labels events by session.

CompileReport(*mediator*)
Compiles an analysis report.

Parameters **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.

Returns analysis report.

Return type *AnalysisReport*

ENABLE_IN_EXTRACTION = False

ExamineEvent(*mediator*, *event*)
Analyzes an EventObject and tags it as part of a session.

Parameters

- **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.
- **event** (EventObject) – event to examine.

NAME = u'sessionize'

SetMaximumPause(*maximum_pause_minutes*)
Sets the maximum pause interval between events to consider a session.

Parameters **maximum_pause_minutes** (*int*) – maximum gap between events that are part of the same session, in minutes.

plaso.analysis.tagging module

A plugin to tag events according to rules in a tagging file.

class plaso.analysis.tagging.**TaggingAnalysisPlugin**
Bases: *plaso.analysis.interface.AnalysisPlugin*

Analysis plugin that tags events according to rules in a tag file.

CompileReport(*mediator*)
Compiles an analysis report.

Parameters **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.

Returns analysis report.

Return type *AnalysisReport*

ENABLE_IN_EXTRACTION = True

ExamineEvent(*mediator*, *event*)
Analyzes an EventObject and tags it according to rules in the tag file.

Parameters

- **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.
- **event** (EventObject) – event to examine.

NAME = u'tagging'

SetAndLoadTagFile(*tagging_file_path*)
Sets the tag file to be used by the plugin.

Parameters **tagging_file_path** (*str*) – path of the tagging file.

plaso.analysis.unique_domains_visited module

A plugin to generate a list of domains visited.

class plaso.analysis.unique_domains_visited.**UniqueDomainsVisitedPlugin**
Bases: *plaso.analysis.interface.AnalysisPlugin*

A plugin to generate a list all domains visited.

This plugin will extract domains from browser history events extracted by Plaso. The list produced can be used to quickly determine if there has been a visit to a site of interest, for example, a known phishing site.

CompileReport(*mediator*)
Compiles an analysis report.

Parameters **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.

Returns The analysis report (instance of AnalysisReport).

ENABLE_IN_EXTRACTION = True

ExamineEvent(*mediator*, *event*)
Analyzes an event and extracts domains from it.

We only evaluate straightforward web history events, not visits which can be inferred by TypedURLs, cookies or other means.

Parameters

- **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.
- **event** (EventObject) – event to examine.

NAME = u'unique_domains_visited'

plaso.analysis.viper module

Analysis plugin to look up files in Viper and tag events.

class plaso.analysis.viper.**ViperAnalysisPlugin**

Bases: *plaso.analysis.interface.HashTaggingAnalysisPlugin*

An analysis plugin for looking up SHA256 hashes in Viper.

DATA_TYPES = [u'pe:compilation:compilation_time']

GenerateLabels(*hash_information*)

Generates a list of strings that will be used in the event tag.

Parameters **hash_information** (*dict[str, object]*) – JSON decoded contents of the result of a Viper lookup, as produced by the ViperAnalyzer.

Returns list of labels to apply to events.

Return type list[str]

NAME = u'viper'

SetHost(*host*)

Sets the address or hostname of the server running Viper server.

Parameters **host** (*str*) – IP address or hostname to query.

SetPort(*port*)

Sets the port where Viper server is listening.

Parameters **port** (*int*) – port to query.

SetProtocol(*protocol*)

Sets the protocol that will be used to query Viper.

Parameters **protocol** (*str*) – protocol to use to query Viper. Either 'http' or 'https'.

Raises ValueError – If an invalid protocol is selected.

TestConnection()

Tests the connection to the Viper server.

Returns True if the Viper server instance is reachable.

Return type bool

URLS = [u'https://viper.li']

class plaso.analysis.viper.**ViperAnalyzer**(*hash_queue*, *hash_analysis_queue*, ***kwargs*)

Bases: *plaso.analysis.interface.HTTPHashAnalyzer*

Class that analyzes file hashes by consulting Viper.

REST API reference: https://viper-framework.readthedocs.org/en/latest/usage/web.html#api

Analyze(*hashes*)
Looks up hashes in Viper using the Viper HTTP API.

Parameters **hashes** (*list[str]*) – hashes to look up.

Returns hash analysis.

Return type list[*HashAnalysis*]

Raises RuntimeError – If no host has been set for Viper.

SUPPORTED_HASHES = [u'md5', u'sha256']

SUPPORTED_PROTOCOLS = [u'http', u'https']

SetHost(*host*)
Sets the address or hostname of the server running Viper server.

Parameters **host** (*str*) – IP address or hostname to query.

SetPort(*port*)
Sets the port where Viper server is listening.

Parameters **port** (*int*) – port to query.

SetProtocol(*protocol*)
Sets the protocol that will be used to query Viper.

Parameters **protocol** (*str*) – protocol to use to query Viper. Either 'http' or 'https'.

Raises ValueError – if the protocol is not supported.

TestConnection()
Tests the connection to the Viper server.

Returns True if the Viper server instance is reachable.

Return type bool

plaso.analysis.virustotal module

Analysis plugin to look up files in VirusTotal and tag events.

class plaso.analysis.virustotal.**VirusTotalAnalysisPlugin**
Bases: *plaso.analysis.interface.HashTaggingAnalysisPlugin*

An analysis plugin for looking up hashes in VirusTotal.

DATA_TYPES = [u'pe:compilation:compilation_time']

EnableFreeAPIKeyRateLimit()
Configures Rate limiting for queries to VirusTotal.

The default rate limit for free VirusTotal API keys is 4 requests per minute.

GenerateLabels(*hash_information*)
Generates a list of strings that will be used in the event tag.

Parameters **hash_information** (*dict[str, object]*) – the JSON decoded contents of the result of a VirusTotal lookup, as produced by the VirusTotalAnalyzer.

Returns strings describing the results from VirusTotal.

Return type list[str]

NAME = u'virustotal'

SetAPIKey(*api_key*)
Sets the VirusTotal API key to use in queries.

Parameters **api_key** (*str*) – VirusTotal API key

TestConnection()
Tests the connection to VirusTotal

Returns True if VirusTotal is reachable.

Return type bool

URLS = [u'https://virustotal.com']

class plaso.analysis.virustotal.**VirusTotalAnalyzer**(*hash_queue*, *hash_analysis_queue*, ***kwargs*)
Bases: *plaso.analysis.interface.HTTPHashAnalyzer*

Class that analyzes file hashes by consulting VirusTotal.

Analyze(*hashes*)
Looks up hashes in VirusTotal using the VirusTotal HTTP API.

The API is documented here: https://www.virustotal.com/en/documentation/public-api/

Parameters **hashes** (*list[str]*) – hashes to look up.

Returns analysis results.

Return type list[*HashAnalysis*]

Raises RuntimeError – If the VirusTotal API key has not been set.

SUPPORTED_HASHES = [u'md5', u'sha1', u'sha256']

SetAPIKey(*api_key*)
Sets the VirusTotal API key to use in queries.

Parameters **api_key** (*str*) – VirusTotal API key

TestConnection()
Tests the connection to VirusTotal

Returns True if VirusTotal is reachable.

Return type bool

plaso.analysis.windows_services module

A plugin to enable quick triage of Windows Services.

class plaso.analysis.windows_services.**WindowsServiceCollection**
Bases: object

Class to hold and de-duplicate Windows Services.

AddService(*new_service*)
Add a new service to the list of ones we know about.

Parameters **new_service** (*WindowsService*) – the service to add.

services
list[WindowsService] – services in this collection.

class plaso.analysis.windows_services.**WindowsServicesAnalysisPlugin**
Bases: *plaso.analysis.interface.AnalysisPlugin*

Provides a single list of for Windows services found in the Registry.

CompileReport(*mediator*)
Compiles an analysis report.

Parameters **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.

Returns report.

Return type *AnalysisReport*

ENABLE_IN_EXTRACTION = True

ExamineEvent(*mediator*, *event*)
Analyzes an event and creates Windows Services as required.

At present, this method only handles events extracted from the Registry.

Parameters

- **mediator** (AnalysisMediator) – mediates interactions between analysis plugins and other components, such as storage and dfvfs.
- **event** (EventObject) – event to examine.

NAME = u'windows_services'

SetOutputFormat(*output_format*)
Sets the output format of the generated report.

Parameters **output_format** – The format the the plugin should used to produce its output, as a string.

Module contents

This file contains an import statement for each analysis plugin.

plaso.analyzers package

Subpackages

plaso.analyzers.hashers package

Submodules

plaso.analyzers.hashers.interface module

The hasher interface.

class plaso.analyzers.hashers.interface.**BaseHasher**
Bases: object

Base class for objects that calculate hashes.

DESCRIPTION = u'Calculates a digest hash over input data.'

GetBinaryDigest()
Retrieves the digest of the hash function as a binary string.

Returns

binary hash digest calculated over the data blocks passed to Update().

Return type bytes

GetStringDigest()
Retrieves the digest of the hash function expressed as a Unicode string.

Returns

string hash digest calculated over the data blocks passed to Update(). The string consists of printable Unicode characters.

Return type str

NAME = u'base_hasher'

Update(*data*)
Updates the current state of the hasher with a new block of data.

Repeated calls to update are equivalent to one single call with the concatenation of the arguments.

Parameters **data** (*bytes*) – data with which to update the context of the hasher.

plaso.analyzers.hashers.manager module

This file contains a class for managing digest hashers for Plaso.

class plaso.analyzers.hashers.manager.**HashersManager**
Bases: object

Class that implements the hashers manager.

classmethod DeregisterHasher(*hasher_class*)
Deregisters a hasher class.

The hasher classes are identified based on their lower case name.

Parameters **hasher_class** (*type*) – class object of the hasher.

Raises KeyError – if hasher class is not set for the corresponding name.

classmethod GetHasher(*hasher_name*)
Retrieves an instance of a specific hasher.

Parameters **hasher_name** – the name of the hasher to retrieve.

Returns hasher.

Return type *BaseHasher*

Raises KeyError – if hasher class is not set for the corresponding name.

classmethod GetHasherClasses(*hasher_names=None*)
Retrieves the registered hashers.

Parameters **hasher_names** (*list[str]*) – names of the hashers to retrieve.

Yields *tuple* –

containing:

str: parser name type: next hasher class.

classmethod GetHasherNames()
Retrieves the names of all loaded hashers.

Returns hasher names.

Return type list[str]

classmethod GetHasherNamesFromString(*hasher_names_string*)
Retrieves a list of a hasher names from a comma separated string.

Takes a string of comma separated hasher names transforms it to a list of hasher names.

Parameters **hasher_names_string** (*str*) – comma separated names of hashers to enable, the string 'all' to enable all hashers or 'none' to disable all hashers.

Returns

names of valid hashers from the string, or an empty list if no valid names are found.

Return type list[str]

classmethod GetHashers(*hasher_names*)
Retrieves instances for all the specified hashers.

Parameters **hasher_names** (*list[str]*) – names of the hashers to retrieve.

Returns hashers.

Return type list[*BaseHasher*]

classmethod GetHashersInformation()
Retrieves the hashers information.

Returns

containing:

str: hasher name. str: hasher description.

Return type list[tuple]

classmethod RegisterHasher(*hasher_class*)
Registers a hasher class.

The hasher classes are identified based on their lower case name.

Parameters **hasher_class** (*type*) – class object of the hasher.

Raises KeyError – if hasher class is already set for the corresponding name.

plaso.analyzers.hashers.md5 module

The MD5 hasher implementation.

class plaso.analyzers.hashers.md5.**MD5Hasher**
Bases: *plaso.analyzers.hashers.interface.BaseHasher*

This class provides MD5 hashing functionality.

DESCRIPTION = u'Calculates an MD5 digest hash over input data.'

GetBinaryDigest()
Returns the digest of the hash function as a binary string.

Returns

binary string hash digest calculated over the data blocks passed to Update().

Return type bytes

GetStringDigest()
Returns the digest of the hash function expressed as a Unicode string.

Returns

string hash digest calculated over the data blocks passed to Update(). The string consists of printable Unicode characters.

Return type str

NAME = u'md5'

Update(*data*)
Updates the current state of the hasher with a new block of data.

Repeated calls to update are equivalent to one single call with the concatenation of the arguments.

Parameters **data** (*bytes*) – block of data with which to update the context of the hasher.

plaso.analyzers.hashers.sha1 module

The SHA-1 Hasher implementation

class plaso.analyzers.hashers.sha1.**SHA1Hasher**
Bases: *plaso.analyzers.hashers.interface.BaseHasher*

This class provides SHA-1 hashing functionality.

DESCRIPTION = u'Calculates a SHA-1 digest hash over input data.'

GetBinaryDigest()
Returns the digest of the hash function as a binary string.

Returns

binary string hash digest calculated over the data blocks passed to Update().

Return type bytes

GetStringDigest()
Returns the digest of the hash function expressed as a Unicode string.

Returns

string hash digest calculated over the data blocks passed to Update(). The string consists of printable Unicode characters.

Return type str

NAME = u'sha1'

Update(*data*)
Updates the current state of the hasher with a new block of data.

Repeated calls to update are equivalent to one single call with the concatenation of the arguments.

Parameters **data** (*bytes*) – block of data with which to update the context of the hasher.

plaso.analyzers.hashers.sha256 module

The SHA-256 Hasher implementation

class plaso.analyzers.hashers.sha256.**SHA256Hasher**
Bases: *plaso.analyzers.hashers.interface.BaseHasher*

This class provides SHA-256 hashing functionality.

DESCRIPTION = u'Calculates a SHA-256 digest hash over input data.'

GetBinaryDigest()
Returns the digest of the hash function as a binary string.

Returns

binary string hash digest calculated over the data blocks passed to Update().

Return type bytes

GetStringDigest()
Returns the digest of the hash function expressed as a Unicode string.

Returns

string hash digest calculated over the data blocks passed to Update(). The string consists of printable Unicode characters.

Return type str

NAME = u'sha256'

Update(*data*)
Updates the current state of the hasher with a new block of data.

Repeated calls to update are equivalent to one single call with the concatenation of the arguments.

Parameters **data** (*bytes*) – block of data with which to update the context of the hasher.

Module contents

This file contains an import statement for each hasher.

Submodules

plaso.analyzers.hashing_analyzer module

The hashing analyzer implementation.

class plaso.analyzers.hashing_analyzer.**HashingAnalyzer**
Bases: *plaso.analyzers.interface.BaseAnalyzer*

This class contains code for calculating file hashes of input files.

Analyze(*data*)
Updates the internal state of the analyzer, processing a block of data.

Repeated calls are equivalent to a single call with the concatenation of all the arguments.

Parameters **data** (*bytes*) – block of data from the data stream.

DESCRIPTION = u'Calculates hashes of file content.'

GetResults()
Retrieves the hashing results.

Returns results.

Return type list[*AnalyzerResult*]

INCREMENTAL_ANALYZER = True

NAME = u'hashing'

PROCESSING_STATUS_HINT = u'hashing'

Reset()
Resets the internal state of the analyzer.

SetHasherNames(*hasher_names_string*)
Sets the hashers that should be enabled.

Parameters **hasher_names_string** (*str*) – comma separated names of hashers to enable.

plaso.analyzers.interface module

Definitions to provide a whole-file processing framework.

class plaso.analyzers.interface.**BaseAnalyzer**
Bases: object

Class that provides the interface for whole-file analysis.

Analyze(*data*)
Analyzes a block of data, updating the state of the analyzer

Parameters **data** (*bytes*) – block of data to process.

DESCRIPTION = u''

GetResults()
Retrieves the results of the analysis.

Returns results.

Return type list[*AnalyzerResult*]

INCREMENTAL_ANALYZER = False

NAME = u'base_analyzer'

PROCESSING_STATUS_HINT = u'analyzing'

Reset()
Resets the internal state of the analyzer.

SIZE_LIMIT = 33554432

plaso.analyzers.manager module

This file contains a class for managing digest analyzers for Plaso.

class plaso.analyzers.manager.**AnalyzersManager**
Bases: object

Class that implements the analyzers manager.

classmethod DeregisterAnalyzer(*analyzer_class*)
Deregisters a analyzer class.

The analyzer classes are identified based on their lower case name.

Parameters **analyzer_class** (*type*) – class object of the analyzer.

Raises KeyError – if analyzer class is not set for the corresponding name.

classmethod GetAnalyzerInstance(*analyzer_name*)
Retrieves an instance of a specific analyzer.

Parameters **analyzer_name** (*str*) – name of the analyzer to retrieve.

Returns analyzer instance.

Return type *BaseAnalyzer*

Raises KeyError – if analyzer class is not set for the corresponding name.

classmethod GetAnalyzerInstances(*analyzer_names*)
Retrieves instances for all the specified analyzers.

Parameters **analyzer_names** (*list[str]*) – names of the analyzers to retrieve.

Returns analyzer instances.

Return type list[*BaseAnalyzer*]

classmethod GetAnalyzerNames()
Retrieves the names of all loaded analyzers.

Returns of analyzer names.

Return type list[str]

classmethod GetAnalyzers()
Retrieves the registered analyzers.

Yields *tuple* –

containing:

str: the uniquely identifying name of the analyzer type: the analyzer class.

classmethod GetAnalyzersInformation()
Retrieves the analyzers information.

Returns

containing:

str: analyzer name. str: analyzer description.

Return type list[tuple]

classmethod RegisterAnalyzer(*analyzer_class*)
Registers a analyzer class.

The analyzer classes are identified by their lower case name.

Parameters **analyzer_class** (*type*) – the analyzer class to register.

Raises KeyError – if analyzer class is already set for the corresponding name.

plaso.analyzers.yara_analyzer module

Analyzer that matches Yara rules.

class plaso.analyzers.yara_analyzer.**YaraAnalyzer**
Bases: *plaso.analyzers.interface.BaseAnalyzer*

Analyzer that matches Yara rules.

Analyze(*data*)
Analyzes a block of data, attempting to match Yara rules to it.

Parameters **data** (*bytes*) – a block of data.

DESCRIPTION = u'Matches Yara rules over input data.'

GetResults()
Retrieves results of the most recent analysis.

Returns results.

Return type list[*AnalyzerResult*]

INCREMENTAL_ANALYZER = False

NAME = u'yara'

PROCESSING_STATUS_HINT = u'yara scan'

Reset()
Resets the internal state of the analyzer.

SetRules(*rules_string*)
Sets the rules that the Yara analyzer will use.

Parameters **rules_string** (*str*) – Yara rule definitions

Module contents

This file contains an import statement for each analyzer.

plaso.cli package

Subpackages

plaso.cli.helpers package

Submodules

plaso.cli.helpers.analysis_plugins module

plaso.cli.helpers.artifact_definitions module

plaso.cli.helpers.data_location module

plaso.cli.helpers.database_config module

plaso.cli.helpers.date_filters module

plaso.cli.helpers.dynamic_output module

plaso.cli.helpers.elastic_output module

plaso.cli.helpers.event_filters module

plaso.cli.helpers.extraction module

plaso.cli.helpers.filter_file module

plaso.cli.helpers.hashers module

plaso.cli.helpers.interface module

plaso.cli.helpers.language module

plaso.cli.helpers.manager module

plaso.cli.helpers.mysql_4n6time_output module

plaso.cli.helpers.nsrlsvr_analysis module

plaso.cli.helpers.output_modules module

plaso.cli.helpers.parsers module

plaso.cli.helpers.profiling module

plaso.cli.helpers.server_config module

plaso.cli.helpers.sessionize_analysis module

plaso.cli.helpers.shared_4n6time_output module

class plaso.cli.logging_filter.**LoggingFilter**(*name=''*)
Bases: logging.Filter

Logging filter.

Some libraries, like binplist, introduce excessive amounts of logging that clutters the debug logs of plaso, making them almost unusable. This class implements a filter designed to make the debug logs more clutter-free.

filter(*record*)
Filter messages sent to the logging infrastructure.

Returns True if the record should be included in the logging.

Return type bool

plaso.cli.pinfo_tool module

plaso.cli.psort_tool module

plaso.cli.psteal_tool module

plaso.cli.status_view module

The status view.

class plaso.cli.status_view.**StatusView**(*output_writer*, *tool_name*)
Bases: object

Processing status view.

GetAnalysisStatusUpdateCallback()
Retrieves the analysis status update callback function.

Returns status update callback function or None.

Return type function

GetExtractionStatusUpdateCallback()
Retrieves the extraction status update callback function.

Returns status update callback function or None.

Return type function

MODE_LINEAR = u'linear'

MODE_WINDOW = u'window'

PrintExtractionStatusHeader(*processing_status*)
Prints the extraction status header.

Parameters **processing_status** (ProcessingStatus) – processing status.

PrintExtractionSummary(*processing_status*)
Prints a summary of the extraction.

Parameters **processing_status** (ProcessingStatus) – processing status.

SetMode(*mode*)
Sets the mode.

Parameters **mode** (*str*) – status view mode.

SetSourceInformation(*source_path*, *source_type*, *filter_file=None*)
Sets the source information.

Parameters

- **source_path** (*str*) – path of the source.
- **source_type** (*str*) – source type.
- **filter_file** (*Optional[str]*) – filter file.

SetStorageFileInformation(*storage_file_path*)
Sets the storage file information.

Parameters **storage_file_path** (*str*) – path to the storage file.

plaso.cli.storage_media_tool module

The storage media CLI tool.

class plaso.cli.storage_media_tool.**StorageMediaTool**(*input_reader=None*, *output_writer=None*)
Bases: *plaso.cli.tools.CLITool*

Class that implements a storage media CLI tool.

AddCredentialOptions(*argument_group*)
Adds the credential options to the argument group.

The credential options are use to unlock encrypted volumes.

Parameters **argument_group** (*argparse._ArgumentGroup*) – argparse argument group.

AddStorageMediaImageOptions(*argument_group*)
Adds the storage media image options to the argument group.

Parameters **argument_group** (*argparse._ArgumentGroup*) – argparse argument group.

AddVSSProcessingOptions(*argument_group*)
Adds the VSS processing options to the argument group.

Parameters **argument_group** (*argparse._ArgumentGroup*) – argparse argument group.

ScanSource(*source_path*)
Scans the source path for volume and file systems.

This function sets the internal source path specification and source type values.

Parameters **source_path** (*str*) – path to the source.

Returns source scanner context.

Return type dfvfs.SourceScannerContext

Raises SourceScannerError – if the format of or within the source is not supported.

plaso.cli.time_slices module

The time slice.

class plaso.cli.time_slices.**TimeSlice**(*event_timestamp*, *duration=5*)
Bases: object

Time slice.

The time slice is used to provide a context of events around an event of interest.

duration
int – duration of the time slice in minutes.

event_timestamp
int – event timestamp of the time slice or None.

end_timestamp
int – slice end timestamp or None.

start_timestamp
int – slice start timestamp or None.

plaso.cli.tool_options module

plaso.cli.tools module

The CLI tools classes.

class plaso.cli.tools.**CLIInputReader**(*encoding=u'utf-8'*)
Bases: object

CLI input reader interface.

Read()
Reads a string from the input.

Returns input.

Return type str

class plaso.cli.tools.**CLIOutputWriter**(*encoding=u'utf-8'*)
Bases: object

CLI output writer interface.

Write(*string*)
Writes a string to the output.

Parameters **string** (*str*) – output.

class plaso.cli.tools.**CLITool**(*input_reader=None*, *output_writer=None*)
Bases: object

CLI tool.

list_timezones
bool – True if the time zones should be listed.

preferred_encoding
str – preferred encoding of single-byte or multi-byte character strings, sometimes referred to as extended ASCII.

AddBasicOptions(*argument_group*)
Adds the basic options to the argument group.

Parameters **argument_group** (*argparse._ArgumentGroup*) – argparse argument group.

AddInformationalOptions(*argument_group*)
Adds the informational options to the argument group.

Parameters **argument_group** (*argparse._ArgumentGroup*) – argparse argument group.

AddLogFileOptions(*argument_group*)
Adds the log file option to the argument group.

Parameters **argument_group** (*argparse._ArgumentGroup*) – argparse argument group.

AddTimeZoneOption(*argument_group*)
Adds the time zone option to the argument group.

Parameters **argument_group** (*argparse._ArgumentGroup*) – argparse argument group.

GetCommandLineArguments()
Retrieves the command line arguments.

Returns command line arguments.

Return type str

ListTimeZones()
Lists the timezones.

NAME = u''

ParseNumericOption(*options*, *name*, *base=10*, *default_value=None*)
Parses a numeric option.

If the option is not set the default value is returned.

Parameters

- **options** (*argparse.Namespace*) – command line arguments.
- **name** (*str*) – name of the numeric option.
- **base** (*Optional[int]*) – base of the numeric value.
- **default_value** (*Optional[object]*) – default value.

Returns numeric value.

Return type int

Raises BadConfigOption – if the options are invalid.

ParseStringOption(*options*, *argument_name*, *default_value=None*)
Parses a string command line argument.

Parameters

- **options** (*argparse.Namespace*) – command line arguments.
- **argument_name** (*str*) – name of the command line argument.
- **default_value** (*Optional[object]*) – default value of the command line argument.

Returns

command line argument value. If the command line argument is not set the default value will be returned.

Return type object

Raises `BadConfigOption` – if the command line argument value cannot be converted to a Unicode string.

PrintSeparatorLine()
Prints a separator line.

class plaso.cli.tools.**FileObjectInputReader**(*file_object*, *encoding=u'utf-8'*)
Bases: *plaso.cli.tools.CLIInputReader*

File-like object input reader.

This input reader relies on the file-like object having a readline method.

Read()
Reads a string from the input.

Returns input.

Return type str

class plaso.cli.tools.**FileObjectOutputWriter**(*file_object*, *encoding=u'utf-8'*)
Bases: *plaso.cli.tools.CLIOutputWriter*

File-like object output writer.

This output writer relies on the file-like object having a write method.

Write(*string*)
Writes a string to the output.

Parameters **string** (*str*) – output.

class plaso.cli.tools.**StdinInputReader**(*encoding=u'utf-8'*)
Bases: *plaso.cli.tools.FileObjectInputReader*

Stdin input reader.

class plaso.cli.tools.**StdoutOutputWriter**(*encoding=u'utf-8'*)
Bases: *plaso.cli.tools.FileObjectOutputWriter*

Stdout output writer.

Write(*string*)
Writes a string to the output.

Parameters **string** (*str*) – output.

plaso.cli.views module

View classes.

class plaso.cli.views.**BaseTableView**(*column_names=None*, *title=None*)
Bases: `object`

Table view interface.

AddRow(*values*)
Adds a row of values.

Parameters **values** (*list[object]*) – values.

Raises ValueError – if the number of values is out of bounds.

Write(*output_writer*)
Writes the table to the output writer.

Parameters **output_writer** (*OutputWriter*) – output writer.

class plaso.cli.views.**CLITableView**(*column_names=None*, *title=None*)
Bases: *plaso.cli.views.BaseTableView*

Command line table view.

Note that currently this table view does not support more than 2 columns.

AddRow(*values*)
Adds a row of values.

Parameters **values** (*list[object]*) – values.

Raises ValueError – if the number of values is out of bounds.

Write(*output_writer*)
Writes the table to the output writer.

Parameters **output_writer** (*OutputWriter*) – output writer.

Raises RuntimeError – if the title exceeds the maximum width or if the table has more than 2 columns or if the column width is out of bounds.

class plaso.cli.views.**CLITabularTableView**(*column_names=None*, *column_sizes=None*, *title=None*)
Bases: *plaso.cli.views.BaseTableView*

Command line tabular table view interface.

AddRow(*values*)
Adds a row of values.

Parameters **values** (*list[object]*) – values.

Raises ValueError – if the number of values is out of bounds.

Write(*output_writer*)
Writes the table to the output writer.

Parameters **output_writer** (*OutputWriter*) – output writer.

class plaso.cli.views.**MarkdownTableView**(*column_names=None*, *title=None*)
Bases: *plaso.cli.views.BaseTableView*

Markdown table view.

Write(*output_writer*)
Writes the table to the output writer.

Parameters **output_writer** (*OutputWriter*) – output writer.

class plaso.cli.views.**ViewsFactory**
Bases: object

Views factory.

FORMAT_TYPE_CLI = u'cli'

FORMAT_TYPE_MARKDOWN = u'markdown'

classmethod GetTableView(*format_type*, *column_names=None*, *title=None*)
Retrieves a table view.

Parameters

- **format_type** (*str*) – table view format type.
- **column_names** (*Optional[list[str]]*) – column names.
- **title** (*Optional[str]*) – title.

Returns table view.

Return type *BaseTableView*

Module contents

plaso.containers package

Submodules

plaso.containers.analyzer_result module

Analyzer result attribute container.

class plaso.containers.analyzer_result.**AnalyzerResult**
Bases: *plaso.containers.interface.AttributeContainer*

Attribute container to store results of analyzers.

Analyzers can produce results with different attribute names. For example, the 'hashing' analyzer could produce an attribute 'md5_hash', with a value of 'd41d8cd98f00b204e9800998ecf8427e'.

analyzer_name
str – name of the analyzer that produce the result.

attribute_name
str – name of the attribute produced.

attribute_value
str – value of the attribute produced.

CONTAINER_TYPE = u'analyzer_result'

plaso.containers.artifacts module

Artifact attribute containers.

class plaso.containers.artifacts.**ArtifactAttributeContainer**
Bases: *plaso.containers.interface.AttributeContainer*

Base class to represent an artifact attribute container.

class plaso.containers.artifacts.**EnvironmentVariableArtifact**(*case_sensitive=True*, *name=None*, *value=None*)
Bases: *plaso.containers.artifacts.ArtifactAttributeContainer*

Environment variable artifact attribute container.

Also see: https://en.wikipedia.org/wiki/Environment_variable

case_sensitive
bool – True if environment variable name is case sensitive.

name
str – environment variable name e.g. 'SystemRoot' as in '%SystemRoot%' or 'HOME' in '$HOME'.

value
str – environment variable value e.g. 'C:Windows' or '/home/user'.

CONTAINER_TYPE = u'environment_variable'

class plaso.containers.artifacts.**HostnameArtifact**(*name=None*, *schema=u'DNS'*)
Bases: *plaso.containers.artifacts.ArtifactAttributeContainer*

Hostname artifact attribute container.

Also see: https://en.wikipedia.org/wiki/Hostname http://cybox.mitre.org/language/version2.1/xsddocs/objects/Hostname_Object.html

name
str – name of the host according to the naming schema.

schema
str – naming schema e.g. DNS, NIS, SMB/NetBIOS.

CONTAINER_TYPE = u'hostname'

class plaso.containers.artifacts.**SystemConfigurationArtifact**(*code_page=None*, *time_zone=None*)
Bases: *plaso.containers.artifacts.ArtifactAttributeContainer*

System configuration artifact attribute container.

The system configuration contains the configuration data of a specific system installation e.g. Windows or Linux.

code_page
str – system code page.

hostname
HostnameArtifact – hostname.

keyboard_layout
str – keyboard layout.

operating_system
str – operating system for example "MacOS" or "Windows".

operating_system_product
str – operating system product for example "Windows XP".

operating_system_version
str – operating system version for example "10.9.2" or "8.1".

time_zone
str – system time zone.

user_accounts
list[UserAccountArtifact] – user accounts.

CONTAINER_TYPE = u'system_configuration'

class plaso.containers.artifacts.**UserAccountArtifact**(*full_name=None*, *group_identifier=None*, *identifier=None*, *user_directory=None*, *username=None*)
Bases: *plaso.containers.artifacts.ArtifactAttributeContainer*

User account artifact attribute container.

Also see: http://cybox.mitre.org/language/version2.1/xsddocs/objects/ User_Account_Object.html

full_name
str – name describing the user e.g. full name.

group_identifier
str – identifier of the primary group the user is part of.

identifier
str – user identifier.

user_directory
str – path of the user (or home or profile) directory.

username
str – name uniquely identifying the user.

CONTAINER_TYPE = u'user_account'

plaso.containers.errors module

Error attribute containers.

class plaso.containers.errors.**ExtractionError**(*message=None*, *parser_chain=None*, *path_spec=None*)
Bases: *plaso.containers.interface.AttributeContainer*

Extraction error attribute container.

message
str – error message.

parser_chain
str – parser chain to which the error applies.

path_spec
dfvfs.PathSpec – path specification of the file entry to which the error applies.

CONTAINER_TYPE = u'extraction_error'

plaso.containers.event_sources module

Event source attribute containers.

class plaso.containers.event_sources.**EventSource**(*path_spec=None*)
Bases: *plaso.containers.interface.AttributeContainer*

Event source attribute container.

The event source object contains information about where a specific event originates e.g. a file, the $STANDARD_INFORMATION MFT attribute, or Application Compatibility cache.

data_type
str – attribute container type indicator.

file_entry_type
str – dfVFS file entry type.

path_spec
dfvfs.PathSpec – path specification.

CONTAINER_TYPE = u'event_source'

DATA_TYPE = None

__lt__(*other*)
Compares if the event source attribute container is less than the other.

Parameters **other** (EventSource) – event source attribute container to compare to.

Returns True if the event source attribute container is less than the other.

Return type bool

class plaso.containers.event_sources.**FileEntryEventSource**(*path_spec=None*)
Bases: *plaso.containers.event_sources.EventSource*

File entry event source.

The file entry event source is an event source that represents a file within a file system.

DATA_TYPE = u'file_entry'

plaso.containers.events module

Event attribute containers.

class plaso.containers.events.**EventData**(*data_type=None*)
Bases: *plaso.containers.interface.AttributeContainer*

Event data attribute container.

data_type
str – event data type indicator.

offset
int – offset relative to the start of the data stream where the event data is stored.

query
str – query that was used to obtain the event data.

CONTAINER_TYPE = u'event_data'

class plaso.containers.events.**EventObject**
Bases: *plaso.containers.interface.AttributeContainer*

Event attribute container.

The framework is designed to parse files and create events from individual records, log lines or keys extracted from files. The event object provides an extensible data store for event attributes.

data_type
str – event data type indicator.

display_name
str – display friendly version of the path specification.

filename
str – name of the file related to the event.

hostname
str – name of the host related to the event.

inode
int – inode of the file related to the event.

offset
int – offset of the event data.

pathspec
dfvfs.PathSpec – path specification of the file related to the event.

tag
EventTag – event tag.

timestamp
int – timestamp, which contains the number of microseconds since January 1, 1970, 00:00:00 UTC.

CONTAINER_TYPE = u'event'

DATA_TYPE = None

GetEventDataIdentifier()
Retrieves the identifier of the event data associated with the event.

The event data identifier is a storage specific value that should not be serialized.

Returns event identifier or None when not set.

Return type *AttributeContainerIdentifier*

SetEventDataIdentifier(*event_data_identifier*)
Sets the identifier of the event data associated with the event.

The event data identifier is a storage specific value that should not be serialized.

Parameters **event_data_identifier** (AttributeContainerIdentifier) – event identifier.

class plaso.containers.events.**EventTag**(*comment=None*)
Bases: *plaso.containers.interface.AttributeContainer*

Event tag attribute container.

comment
str – comments.

event_entry_index
int – serialized data stream entry index of the event, this attribute is used by the ZIP and GZIP storage files to uniquely identify the event linked to the tag.

event_stream_number
int – number of the serialized event stream, this attribute is used by the ZIP and GZIP storage files to uniquely identify the event linked to the tag.

labels
list[str] – labels, such as "malware", "application_execution".

AddComment(*comment*)
Adds a comment to the event tag.

Parameters **comment** (*str*) – comment.

AddLabel(*label*)
Adds a label to the event tag.

Parameters **label** (*str*) – label.

Raises ValueError – if a label is malformed.

AddLabels(*labels*)
Adds labels to the event tag.

Parameters **labels** (*list[str]*) – labels.

Raises ValueError – if a label is malformed.

CONTAINER_TYPE = u'event_tag'

classmethod CopyTextToLabel(*text*, *prefix=u''*)
Copies a string to a label.

A label only supports a limited set of characters therefore unsupported characters are replaced with an underscore.

Parameters

- **text** (*str*) – label text.
- **prefix** (*Optional[str]*) – label prefix.

Returns label.

Return type str

CopyToDict()
Copies the event tag to a dictionary.

Returns event tag attributes.

Return type dict[str, object]

GetEventIdentifier()
Retrieves the identifier of the event associated with the event tag.

The event identifier is a storage specific value that should not be serialized.

Returns event identifier or None when not set.

Return type *AttributeContainerIdentifier*

SetEventIdentifier(*event_identifier*)
Sets the identifier of the event associated with the event tag.

The event identifier is a storage specific value that should not be serialized.

Parameters **event_identifier** (AttributeContainerIdentifier) – event identifier.

plaso.containers.interface module

The attribute container interface.

class plaso.containers.interface.**AttributeContainer**
Bases: object

The attribute container interface.

This is the the base class for those object that exists primarily as a container of attributes with basic accessors and mutators.

The CONTAINER_TYPE class attribute contains a string that identifies the container type e.g. the container type "event" identifiers an event object.

Attributes are public class members of an serializable type. Protected and private class members are not to be serialized.

CONTAINER_TYPE = None

CopyFromDict(*attributes*)
Copies the attribute container from a dictionary.

Parameters **attributes** (*dict[str, object]*) – attribute values per name.

CopyToDict()
Copies the attribute container to a dictionary.

Returns attribute values per name.

Return type dict[str, object]

GetAttributeNames()
Retrieves the names of all attributes.

Returns attribute names.

Return type list[str]

GetAttributeValuesHash()
Retrieves a comparable string of the attribute values.

Returns hash of comparable string of the attribute values.

Return type int

GetAttributeValuesString()
Retrieves a comparable string of the attribute values.

Returns comparable string of the attribute values.

Return type str

GetAttributes()
Retrieves the attribute names and values.

Attributes that are set to None are ignored.

Yields *tuple[str, object]* – attribute name and value.

GetIdentifier()
Retrieves the identifier.

The identifier is a storage specific value that should not be serialized.

Returns an unique identifier for the container.

Return type *AttributeContainerIdentifier*

GetSessionIdentifier()
Retrieves the session identifier.

The session identifier is a storage specific value that should not be serialized.

Returns session identifier.

Return type str

SetIdentifier(*identifier*)
Sets the identifier.

The identifier is a storage specific value that should not be serialized.

Parameters **identifier** (AttributeContainerIdentifier) – identifier.

SetSessionIdentifier(*session_identifier*)
Sets the session identifier.

The session identifier is a storage specific value that should not be serialized.

Parameters **session_identifier** (*str*) – session identifier.

class plaso.containers.interface.**AttributeContainerIdentifier**
Bases: object

The attribute container identifier.

The identifier is used to uniquely identify attribute containers. The value should be unique at runtime and in storage.

CopyToString()
Copies the identifier to a string representation.

Returns unique identifier or None.

Return type str

plaso.containers.manager module

This file contains the attribute container manager class.

class plaso.containers.manager.**AttributeContainersManager**
Bases: object

Class that implements the attribute container manager.

classmethod DeregisterAttributeContainer(*attribute_container_class*)
Deregisters an attribute container class.

The attribute container classes are identified based on their lower case containter type.

Parameters **attribute_container_class** (*type*) – attribute container class.

Raises KeyError – if attribute container class is not set for the corresponding container type.

classmethod GetAttributeContainer(*container_type*)
Retrieves the attribute container for a specific container type.

Parameters **container_type** (*str*) – container type.

Returns attribute container.

Return type *AttributeContainer*

classmethod RegisterAttributeContainer(*attribute_container_class*)
Registers a attribute container class.

The attribute container classes are identified based on their lower case container type.

Parameters **attribute_container_class** (*type*) – attribute container class.

Raises KeyError – if attribute container class is already set for the corresponding container type.

classmethod RegisterAttributeContainers(*attribute_container_classes*)
Registers attribute container classes.

The attribute container classes are identified based on their lower case container type.

Parameters **attribute_container_classes** (*list[type]*) – attribute container classes.

Raises KeyError – if attribute container class is already set for the corresponding container type.

plaso.containers.plist_event module

Plist event attribute containers.

class plaso.containers.plist_event.**PlistTimeEventData**
Bases: *plaso.containers.events.EventData*

Plist event data attribute container.

desc
str – description.

host
str – hostname.

key
str – name of plist key.

root
str – path from the root to this plist key.

user
str – unique username.

DATA_TYPE = u'plist:key'

plaso.containers.reports module

Report related attribute container definitions.

class plaso.containers.reports.**AnalysisReport**(*plugin_name=None*, *text=None*)
Bases: *plaso.containers.interface.AttributeContainer*

Analysis report attribute container.

filter_string
str – event filter expression.

plugin_name
str – name of the analysis plugin that generated the report.

report_array
array[str] – ???

report_dict
dict[str] – ???

text
str – report text.

time_compiled
int – timestamp of the date and time the report was compiled.

CONTAINER_TYPE = u'analysis_report'

CopyToDict()
Copies the attribute container to a dictionary.

Returns attribute values per name.

Return type dict[str, object]

GetString()
Retrieves a string representation of the report.

Returns string representation of the report.

Return type str

plaso.containers.sessions module

Session related attribute container definitions.

class plaso.containers.sessions.**Session**
Bases: *plaso.containers.interface.AttributeContainer*

Session attribute container.

aborted
bool – True if the session was aborted.

analysis_reports_counter
collections.Counter – number of analysis reports per analysis plugin.

command_line_arguments
str – command line arguments.

completion_time
int – time that the session was completed. Contains the number of micro seconds since January 1, 1970, 00:00:00 UTC.

debug_mode
bool – True if debug mode was enabled.

enabled_parser_names
list[str] – parser and parser plugin names that were enabled.

event_labels_counter
collections.Counter – number of event tags per label.

filter_file
str – path to a file with find specifications.

identifier
str – unique identifier of the session.

parser_filter_expression
str – parser filter expression.

parsers_counter
collections.Counter – number of events per parser or parser plugin.

preferred_encoding
str – preferred encoding.

preferred_time_zone
str – preferred time zone.

preferred_year
int – preferred year.

product_name
str – name of the product that created the session e.g. 'log2timeline'.

product_version
str – version of the product that created the session.

start_time
int – time that the session was started. Contains the number of micro seconds since January 1, 1970, 00:00:00 UTC.

CONTAINER_TYPE = u'session'

CopyAttributesFromSessionCompletion(*session_completion*)
Copies attributes from a session completion.

Parameters **session_completion** (SessionCompletion) – session completion attribute container.

Raises ValueError – if the identifier fo the session completion does not match that of the session.

CopyAttributesFromSessionStart(*session_start*)
Copies attributes from a session start.

Parameters **session_start** (SessionStart) – session start attribute container.

CreateSessionCompletion()
Creates a session completion.

Returns session completion attribute container.

Return type *SessionCompletion*

CreateSessionStart()
Creates a session start.

Returns session start attribute container.

Return type *SessionStart*

class plaso.containers.sessions.**SessionCompletion**(*identifier=None*)
Bases: *plaso.containers.interface.AttributeContainer*

Session completion attribute container.

aborted
bool – True if the session was aborted.

analysis_reports_counter
collections.Counter – number of analysis reports per analysis plugin.

event_labels_counter
collections.Counter – number of event tags per label.

identifier
str – unique identifier of the session.

parsers_counter
collections.Counter – number of events per parser or parser plugin.

timestamp
int – time that the session was completed. Contains the number of micro seconds since January 1, 1970, 00:00:00 UTC.

CONTAINER_TYPE = u'session_completion'

class plaso.containers.sessions.**SessionStart**(*identifier=None*)
Bases: *plaso.containers.interface.AttributeContainer*

Session start attribute container.

command_line_arguments
str – command line arguments.

debug_mode
bool – True if debug mode was enabled.

enabled_parser_names
list[str] – parser and parser plugin names that were enabled.

filter_file
str – path to a file with find specifications.

identifier
str – unique identifier of the session.

parser_filter_expression
str – parser filter expression.

preferred_encoding
str – preferred encoding.

preferred_time_zone
str – preferred time zone.

preferred_year
int – preferred year.

product_name
str – name of the product that created the session e.g. 'log2timeline'.

product_version
str – version of the product that created the session.

timestamp
int – time that the session was started. Contains the number of micro seconds since January 1, 1970, 00:00:00 UTC.

CONTAINER_TYPE = u'session_start'

plaso.containers.shell_item_events module

Shell item event attribute container.

class plaso.containers.shell_item_events.**ShellItemFileEntryEventData**
Bases: *plaso.containers.events.EventData*

Shell item file entry event data attribute container.

name
str – name of the file entry shell item.

long_name
str – long name of the file entry shell item.

localized_name
str – localized name of the file entry shell item.

file_reference
str – NTFS file reference, in the format: "MTF entry - sequence number".

shell_item_path
str – shell item path.

origin
str – origin of the event.

DATA_TYPE = u'windows:shell_item:file_entry'

plaso.containers.storage_media module

Storage media related attribute container definitions.

class plaso.containers.storage_media.**MountPoint**(*mount_path=None*, *path_specification=None*)
Bases: *plaso.containers.interface.AttributeContainer*

Mount point attribute container.

mount_path
str – path where the path specification is mounted, such as "/mnt/image" or "C:".

path_spec
dfvfs.PathSpec – path specification.

CONTAINER_TYPE = u'mount_point'

plaso.containers.tasks module

Task related attribute container definitions.

class plaso.containers.tasks.**Task**(*session_identifier=None*)
Bases: *plaso.containers.interface.AttributeContainer*

Task attribute container.

A task describes a piece of work for a multi processing worker process e.g. to process a path specification or to analyze an event.

aborted
bool – True if the session was aborted.

completion_time
int – time that the task was completed. Contains the number of micro seconds since January 1, 1970, 00:00:00 UTC.

file_entry_type
str – dfVFS type of the file entry the path specification is referencing.

identifier
str – unique identifier of the task.

last_processing_time
int – the last time the task was marked as being processed as number of milliseconds since January 1, 1970, 00:00:00 UTC.

merge_priority
int – priority used for the task storage file merge, where a lower value indicates a higher priority to merge.

original_task_identifier
str – the identifier of the task that this task is an attempt to retry, or None if this task isn't a retry.

path_spec
dfvfs.PathSpec – path specification.

retried
bool – True if this task been retried.

session_identifier
str – the identifier of the session the task is part of.

start_time
int – time that the task was started. Contains the number of micro seconds since January 1, 1970, 00:00:00 UTC.

storage_file_size
int – size of the storage file in bytes.

CONTAINER_TYPE = u'task'

CreateRetry()
Creates a new task that's an attempt to retry the original task.

Returns a task that's a retry of the existing task.

Return type *Task*

CreateTaskCompletion()
Creates a task completion.

Returns task completion attribute container.

Return type *TaskCompletion*

CreateTaskStart()
Creates a task start.

Returns task start attribute container.

Return type *TaskStart*

UpdateProcessingTime()
Updates the processing time to now.

__lt__(*other*)
Compares if the task attribute container is less than the other.

Parameters **other** (Task) – task attribute container to compare to.

Returns True if the task attribute container is less than the other.

Return type bool

class plaso.containers.tasks.**TaskCompletion**(*identifier=None*, *session_identifier=None*)
Bases: *plaso.containers.interface.AttributeContainer*

Task completion attribute container.

aborted
bool – True if the session was aborted.

identifier
str – unique identifier of the task.

session_identifier
str – the identifier of the session the task is part of.

timestamp
int – time that the task was completed. Contains the number of micro seconds since January 1, 1970, 00:00:00 UTC.

CONTAINER_TYPE = u'task_completion'

class plaso.containers.tasks.**TaskStart**(*identifier=None*, *session_identifier=None*)
Bases: *plaso.containers.interface.AttributeContainer*

Task start attribute container.

identifier
str – unique identifier of the task.

session_identifier
str – the identifier of the session the task is part of.

timestamp
int – time that the task was started. Contains the number of micro seconds since January 1, 1970, 00:00:00 UTC.

CONTAINER_TYPE = u'task_start'

plaso.containers.time_events module

Time-based event attribute containers.

class plaso.containers.time_events.**DateTimeValuesEvent**(*date_time*, *date_time_description*, *data_type=None*, *time_zone=None*)
Bases: *plaso.containers.time_events.TimestampEvent*

dfDateTime date time values-based event attribute container.

class plaso.containers.time_events.**PythonDatetimeEvent**(*datetime_value*, *date_time_description*, *data_type=None*, *time_zone=None*)
Bases: *plaso.containers.time_events.DateTimeValuesEvent*

Python datetime-based event attribute container.

class plaso.containers.time_events.**TimestampEvent**(*timestamp*, *timestamp_description*, *data_type=None*)
Bases: *plaso.containers.events.EventObject*

Plaso timestamp-based event attribute container.

data_type
str – event data type.

timestamp
int – timestamp, which contains the number of microseconds since January 1, 1970, 00:00:00 UTC.

timestamp_desc
str – description of the meaning of the timestamp.

plaso.containers.windows_events module

Windows event data attribute containers.

class plaso.containers.windows_events.**WindowsDistributedLinkTrackingEventData**(*uuid*, *origin*)

Bases: *plaso.containers.events.EventData*

Windows distributed link event data attribute container.

mac_address
str – MAC address stored in the UUID.

origin
str – origin of the event (event source). E.g. the path of the corresponding LNK file or file reference MFT entry with the corresponding NTFS $OBJECT_ID attribute.

uuid
str – UUID.

DATA_TYPE = u'windows:distributed_link_tracking:creation'

class plaso.containers.windows_events.**WindowsRegistryEventData**

Bases: *plaso.containers.events.EventData*

Windows Registry event data attribute container.

key_path
str – Windows Registry key path.

regvalue
dict[str, object] – values in the key.

source_append
str – text to append to the source_long of the event.

urls
list[str] – URLs.

DATA_TYPE = u'windows:registry:key_value'

class plaso.containers.windows_events.**WindowsRegistryInstallationEventData**

Bases: *plaso.containers.events.EventData*

Windows installation event data attribute container.

key_path
str – Windows Registry key path.

owner
str – owner.

product_name
str – product name.

service_pack
str – service pack.

version
str – version.

DATA_TYPE = u'windows:registry:installation'

class plaso.containers.windows_events.**WindowsRegistryListEventData**
Bases: *plaso.containers.events.EventData*

Windows Registry list event data attribute container.

Windows Registry list event data is used to store a MRU.

key_path
str – Windows Registry key path.

list_name
str – name of the list.

list_values
str – values in the list.

value_name
str – Windows Registry value name.

DATA_TYPE = u'windows:registry:list'

class plaso.containers.windows_events.**WindowsRegistryServiceEventData**
Bases: *plaso.containers.events.EventData*

Windows Registry service event data attribute container.

key_path
str – Windows Registry key path.

offset
int – data offset of the Windows Registry key or value.

regvalue
dict[str, str] – values of a key.

urls
Optional[list[str]] – URLs.

DATA_TYPE = u'windows:registry:service'

class plaso.containers.windows_events.**WindowsVolumeEventData**
Bases: *plaso.containers.events.EventData*

Windows volume event data attribute container.

device_path
str – volume device path.

origin
str – origin of the event (event source), for example the corresponding Prefetch file name.

serial_number
str – volume serial number.

DATA_TYPE = u'windows:volume:creation'

Module contents

This file contains an import statement for each attribute container.

plaso.engine package

Submodules

plaso.engine.configurations module

Processing configuration classes.

class plaso.engine.configurations.**CredentialConfiguration**(*credential_data=None*, *credential_type=None*, *path_spec=None*)

Bases: *plaso.containers.interface.AttributeContainer*

Configuration settings for a credential.

credential_data
bytes – credential data.

credential_type
str – credential type.

path_spec
dfvfs.PathSpec – path specification.

CONTAINER_TYPE = u'credential_configuration'

class plaso.engine.configurations.**EventExtractionConfiguration**

Bases: *plaso.containers.interface.AttributeContainer*

Configuration settings for event extraction.

These settings are primarily used by the parser mediator.

filter_object
objectfilter.Filter – filter that specifies which events to include.

text_prepend
str – text to prepend to every event.

CONTAINER_TYPE = u'event_extraction_configuration'

class plaso.engine.configurations.**ExtractionConfiguration**

Bases: *plaso.containers.interface.AttributeContainer*

Configuration settings for extraction.

These settings are primarily used by the extraction worker.

hasher_file_size_limit
int – maximum file size that hashers should process, where 0 or None represents unlimited.

hasher_names_string
str – comma separated string of names of hashers to use during processing.

process_archives
bool – True if archive files should be scanned for file entries.

process_compressed_streams
bool – True if file content in compressed streams should be processed.

yara_rules_string
str – Yara rule definitions.

CONTAINER_TYPE = u'extraction_configuration'

class plaso.engine.configurations.**InputSourceConfiguration**
Bases: *plaso.containers.interface.AttributeContainer*

Configuration settings of an input source.

mount_path
str – path of a "mounted" directory input source.

CONTAINER_TYPE = u'input_source'

class plaso.engine.configurations.**ProcessingConfiguration**
Bases: *plaso.containers.interface.AttributeContainer*

Configuration settings for processing.

credentials
list[CredentialConfiguration] – credential configurations.

data_location
str – path to the data files.

debug_output
bool – True if debug output should be enabled.

event_extraction
EventExtractionConfiguration – event extraction configuration.

extraction
ExtractionConfiguration – extraction configuration.

filter_file
str – path to a file with find specifications.

input_source
InputSourceConfiguration – input source configuration.

log_filename
str – name of the log file.

parser_filter_expression
str – parser filter expression, where None represents all parsers and plugins.

preferred_year
int – preferred initial year value for year-less date and time values.

profiling
ProfilingConfiguration – profiling configuration.

temporary_directory
str – path of the directory for temporary files.

CONTAINER_TYPE = u'processing_configuration'

class plaso.engine.configurations.**ProfilingConfiguration**
Bases: *plaso.containers.interface.AttributeContainer*

Configuration settings for profiling.

directory
str – path to the directory where the profiling sample files should be stored.

profilers
set(str) – names of the profilers to enable. Supported profilers are:

- 'guppy', which profiles memory usage using guppy;

- 'memory', which profiles memory usage;
- 'parsers', which profiles CPU time consumed by individual parsers;
- 'processing', which profiles CPU time consumed by different parts of processing;
- 'serializers', which profiles CPU time consumed by individual serializers.

sample_rate
int – the profiling sample rate. Contains the number of event sources processed.

CONTAINER_TYPE = u'profiling_configuration'

HaveProfileMemory()
Determines if memory profiling is configured.

Returns True if memory profiling is configured.

Return type bool

HaveProfileMemoryGuppy()
Determines if memory profiling with guppy is configured.

Returns True if memory profiling with guppy is configured.

Return type bool

HaveProfileParsers()
Determines if parsers profiling is configured.

Returns True if parsers profiling is configured.

Return type bool

HaveProfileProcessing()
Determines if processing profiling is configured.

Returns True if processing profiling is configured.

Return type bool

HaveProfileSerializers()
Determines if serializers profiling is configured.

Returns True if serializers profiling is configured.

Return type bool

plaso.engine.engine module

plaso.engine.extractors module

plaso.engine.filter_file module

Filter file.

class plaso.engine.filter_file.**FilterFile**(*path*)
Bases: object

Filter file.

A filter file contains one or more path filters.

A path filter may contain path expansion attributes. Such an attribute is defined as anything within a curly bracket, for example "System{my_attribute}PathKeyname". If the attribute "my_attribute" is defined its runtime value will be replaced with placeholder in the path filter such as "SystemMyValuePathKeyname".

If the path filter needs to have curly brackets in the path then these need to be escaped with another curly bracket, for example "System{my_attribute}{{123-AF25-E523}}KeyName", where "{{123-AF25-E523}}" will be replaced with "{123-AF25-E523}" at runtime.

BuildFindSpecs(*environment_variables=None*)
Build find specification from a filter file.

Parameters **environment_variables**(*Optional[list[EnvironmentVariableArtifact]]*) – environment variables.

Returns find specification.

Return type list[dfvfs.FindSpec]

plaso.engine.knowledge_base module

The artifact knowledge base object.

The knowledge base is filled by user provided input and the pre-processing phase. It is intended to provide successive phases, like the parsing and analysis phases, with essential information like e.g. the timezone and codepage of the source data.

class plaso.engine.knowledge_base.**KnowledgeBase**
Bases: object

Class that implements the artifact knowledge base.

AddEnvironmentVariable(*environment_variable*)
Adds an environment variable.

Parameters **environment_variable** (EnvironmentVariableArtifact) – environment variable artifact.

Raises KeyError – if the environment variable already exists.

AddUserAccount(*user_account*, *session_identifier=0*)
Adds an user account.

Parameters

- **user_account** (UserAccountArtifact) – user account artifact.
- **session_identifier** (*Optional[str]*) – session identifier, where CURRENT_SESSION represents the active session.

Raises KeyError – if the user account already exists.

CURRENT_SESSION = 0

GetEnvironmentVariable(*name*)
Retrieves an environment variable.

Parameters **name** (*str*) – name of the environment variable.

Returns

environment variable artifact or None if there was no value set for the given name.

Return type *EnvironmentVariableArtifact*

GetEnvironmentVariables()
Retrieves the environment variables.

Returns environment variable artifacts.

Return type list[*EnvironmentVariableArtifact*]

GetHostname(*session_identifier=0*)
Retrieves the hostname related to the event.

If the hostname is not stored in the event it is determined based on the preprocessing information that is stored inside the storage file.

Parameters **session_identifier** (*Optional[str]*) – session identifier, where CURRENT_SESSION represents the active session.

Returns hostname.

Return type str

GetStoredHostname()
Retrieves the stored hostname.

The hostname is determined based on the preprocessing information that is stored inside the storage file.

Returns hostname.

Return type str

GetSystemConfigurationArtifact(*session_identifier=0*)
Retrieves the knowledge base as a system configuration artifact.

Parameters **session_identifier** (*Optional[str]*) – session identifier, where CURRENT_SESSION represents the active session.

Returns system configuration artifact.

Return type *SystemConfigurationArtifact*

GetUsernameByIdentifier(*user_identifier*, *session_identifier=0*)
Retrieves the username based on an user identifier.

Parameters

- **user_identifier** (*str*) – user identifier, either a UID or SID.
- **session_identifier** (*Optional[str]*) – session identifier, where CURRENT_SESSION represents the active session.

Returns username.

Return type str

GetUsernameForPath(*path*)
Retrieves a username for a specific path.

This is determining if a specific path is within a user's directory and returning the username of the user if so.

Parameters **path** (*str*) – path.

Returns

username or None if the path does not appear to be within a user's directory.

Return type str

GetValue(*identifier*, *default_value=None*)
Retrieves a value by identifier.

Parameters

- **identifier** (*str*) – case insensitive unique identifier for the value.
- **default_value** (*object*) – default value.

Returns value or default value if not available.

Return type object

Raises TypeError – if the identifier is not a string type.

HasUserAccounts()
Determines if the knowledge base contains user accounts.

Returns True if the knowledge base contains user accounts.

Return type bool

ReadSystemConfigurationArtifact(*system_configuration*, *session_identifier=0*)
Reads the knowledge base values from a system configuration artifact.

Note that this overwrites existing values in the knowledge base.

Parameters

- **system_configuration** (SystemConfigurationArtifact) – system configuration artifact.
- **session_identifier** (*Optional[str]*) – session identifier, where CURRENT_SESSION represents the active session.

SetCodepage(*codepage*)
Sets the codepage.

Parameters **codepage** (*str*) – codepage.

Raises ValueError – if the codepage is not supported.

SetEnvironmentVariable(*environment_variable*)
Sets an environment variable.

Parameters **environment_variable** (EnvironmentVariableArtifact) – environment variable artifact.

SetHostname(*hostname*, *session_identifier=0*)
Sets a hostname.

Parameters

- **hostname** (HostnameArtifact) – hostname artifact.
- **session_identifier** (*Optional[str]*) – session identifier, where CURRENT_SESSION represents the active session.

SetTimeZone(*time_zone*)
Sets the time zone.

Parameters **time_zone** (*str*) – time zone.

Raises ValueError – if the timezone is not supported.

SetValue(*identifier*, *value*)
Sets a value by identifier.

Parameters

- **identifier** (*str*) – case insensitive unique identifier for the value.
- **value** (*object*) – value.

Raises TypeError – if the identifier is not a string type.

codepage
str – codepage of the current session.

hostname
str – hostname of the current session.

timezone
datetime.tzinfo – timezone of the current session.

user_accounts
list[UserAccountArtifact] – user accounts of the current session.

year
int – year of the current session.

plaso.engine.path_helper module

The path helper.

class plaso.engine.path_helper.**PathHelper**
Bases: object

Class that implements the path helper.

classmethod ExpandWindowsPath(*path*, *environment_variables*)
Expands a Windows path containing environment variables.

Parameters

- **path** (*str*) – Windows path with environment variables.
- **environment_variables** (*list[EnvironmentVariableArtifact]*) – environment variables.

Returns expanded Windows path.

Return type str

classmethod GetDisplayNameForPathSpec(*path_spec*, *mount_path=None*, *text_prepend=None*)
Retrieves the display name of a path specification.

Parameters

- **path_spec** (*dfvfs.PathSpec*) – path specification.
- **mount_path** (*Optional[str]*) – path where the file system that is used by the path specification is mounted, such as "/mnt/image". The mount path will be stripped from the absolute path defined by the path specification.
- **text_prepend** (*Optional[str]*) – text to prepend.

Returns human readable version of the path specification or None.

Return type str

classmethod GetRelativePathForPathSpec(*path_spec*, *mount_path=None*)
Retrieves the relative path of a path specification.

If a mount path is defined the path will be relative to the mount point, otherwise the path is relative to the root of the file system that is used by the path specification.

Parameters

- **path_spec** (*dfvfs.PathSpec*) – path specification.
- **mount_path** (*Optional[str]*) – path where the file system that is used by the path specification is mounted, such as "/mnt/image". The mount path will be stripped from the absolute path defined by the path specification.

Returns relative path or None.

Return type str

plaso.engine.plaso_queue module

Queue management implementation for Plaso.

This file contains an implementation of a queue used by plaso for queue management.

The queue has been abstracted in order to provide support for different implementations of the queueing mechanism, to support multi processing and scalability.

class plaso.engine.plaso_queue.**Queue**
Bases: object

Class that implements the queue interface.

Close(*abort=False*)
Closes the queue.

Parameters **abort** (*Optional[bool]*) – whether the Close is the result of an abort condition. If True, queue contents may be lost.

IsEmpty()
Determines if the queue is empty.

Open()
Opens the queue, ready to enqueue or dequeue items.

PopItem()
Pops an item off the queue.

Raises QueueEmpty – when the queue is empty.

PushItem(*item*, *block=True*)
Pushes an item onto the queue.

Parameters

- **item** (*object*) – item to add.
- **block** (*bool*) – whether to block if the queue is full.

Raises QueueFull – if the queue is full, and the item could not be added.

class plaso.engine.plaso_queue.**QueueAbort**
Bases: object

Class that implements a queue abort.

plaso.engine.process_info module

This file contains a class to get process information.

class plaso.engine.process_info.**ProcessInfo**(*pid*)
Bases: object

Provides information about a running process.

GetUsedMemory()
Retrieves the amount of memory used by the process.

Returns

amount of memory in bytes used by the process or None if not available.

Return type int

plaso.engine.processing_status module

Processing status classes.

class plaso.engine.processing_status.**ProcessStatus**
Bases: object

The status of an individual process.

display_name
str – human readable of the file entry currently being processed by the process.

identifier
str – process identifier.

last_running_time
int – timestamp of the last update when the process had a running process status.

number_of_consumed_errors
int – total number of errors consumed by the process.

number_of_consumed_errors_delta
int – number of errors consumed by the process since the last status update.

number_of_consumed_event_tags
int – total number of event tags consumed by the process.

number_of_consumed_event_tags_delta
int – number of event tags consumed by the process since the last status update.

number_of_consumed_events
int – total number of events consumed by the process.

number_of_consumed_events_delta
int – number of events consumed by the process since the last status update.

number_of_consumed_reports
int – total number of event reports consumed by the process.

number_of_consumed_reports_delta
int – number of event reports consumed by the process since the last status update.

number_of_consumed_sources
int – total number of event sources consumed by the process.

number_of_consumed_sources_delta
int – number of event sources consumed by the process since the last status update.

number_of_produced_errors
int – total number of errors produced by the process.

number_of_produced_errors_delta
int – number of errors produced by the process since the last status update.

number_of_produced_event_tags
int – total number of event tags produced by the process.

number_of_produced_event_tags_delta
int – number of event tags produced by the process since the last status update.

number_of_produced_events
int – total number of events produced by the process.

number_of_produced_events_delta
int – number of events produced by the process since the last status update.

number_of_produced_reports
int – total number of event reports produced by the process.

number_of_produced_reports_delta
int – number of event reports produced by the process since the last status update.

number_of_produced_sources
int – total number of event sources produced by the process.

number_of_produced_sources_delta
int – number of event sources produced by the process since the last status update.

pid
int – process identifier (PID).

status
str – human readable status indication e.g. 'Hashing', 'Idle'.

used_memory
int – size of used memory in bytes.

UpdateNumberOfErrors(*number_of_consumed_errors*, *number_of_produced_errors*)
Updates the number of errors.

Parameters

- **number_of_consumed_errors** (*int*) – total number of errors consumed by the process.
- **number_of_produced_errors** (*int*) – total number of errors produced by the process.

Returns True if either number of errors has increased.

Return type bool

Raises ValueError – if the consumed or produced number of errors is smaller than the value of the previous update.

UpdateNumberOfEventReports(*number_of_consumed_reports*, *number_of_produced_reports*)
Updates the number of event reports.

Parameters

- **number_of_consumed_reports** (*int*) – total number of event reports consumed by the process.
- **number_of_produced_reports** (*int*) – total number of event reports produced by the process.

Returns True if either number of event reports has increased.

Return type bool

Raises ValueError – if the consumed or produced number of event reports is smaller than the value of the previous update.

UpdateNumberOfEventSources(*number_of_consumed_sources*, *number_of_produced_sources*)
Updates the number of event sources.

Parameters

- **number_of_consumed_sources** (*int*) – total number of event sources consumed by the process.
- **number_of_produced_sources** (*int*) – total number of event sources produced by the process.

Returns True if either number of event sources has increased.

Return type bool

Raises ValueError – if the consumed or produced number of event sources is smaller than the value of the previous update.

UpdateNumberOfEventTags(*number_of_consumed_event_tags*, *number_of_produced_event_tags*)
Updates the number of event tags.

Parameters

- **number_of_consumed_event_tags** (*int*) – total number of event tags consumed by the process.
- **number_of_produced_event_tags** (*int*) – total number of event tags produced by the process.

Returns True if either number of event tags has increased.

Return type bool

Raises ValueError – if the consumed or produced number of event tags is smaller than the value of the previous update.

UpdateNumberOfEvents(*number_of_consumed_events*, *number_of_produced_events*)
Updates the number of events.

Parameters

- **number_of_consumed_events** (*int*) – total number of events consumed by the process.
- **number_of_produced_events** (*int*) – total number of events produced by the process.

Returns True if either number of events has increased.

Return type bool

Raises ValueError – if the consumed or produced number of events is smaller than the value of the previous update.

class plaso.engine.processing_status.**ProcessingStatus**
Bases: object

The status of the overall extraction process (processing).

aborted
bool – True if processing was aborted.

error_path_specs
list[dfvfs.PathSpec] – path specifications that caused critical errors during processing.

foreman_status
ProcessingStatus – foreman processing status.

tasks_status
TasksStatus – status information about tasks.

UpdateForemanStatus(*identifier*, *status*, *pid*, *used_memory*, *display_name*, *number_of_consumed_sources*, *number_of_produced_sources*, *number_of_consumed_events*, *number_of_produced_events*, *number_of_consumed_event_tags*, *number_of_produced_event_tags*, *number_of_consumed_errors*, *number_of_produced_errors*, *number_of_consumed_reports*, *number_of_produced_reports*)
Updates the status of the foreman.

Parameters

- **identifier** (*str*) – foreman identifier.
- **status** (*str*) – human readable status of the foreman e.g. 'Idle'.
- **pid** (*int*) – process identifier (PID).
- **used_memory** (*int*) – size of used memory in bytes.
- **display_name** (*str*) – human readable of the file entry currently being processed by the foreman.
- **number_of_consumed_sources** (*int*) – total number of event sources consumed by the foreman.
- **number_of_produced_sources** (*int*) – total number of event sources produced by the foreman.
- **number_of_consumed_events** (*int*) – total number of events consumed by the foreman.
- **number_of_produced_events** (*int*) – total number of events produced by the foreman.
- **number_of_consumed_event_tags** (*int*) – total number of event tags consumed by the foreman.
- **number_of_produced_event_tags** (*int*) – total number of event tags produced by the foreman.
- **number_of_consumed_errors** (*int*) – total number of errors consumed by the foreman.
- **number_of_produced_errors** (*int*) – total number of errors produced by the foreman.
- **number_of_consumed_reports** (*int*) – total number of event reports consumed by the process.

- **number_of_produced_reports** (*int*) – total number of event reports produced by the process.

UpdateTasksStatus(*tasks_status*)
Updates the tasks status.

Parameters **tasks_status** (TasksStatus) – status information about tasks.

UpdateWorkerStatus(*identifier*, *status*, *pid*, *used_memory*, *display_name*, *number_of_consumed_sources*, *number_of_produced_sources*, *number_of_consumed_events*, *number_of_produced_events*, *number_of_consumed_event_tags*, *number_of_produced_event_tags*, *number_of_consumed_errors*, *number_of_produced_errors*, *number_of_consumed_reports*, *number_of_produced_reports*)
Updates the status of a worker.

Parameters

- **identifier** (*str*) – worker identifier.
- **status** (*str*) – human readable status of the worker e.g. 'Idle'.
- **pid** (*int*) – process identifier (PID).
- **used_memory** (*int*) – size of used memory in bytes.
- **display_name** (*str*) – human readable of the file entry currently being processed by the worker.
- **number_of_consumed_sources** (*int*) – total number of event sources consumed by the worker.
- **number_of_produced_sources** (*int*) – total number of event sources produced by the worker.
- **number_of_consumed_events** (*int*) – total number of events consumed by the worker.
- **number_of_produced_events** (*int*) – total number of events produced by the worker.
- **number_of_consumed_event_tags** (*int*) – total number of event tags consumed by the worker.
- **number_of_produced_event_tags** (*int*) – total number of event tags produced by the worker.
- **number_of_consumed_errors** (*int*) – total number of errors consumed by the worker.
- **number_of_produced_errors** (*int*) – total number of errors produced by the worker.
- **number_of_consumed_reports** (*int*) – total number of event reports consumed by the process.
- **number_of_produced_reports** (*int*) – total number of event reports produced by the process.

workers_status
The worker status objects sorted by identifier.

class plaso.engine.processing_status.**TasksStatus**
Bases: object

The status of the tasks.

number_of_abandoned_tasks
int – number of abandoned tasks.

number_of_queued_tasks
int – number of active tasks.

number_of_tasks_pending_merge
int – number of tasks pending merge.

number_of_tasks_processing
int – number of tasks processing.

total_number_of_tasks
int – total number of tasks.

plaso.engine.profiler module

The profiler classes.

class plaso.engine.profiler.**BaseMemoryProfiler**(*identifier*, *path=None*, *profiling_sample_rate=1000*)

Bases: object

The memory profiler interface.

classmethod IsSupported()
Determines if the profiler is supported.

Returns True if the profiler is supported.

Return type bool

Sample()
Takes a sample for profiling.

Start()
Starts the profiler.

Stop()
Stops the profiler.

class plaso.engine.profiler.**CPUTimeMeasurements**

Bases: object

The CPU time measurements.

number_of_samples
int – number of samples.

total_cpu_time
int – total CPU time measured by the samples.

total_system_time
int – total system time measured by the samples.

SampleStart()
Starts measuring the CPU and system time.

SampleStop()
Stops the current measurement and adds the sample.

class plaso.engine.profiler.**CPUTimeProfiler**(*identifier*, *path=None*)
Bases: object

The CPU time profiler.

StartTiming(*profile_name*)
Starts timing CPU time.

Parameters **profile_name** (*str*) – name of the profile to sample.

StopTiming(*profile_name*)
Stops timing CPU time.

Parameters **profile_name** (*str*) – name of the profile to sample.

Write()
Writes the CPU time measurements to a sample file.

class plaso.engine.profiler.**GuppyMemoryProfiler**(*identifier*, *path=None*, *profiling_sample_rate=1000*)
Bases: *plaso.engine.profiler.BaseMemoryProfiler*

The guppy-based memory profiler.

classmethod IsSupported()
Determines if the profiler is supported.

Returns True if the profiler is supported.

Return type bool

Start()
Starts the profiler.

Stop()
Stops the profiler.

class plaso.engine.profiler.**ParsersProfiler**(*identifier*, *path=None*)
Bases: *plaso.engine.profiler.CPUTimeProfiler*

The parsers profiler.

class plaso.engine.profiler.**ProcessingProfiler**(*identifier*, *path=None*)
Bases: *plaso.engine.profiler.CPUTimeProfiler*

The processing profiler.

class plaso.engine.profiler.**SerializersProfiler**(*identifier*, *path=None*)
Bases: *plaso.engine.profiler.CPUTimeProfiler*

The serializers profiler.

plaso.engine.single_process module

plaso.engine.worker module

plaso.engine.zeromq_queue module

ZeroMQ implementations of the Plaso queue interface.

class plaso.engine.zeromq_queue.**ZeroMQBufferedQueue**(*buffer_timeout_seconds=2*, *buffer_max_size=10000*, *delay_open=True*, *linger_seconds=10*, *maximum_items=1000*, *name=u'Unnamed'*, *port=None*, *timeout_seconds=5*)

Bases: *plaso.engine.zeromq_queue.ZeroMQQueue*

Parent class for buffered Plaso queues.

Buffered queues use a regular Python queue to store items that are pushed or popped from the queue without blocking on underlying ZeroMQ operations.

This class should not be instantiated directly, a subclass should be instantiated instead.

Close(*abort=False*)

Closes the queue.

Parameters **abort** (*Optional[bool]*) – whether the Close is the result of an abort condition. If True, queue contents may be lost.

Raises

- QueueAlreadyClosed – If the queue is not started, or has already been closed.
- RuntimeError – if closed or terminate event is missing.

Empty()

Removes all items from the internal buffer.

class plaso.engine.zeromq_queue.**ZeroMQBufferedReplyBindQueue**(*buffer_timeout_seconds=2*, *buffer_max_size=10000*, *delay_open=True*, *linger_seconds=10*, *maximum_items=1000*, *name=u'Unnamed'*, *port=None*, *timeout_seconds=5*)

Bases: *plaso.engine.zeromq_queue.ZeroMQBufferedReplyQueue*

A Plaso queue backed by a ZeroMQ REP socket that binds to a port.

This queue may only be used to pop items, not to push.

SOCKET_CONNECTION_TYPE = 1

class plaso.engine.zeromq_queue.**ZeroMQBufferedReplyQueue**(*buffer_timeout_seconds=2*, *buffer_max_size=10000*, *delay_open=True*, *linger_seconds=10*, *maximum_items=1000*, *name=u'Unnamed'*, *port=None*, *timeout_seconds=5*)

Bases: *plaso.engine.zeromq_queue.ZeroMQBufferedQueue*

Parent class for buffered Plaso queues backed by ZeroMQ REP sockets.

This class should not be instantiated directly, a subclass should be instantiated instead.

Instances of this class or subclasses may only be used to push items, not to pop.

PopItem()
Pops an item of the queue.

Provided for compatibility with the API, but doesn't actually work.

Raises WrongQueueType – As Pop is not supported by this queue.

PushItem(*item*, *block=True*)
Push an item on to the queue.

If no ZeroMQ socket has been created, one will be created the first time this method is called.

Parameters

- **item** (*object*) – item to push on the queue.
- **block** (*Optional[bool]*) – whether the push should be performed in blocking or non-block mode.

Raises

- QueueAlreadyClosed – If the queue is closed.
- QueueFull – If the internal buffer was full and it was not possible to push the item to the buffer within the timeout.
- RuntimeError – if closed event is missing.

class plaso.engine.zeromq_queue.**ZeroMQPullConnectQueue**(*delay_open=True*, *linger_seconds=10*, *maximum_items=1000*, *name=u'Unnamed'*, *port=None*, *timeout_seconds=5*)
Bases: *plaso.engine.zeromq_queue.ZeroMQPullQueue*

A Plaso queue backed by a ZeroMQ PULL socket that connects to a port.

This queue may only be used to pop items, not to push.

SOCKET_CONNECTION_TYPE = 2

class plaso.engine.zeromq_queue.**ZeroMQPullQueue**(*delay_open=True*, *linger_seconds=10*, *maximum_items=1000*, *name=u'Unnamed'*, *port=None*, *timeout_seconds=5*)
Bases: *plaso.engine.zeromq_queue.ZeroMQQueue*

Parent class for Plaso queues backed by ZeroMQ PULL sockets.

This class should not be instantiated directly, a subclass should be instantiated instead.

Instances of this class or subclasses may only be used to pop items, not to push.

PopItem()
Pops an item off the queue.

If no ZeroMQ socket has been created, one will be created the first time this method is called.

Returns item from the queue.

Return type object

Raises

- QueueEmpty – If the queue is empty, and no item could be popped within the queue timeout.

- `RuntimeError` – if closed or terminate event is missing.
- `zmq.error.ZMQError` – If a ZeroMQ error occurs.

PushItem(*item*, *block=True*)
Pushes an item on to the queue.

Provided for compatibility with the API, but doesn't actually work.

Parameters

- **item** (*object*) – item to push on the queue.
- **block** (*Optional[bool]*) – whether the push should be performed in blocking or non-block mode.

Raises `WrongQueueType` – As Push is not supported this queue.

class plaso.engine.zeromq_queue.**ZeroMQPushBindQueue**(*delay_open=True*, *linger_seconds=10*, *maximum_items=1000*, *name=u'Unnamed'*, *port=None*, *timeout_seconds=5*)
Bases: *plaso.engine.zeromq_queue.ZeroMQPushQueue*

A Plaso queue backed by a ZeroMQ PUSH socket that binds to a port.

This queue may only be used to push items, not to pop.

SOCKET_CONNECTION_TYPE = 1

class plaso.engine.zeromq_queue.**ZeroMQPushQueue**(*delay_open=True*, *linger_seconds=10*, *maximum_items=1000*, *name=u'Unnamed'*, *port=None*, *timeout_seconds=5*)
Bases: *plaso.engine.zeromq_queue.ZeroMQQueue*

Parent class for Plaso queues backed by ZeroMQ PUSH sockets.

This class should not be instantiated directly, a subclass should be instantiated instead.

Instances of this class or subclasses may only be used to push items, not to pop.

PopItem()
Pops an item of the queue.

Provided for compatibility with the API, but doesn't actually work.

Raises `WrongQueueType` – As Pull is not supported this queue.

PushItem(*item*, *block=True*)
Push an item on to the queue.

If no ZeroMQ socket has been created, one will be created the first time this method is called.

Parameters

- **item** (*object*) – item to push on the queue.
- **block** (*Optional[bool]*) – whether the push should be performed in blocking or non-block mode.

Raises

- `KeyboardInterrupt` – if the process is sent a KeyboardInterrupt while pushing an item.

- `QueueFull` – if it was not possible to push the item to the queue within the timeout.
- `RuntimeError` – if terminate event is missing.
- `zmq.error.ZMQError` – if a ZeroMQ specific error occurs.

class plaso.engine.zeromq_queue.**ZeroMQQueue**(*delay_open=True*, *linger_seconds=10*, *maximum_items=1000*, *name=u'Unnamed'*, *port=None*, *timeout_seconds=5*)

Bases: *plaso.engine.plaso_queue.Queue*

Interface for a ZeroMQ backed queue.

name
str – name to identify the queue.

port
int – TCP port that the queue is connected or bound to. If the queue is not yet bound or connected to a port, this value will be None.

timeout_seconds
int – number of seconds that calls to PopItem and PushItem may block for, before returning queue.QueueEmpty.

Close(*abort=False*)
Closes the queue.

Parameters **abort** (*Optional[bool]*) – whether the Close is the result of an abort condition. If True, queue contents may be lost.

Raises

- `QueueAlreadyClosed` – If the queue is not started, or has already been closed.
- `RuntimeError` – if closed or terminate event is missing.

IsBound()
Checks if the queue is bound to a port.

IsConnected()
Checks if the queue is connected to a port.

IsEmpty()
Checks if the queue is empty.

ZeroMQ queues don't have a concept of "empty" - there could always be messages on the queue that a producer or consumer is unaware of. Thus, the queue is never empty, so we return False. Note that it is possible that a queue is unable to pop an item from a queue within a timeout, which will cause PopItem to raise a QueueEmpty exception, but this is a different condition.

Returns False, to indicate the the queue isn't empty.

Return type bool

Open()
Opens this queue, causing the creation of a ZeroMQ socket.

Raises `QueueAlreadyStarted` – If the queue is already started, and a socket already exists.

PopItem()
Pops an item off the queue.

Returns item from the queue.

Return type object

Raises `QueueEmpty` – If the queue is empty, and no item could be popped within the queue timeout.

PushItem(*item*, *block=True*)
Pushes an item on to the queue.

Parameters

- **item** (*object*) – item to push on the queue.
- **block** (*Optional[bool]*) – whether the push should be performed in blocking or non-block mode.

Raises `QueueAlreadyClosed` – If the queue is closed.

SOCKET_CONNECTION_BIND = 1

SOCKET_CONNECTION_CONNECT = 2

SOCKET_CONNECTION_TYPE = None

class plaso.engine.zeromq_queue.**ZeroMQRequestConnectQueue**(*delay_open=True*, *linger_seconds=10*, *maximum_items=1000*, *name=u'Unnamed'*, *port=None*, *timeout_seconds=5*)
Bases: *plaso.engine.zeromq_queue.ZeroMQRequestQueue*

A Plaso queue backed by a ZeroMQ REQ socket that connects to a port.

This queue may only be used to pop items, not to push.

SOCKET_CONNECTION_TYPE = 2

class plaso.engine.zeromq_queue.**ZeroMQRequestQueue**(*delay_open=True*, *linger_seconds=10*, *maximum_items=1000*, *name=u'Unnamed'*, *port=None*, *timeout_seconds=5*)
Bases: *plaso.engine.zeromq_queue.ZeroMQQueue*

Parent class for Plaso queues backed by ZeroMQ REQ sockets.

This class should not be instantiated directly, a subclass should be instantiated instead.

Instances of this class or subclasses may only be used to pop items, not to push.

PopItem()
Pops an item off the queue.

If no ZeroMQ socket has been created, one will be created the first time this method is called.

Returns item from the queue.

Return type object

Raises

- `KeyboardInterrupt` – if the process is sent a KeyboardInterrupt while popping an item.
- `QueueEmpty` – if the queue is empty, and no item could be popped within the queue timeout.
- `RuntimeError` – if terminate event is missing.

- zmq.error.ZMQError – if an error occurs in ZeroMQ.

PushItem(*item*, *block=True*)
Pushes an item on to the queue.

Provided for compatibility with the API, but doesn't actually work.

Parameters

- **item** (*object*) – item to push on the queue.
- **block** (*Optional[bool]*) – whether the push should be performed in blocking or non-block mode.

Raises WrongQueueType – As Push is not supported this queue.

Module contents

plaso.filters package

Submodules

plaso.filters.dynamic_filter module

plaso.filters.event_filter module

plaso.filters.file_entry module

plaso.filters.filter_list module

plaso.filters.interface module

plaso.filters.manager module

plaso.filters.path_filter module

Module contents

plaso.formatters package

Submodules

plaso.formatters.amcache module

The Windows Registry Amcache entries event formatter.

class plaso.formatters.amcache.**AmcacheFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an Amcache Windows Registry event.

DATA_TYPE = u'windows:registry:amcache'

FORMAT_STRING_PIECES = [u'path: {full_path}', u'sha1: {sha1}', u'productname: {prod

FORMAT_STRING_SHORT_PIECES = [u'path: {full_path}']

SOURCE_LONG = u'Amcache Registry Entry'

SOURCE_SHORT = u'AMCACHE'

class plaso.formatters.amcache.**AmcacheProgramsFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an Amcache Programs Windows Registry event.

DATA_TYPE = u'windows:registry:amcache:programs'

FORMAT_STRING_PIECES = [u'name: {name}', u'version: {version}', u'publisher: {publi

FORMAT_STRING_SHORT_PIECES = [u'name: {name}']

SOURCE_LONG = u'Amcache Programs Registry Entry'

SOURCE_SHORT = u'AMCACHEPROGRAM'

plaso.formatters.android_app_usage module

The Android Application Usage event formatter.

class plaso.formatters.android_app_usage.**AndroidApplicationFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an Application Last Resumed event.

DATA_TYPE = u'android:event:last_resume_time'

FORMAT_STRING_PIECES = [u'Package: {package}', u'Component: {component}']

SOURCE_LONG = u'Android App Usage'

SOURCE_SHORT = u'LOG'

plaso.formatters.android_calls module

The Android contacts2.db database event formatter.

class plaso.formatters.android_calls.**AndroidCallFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an Android call history event.

DATA_TYPE = u'android:event:call'

FORMAT_STRING_PIECES = [u'{call_type}', u'Number: {number}', u'Name: {name}', u'Dura

FORMAT_STRING_SHORT_PIECES = [u'{call_type} Call']

SOURCE_LONG = u'Android Call History'

SOURCE_SHORT = u'LOG'

plaso.formatters.android_sms module

The Android mmssms.db database event formatter.

class plaso.formatters.android_sms.**AndroidSmsFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an Android SMS event.

DATA_TYPE = u'android:messaging:sms'

FORMAT_STRING_PIECES = [u'Type: {sms_type}', u'Address: {address}', u'Status: {sms_

FORMAT_STRING_SHORT_PIECES = [u'{body}']

SOURCE_LONG = u'Android SMS messages'

SOURCE_SHORT = u'SMS'

plaso.formatters.android_webview module

The Android WebView database event formatter.

class plaso.formatters.android_webview.**AndroidWebViewCookieEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for Android WebView Cookie event data.

DATA_TYPE = u'webview:cookie'

FORMAT_STRING_PIECES = [u'Domain: {domain}', u'Path: {path}', u'Cookie name: {name}

FORMAT_STRING_SHORT_PIECES = [u'{domain}', u'{name}', u'{value}']

SOURCE_LONG = u'Android WebView'

SOURCE_SHORT = u'WebView'

plaso.formatters.android_webviewcache module

The Android WebViewCache database event formatter.

class plaso.formatters.android_webviewcache.**AndroidWebViewCacheFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for Android WebViewCache event data.

DATA_TYPE = u'android:webviewcache'

FORMAT_STRING_PIECES = [u'URL: {url}', u'Content Length: {content_length}']

FORMAT_STRING_SHORT_PIECES = [u'{url}']

SOURCE_LONG = u'Android WebViewCache'

SOURCE_SHORT = u'WebViewCache'

plaso.formatters.appcompatcache module

The Windows Registry AppCompatCache entries event formatter.

class plaso.formatters.appcompatcache.**AppCompatCacheFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an AppCompatCache Windows Registry event.

DATA_TYPE = u'windows:registry:appcompatcache'

FORMAT_STRING_PIECES = [u'[{key_path}]', u'Cached entry: {entry_index}', u'Path: {pa

FORMAT_STRING_SHORT_PIECES = [u'Path: {path}']

SOURCE_LONG = u'AppCompatCache Registry Entry'

SOURCE_SHORT = u'REG'

plaso.formatters.appusage module

The MacOS application usage event formatter.

class plaso.formatters.appusage.**ApplicationUsageFormatter**
Bases: *plaso.formatters.interface.EventFormatter*

Formatter for a MacOS Application usage event.

DATA_TYPE = u'macosx:application_usage'

FORMAT_STRING = u'{application} v.{app_version} (bundle: {bundle_id}). Launched: {co

FORMAT_STRING_SHORT = u'{application} ({count} time(s))'

SOURCE_LONG = u'Application Usage'

SOURCE_SHORT = u'LOG'

plaso.formatters.asl module

The Apple System Log (ASL) event formatter.

class plaso.formatters.asl.**ASLFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an Apple System Log (ASL) log event.

DATA_TYPE = u'mac:asl:event'

FORMAT_STRING_PIECES = [u'MessageID: {message_id}', u'Level: {level}', u'User ID: {us

FORMAT_STRING_SHORT_PIECES = [u'Host: {host}', u'Sender: {sender}', u'Facility: {fa

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'ASL entry'

SOURCE_SHORT = u'LOG'

plaso.formatters.bash_history module

The Bash history event formatter.

class plaso.formatters.bash_history.**BashHistoryEventFormatter**
Bases: *plaso.formatters.interface.EventFormatter*

Formatter for Bash history events.

DATA_TYPE = u'bash:history:command'

FORMAT_STRING = u'Command executed: {command}'

FORMAT_STRING_SHORT = u'{command}'

SOURCE_LONG = u'Bash History'

SOURCE_SHORT = u'LOG'

plaso.formatters.bencode_parser module

The bencode parser event formatters.

class plaso.formatters.bencode_parser.**TransmissionEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Transmission active torrents event.

DATA_TYPE = u'p2p:bittorrent:transmission'

FORMAT_STRING_PIECES = [u'Saved to {destination}', u'Minutes seeded: {seedtime}']

FORMAT_STRING_SEPARATOR = u'; '

SOURCE_LONG = u'Transmission Active Torrents'

SOURCE_SHORT = u'TORRENT'

class plaso.formatters.bencode_parser.**UTorrentEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a BitTorrent uTorrent active torrents event.

DATA_TYPE = u'p2p:bittorrent:utorrent'

FORMAT_STRING_PIECES = [u'Torrent {caption}', u'Saved to {path}', u'Minutes seeded: {

FORMAT_STRING_SEPARATOR = u'; '

SOURCE_LONG = u'uTorrent Active Torrents'

SOURCE_SHORT = u'TORRENT'

plaso.formatters.bsm module

The Basic Security Module (BSM) binary files event formatter.

class plaso.formatters.bsm.**BSMFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a BSM log entry.

DATA_TYPE = u'bsm:event'

FORMAT_STRING_PIECES = [u'Type: {event_type}', u'Return: {return_value}', u'Informat

FORMAT_STRING_SHORT_PIECES = [u'Type: {event_type}', u'Return: {return_value}']

SOURCE_LONG = u'BSM entry'

SOURCE_SHORT = u'LOG'

plaso.formatters.ccleaner module

The CCleaner event formatter.

class plaso.formatters.ccleaner.**CCleanerUpdateEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a CCleaner update event.

DATA_TYPE = u'ccleaner:update'

FORMAT_STRING_PIECES = [u'Origin: {key_path}']

FORMAT_STRING_SHORT_PIECES = [u'Origin: {key_path}']

SOURCE_LONG = u'System'

SOURCE_SHORT = u'LOG'

plaso.formatters.chrome module

The Google Chrome history event formatters.

class plaso.formatters.chrome.**ChromeFileDownloadFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Chrome file download event.

DATA_TYPE = u'chrome:history:file_downloaded'

FORMAT_STRING_PIECES = [u'{url}', u'({full_path}).', u'Received: {received_bytes} byt

FORMAT_STRING_SHORT_PIECES = [u'{full_path} downloaded', u'({received_bytes} bytes)']

SOURCE_LONG = u'Chrome History'

SOURCE_SHORT = u'WEBHIST'

class plaso.formatters.chrome.**ChromePageVisitedFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Chrome page visited event.

DATA_TYPE = u'chrome:history:page_visited'

FORMAT_STRING_PIECES = [u'{url}', u'({title})', u'[count: {typed_count}]', u'Visit fr

FORMAT_STRING_SHORT_PIECES = [u'{url}', u'({title})']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Chrome History'

SOURCE_SHORT = u'WEBHIST'

plaso.formatters.chrome_cache module

The Google Chrome Cache files event formatter.

class plaso.formatters.chrome_cache.**ChromeCacheEntryEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Chrome Cache entry event.

DATA_TYPE = u'chrome:cache:entry'

FORMAT_STRING_PIECES = [u'Original URL: {original_url}']

SOURCE_LONG = u'Chrome Cache'

SOURCE_SHORT = u'WEBHIST'

plaso.formatters.chrome_cookies module

The Google Chrome cookies database event formatter.

class plaso.formatters.chrome_cookies.**ChromeCookieFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Chrome cookie event.

DATA_TYPE = u'chrome:cookie:entry'

FORMAT_STRING_PIECES = [u'{url}', u'({cookie_name})', u'Flags:', u'[HTTP only] = {htt

FORMAT_STRING_SHORT_PIECES = [u'{host}', u'({cookie_name})']

SOURCE_LONG = u'Chrome Cookies'

SOURCE_SHORT = u'WEBHIST'

plaso.formatters.chrome_extension_activity module

The Google Chrome extension activity database event formatter.

class plaso.formatters.chrome_extension_activity.**ChromeExtensionActivityEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Chrome extension activity event.

DATA_TYPE = u'chrome:extension_activity:activity_log'

FORMAT_STRING_PIECES = [u'Chrome extension: {extension_id}', u'Action type: {action_

FORMAT_STRING_SHORT_PIECES = [u'{extension_id}', u'{api_name}', u'{args}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Chrome Extension Activity'

SOURCE_SHORT = u'WEBHIST'

plaso.formatters.chrome_preferences module

The Google Chrome Preferences file event formatter.

class plaso.formatters.chrome_preferences.**ChromeContentSettingsExceptionsFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Chrome content_settings exceptions event.

DATA_TYPE = u'chrome:preferences:content_settings:exceptions'

FORMAT_STRING_PIECES = [u'Permission {permission}', u'used by {subject}']

FORMAT_STRING_SHORT_PIECES = [u'Permission {permission}', u'used by {subject}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Chrome Permission Event'

SOURCE_SHORT = u'LOG'

class plaso.formatters.chrome_preferences.**ChromeExtensionInstallationEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Chrome extension installation event.

DATA_TYPE = u'chrome:preferences:extension_installation'

FORMAT_STRING_PIECES = [u'CRX ID: {extension_id}', u'CRX Name: {extension_name}', u'P

FORMAT_STRING_SHORT_PIECES = [u'{extension_id}', u'{path}']

SOURCE_LONG = u'Chrome Extension Installation'

SOURCE_SHORT = u'LOG'

class plaso.formatters.chrome_preferences.**ChromeExtensionsAutoupdaterEvent**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for Chrome Extensions Autoupdater events.

DATA_TYPE = u'chrome:preferences:extensions_autoupdater'

FORMAT_STRING_PIECES = [u'{message}']

FORMAT_STRING_SHORT_PIECES = [u'{message}']

SOURCE_LONG = u'Chrome Extensions Autoupdater'

SOURCE_SHORT = u'LOG'

class plaso.formatters.chrome_preferences.**ChromePreferencesClearHistoryEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for Chrome history clearing events.

DATA_TYPE = u'chrome:preferences:clear_history'

FORMAT_STRING_PIECES = [u'{message}']

FORMAT_STRING_SHORT_PIECES = [u'{message}']

SOURCE_LONG = u'Chrome History Deletion'

SOURCE_SHORT = u'LOG'

plaso.formatters.cron module

The syslog cron formatters.

class plaso.formatters.cron.**CronTaskRunEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a syslog cron task run event.

DATA_TYPE = u'syslog:cron:task_run'

FORMAT_STRING_PIECES = [u'Cron ran: {command}', u'for user: {username}', u'pid: {pi

FORMAT_STRING_SEPARATOR = u' '

FORMAT_STRING_SHORT = u'{body}'

SOURCE_LONG = u'Cron log'

SOURCE_SHORT = u'LOG'

plaso.formatters.cups_ipp module

The CUPS IPP file event formatter.

class plaso.formatters.cups_ipp.**CupsIppFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a CUPS IPP event.

DATA_TYPE = u'cups:ipp:event'

FORMAT_STRING_PIECES = [u'Status: {status}', u'User: {user}', u'Owner: {owner}', u'

FORMAT_STRING_SHORT_PIECES = [u'Status: {status}', u'Job Name: {job_name}']

SOURCE_LONG = u'CUPS IPP Log'

SOURCE_SHORT = u'LOG'

plaso.formatters.default module

The default event formatter.

class plaso.formatters.default.**DefaultFormatter**
Bases: *plaso.formatters.interface.EventFormatter*

Formatter for events that do not have any defined formatter.

DATA_TYPE = u'event'

FORMAT_STRING = u'<WARNING DEFAULT FORMATTER> Attributes: {attribute_driven}'

FORMAT_STRING_SHORT = u'<DEFAULT> {attribute_driven}'

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

plaso.formatters.docker module

The Docker event formatter.

class plaso.formatters.docker.**DockerBaseEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Class that contains common Docker event formatter functionality.

DATA_TYPE = u'docker:json'

FORMAT_STRING_SHORT_PIECES = [u'{id}']

SOURCE_SHORT = u'DOCKER'

class plaso.formatters.docker.**DockerContainerEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Docker event.

DATA_TYPE = u'docker:json:container'

FORMAT_STRING_PIECES = [u'Action: {action}', u'Container Name: {container_name}', u'

FORMAT_STRING_SEPARATOR = u', '

SOURCE_LONG = u'Docker Container'

SOURCE_SHORT = u'DOCKER'

class plaso.formatters.docker.**DockerContainerLogEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Docker container log event

DATA_TYPE = u'docker:json:container:log'

FORMAT_STRING_PIECES = (u'Text: {log_line}', u'Container ID: {container_id}', u'Source

FORMAT_STRING_SEPARATOR = u', '

SOURCE_LONG = u'Docker Container Logs'

SOURCE_SHORT = u'DOCKER'

class plaso.formatters.docker.**DockerLayerEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Docker layer event.

DATA_TYPE = u'docker:json:layer'

FORMAT_STRING_PIECES = (u'Command: {command}', u'Layer ID: {layer_id}')

FORMAT_STRING_SEPARATOR = u', '

SOURCE_LONG = u'Docker Layer'

SOURCE_SHORT = u'DOCKER'

plaso.formatters.dpkg module

The dpkg.log event formatter.

class plaso.formatters.dpkg.**DpkgFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a dpkg log file event.

DATA_TYPE = u'dpkg:line'

FORMAT_STRING_PIECES = [u'{body}']

FORMAT_STRING_SEPARATOR = u''

SOURCE_LONG = u'dpkg log File'

SOURCE_SHORT = u'LOG'

plaso.formatters.file_history module

The file history ESE database event formatter.

class plaso.formatters.file_history.**FileHistoryNamespaceEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a file history ESE database namespace table record.

DATA_TYPE = u'file_history:namespace:event'

FORMAT_STRING_PIECES = [u'Filename: {original_filename}', u'Identifier: {identifier}

FORMAT_STRING_SHORT_PIECES = [u'Filename: {original_filename}']

SOURCE_LONG = u'File History Namespace'

SOURCE_SHORT = u'LOG'

plaso.formatters.file_system module

The file system stat event formatter.

class plaso.formatters.file_system.**FileStatEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

The file system stat event formatter.

DATA_TYPE = u'fs:stat'

FORMAT_STRING_PIECES = [u'{display_name}', u'Type: {file_entry_type}', u'({unallocate

FORMAT_STRING_SHORT_PIECES = [u'{filename}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

GetSources(*event*)
Determines the the short and long source for an event object.

Parameters **event** (EventObject) – event.

Returns short and long source string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_SHORT = u'FILE'

class plaso.formatters.file_system.**NTFSFileStatEventFormatter**
Bases: *plaso.formatters.file_system.FileStatEventFormatter*

The NTFS file system stat event formatter.

DATA_TYPE = u'fs:stat:ntfs'

FORMAT_STRING_PIECES = [u'{display_name}', u'File reference: {file_reference}', u'Att

FORMAT_STRING_SHORT_PIECES = [u'{filename}', u'{file_reference}', u'{attribute_name}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_SHORT = u'FILE'

class plaso.formatters.file_system.**NTFSUSNChangeEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

The NTFS USN change event formatter.

DATA_TYPE = u'fs:ntfs:usn_change'

FORMAT_STRING_PIECES = [u'{filename}', u'File reference: {file_reference}', u'Parent

FORMAT_STRING_SHORT_PIECES = [u'{filename}', u'{file_reference}', u'{update_reason}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_SHORT = u'FILE'

plaso.formatters.firefox module

The Mozilla Firefox history event formatter.

class plaso.formatters.firefox.**FirefoxBookmarkAnnotationFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

The Firefox bookmark annotation event formatter.

DATA_TYPE = u'firefox:places:bookmark_annotation'

FORMAT_STRING_PIECES = [u'Bookmark Annotation: [{content}]', u'to bookmark [{title}]'

FORMAT_STRING_SHORT_PIECES = [u'Bookmark Annotation: {title}']

SOURCE_LONG = u'Firefox History'

SOURCE_SHORT = u'WEBHIST'

class plaso.formatters.firefox.**FirefoxBookmarkFolderFormatter**
Bases: *plaso.formatters.interface.EventFormatter*

The Firefox bookmark folder event formatter.

DATA_TYPE = u'firefox:places:bookmark_folder'

FORMAT_STRING = u'{title}'

SOURCE_LONG = u'Firefox History'

SOURCE_SHORT = u'WEBHIST'

class plaso.formatters.firefox.**FirefoxBookmarkFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

The Firefox URL bookmark event formatter.

DATA_TYPE = u'firefox:places:bookmark'

FORMAT_STRING_PIECES = [u'Bookmark {type}', u'{title}', u'({url})', u'[{places_title}]

FORMAT_STRING_SHORT_PIECES = [u'Bookmarked {title}', u'({url})']

SOURCE_LONG = u'Firefox History'

SOURCE_SHORT = u'WEBHIST'

class plaso.formatters.firefox.**FirefoxDowloadFormatter**
Bases: *plaso.formatters.interface.EventFormatter*

The Firefox download event formatter.

DATA_TYPE = u'firefox:downloads:download'

FORMAT_STRING = u'{url} ({full_path}). Received: {received_bytes} bytes out of: {tota

FORMAT_STRING_SHORT = u'{full_path} downloaded ({received_bytes} bytes)'

SOURCE_LONG = u'Firefox History'

SOURCE_SHORT = u'WEBHIST'

class plaso.formatters.firefox.**FirefoxPageVisitFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

The Firefox page visited event formatter.

DATA_TYPE = u'firefox:places:page_visited'

FORMAT_STRING_PIECES = [u'{url}', u'({title})', u'[count: {visit_count}]', u'Host: {

FORMAT_STRING_SHORT_PIECES = [u'URL: {url}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Firefox History'

SOURCE_SHORT = u'WEBHIST'

plaso.formatters.firefox_cache module

The Firefox cache record event formatter.

class plaso.formatters.firefox_cache.**FirefoxCacheFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

The Firefox cache record event formatter.

DATA_TYPE = u'firefox:cache:record'

FORMAT_STRING_PIECES = [u'Fetched {fetch_count} time(s)', u'[{response_code}]', u'{req

FORMAT_STRING_SHORT_PIECES = [u'[{response_code}]', u'{request_method}', u'"{url}"']

SOURCE_LONG = u'Firefox Cache'

SOURCE_SHORT = u'WEBHIST'

plaso.formatters.firefox_cookies module

The Firefox cookie entry event formatter.

class plaso.formatters.firefox_cookies.**FirefoxCookieFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

The Firefox cookie entry event formatter.

DATA_TYPE = u'firefox:cookie:entry'

FORMAT_STRING_PIECES = [u'{url}', u'({cookie_name})', u'Flags:', u'[HTTP only]: {htt

FORMAT_STRING_SHORT_PIECES = [u'{host}', u'({cookie_name})']

SOURCE_LONG = u'Firefox Cookies'

SOURCE_SHORT = u'WEBHIST'

plaso.formatters.fseventsd module

The fseventsd event formatter.

class plaso.formatters.fseventsd.**FSEventsdEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

The fseventsd event formatter.

DATA_TYPE = u'macos:fseventsd:record'

FORMAT_STRING_PIECES = [u'{path}', u'Flag Values:', u'{flag_values}', u'Flags:', u'{

FORMAT_STRING_SHORT_PIECES = [u'{path}', u'{flag_values}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_SHORT = u'FSEVENT'

plaso.formatters.ganalytics module

The Google Analytics cookie event formatters.

class plaso.formatters.ganalytics.**AnalyticsUtmaCookieFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

The UTMA Google Analytics cookie event formatter.

DATA_TYPE = u'cookie:google:analytics:utma'

FORMAT_STRING_PIECES = [u'{url}', u'({cookie_name})', u'Sessions: {sessions}', u'Doma

FORMAT_STRING_SHORT_PIECES = [u'{url}', u'({cookie_name})']

SOURCE_LONG = u'Google Analytics Cookies'

SOURCE_SHORT = u'WEBHIST'

class plaso.formatters.ganalytics.**AnalyticsUtmbCookieFormatter**
Bases: *plaso.formatters.ganalytics.AnalyticsUtmaCookieFormatter*

The UTMB Google Analytics cookie event formatter.

DATA_TYPE = u'cookie:google:analytics:utmb'

FORMAT_STRING_PIECES = [u'{url}', u'({cookie_name})', u'Pages Viewed: {pages_viewed}'

class plaso.formatters.ganalytics.**AnalyticsUtmtCookieFormatter**
Bases: *plaso.formatters.ganalytics.AnalyticsUtmaCookieFormatter*

The UTMT Google Analytics cookie event formatter.

DATA_TYPE = u'cookie:google:analytics:utmt'

FORMAT_STRING_PIECES = [u'{url}', u'({cookie_name})']

class plaso.formatters.ganalytics.**AnalyticsUtmzCookieFormatter**
Bases: *plaso.formatters.ganalytics.AnalyticsUtmaCookieFormatter*

The UTMZ Google Analytics cookie event formatter.

DATA_TYPE = u'cookie:google:analytics:utmz'

FORMAT_STRING_PIECES = [u'{url}', u'({cookie_name})', u'Sessions: {sessions}', u'Doma

plaso.formatters.gdrive module

The Google Drive snapshots event formatter.

class plaso.formatters.gdrive.**GDriveCloudEntryFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Google Drive snapshot cloud event.

DATA_TYPE = u'gdrive:snapshot:cloud_entry'

FORMAT_STRING_PIECES = [u'File Path: {path}', u'[{shared}]', u'Size: {size}', u'URL:

FORMAT_STRING_SHORT_PIECES = [u'{path}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Google Drive (cloud entry)'

SOURCE_SHORT = u'LOG'

class plaso.formatters.gdrive.**GDriveLocalEntryFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Google Drive snapshot local event.

DATA_TYPE = u'gdrive:snapshot:local_entry'

FORMAT_STRING_PIECES = [u'File Path: {path}', u'Size: {size}']

FORMAT_STRING_SHORT_PIECES = [u'{path}']

SOURCE_LONG = u'Google Drive (local entry)'

SOURCE_SHORT = u'LOG'

plaso.formatters.gdrive_synclog module

Google Drive Sync log event formatter.

class plaso.formatters.gdrive_synclog.**GoogleDriveSyncLogFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Google Drive Sync log file event.

DATA_TYPE = u'gdrive_sync:log:line'

FORMAT_STRING_PIECES = [u'[{log_level}', u'{pid}', u'{thread}', u'{source_code}]', u'{

FORMAT_STRING_SHORT_PIECES = [u'{message}']

SOURCE_LONG = u'GoogleDriveSync Log File'

SOURCE_SHORT = u'LOG'

plaso.formatters.hachoir module

The Hachoir event formatter.

class plaso.formatters.hachoir.**HachoirFormatter**
Bases: *plaso.formatters.interface.EventFormatter*

Formatter for a Hachoir event.

DATA_TYPE = u'metadata:hachoir'

FORMAT_STRING = u'{data}'

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Hachoir Metadata'

SOURCE_SHORT = u'META'

plaso.formatters.iis module

The Microsoft IIS log file event formatter.

class plaso.formatters.iis.**IISLogFileEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Microsoft IIS log file event.

DATA_TYPE = u'iis:log:line'

FORMAT_STRING_PIECES = [u'{http_method}', u'{requested_uri_stem}', u'[', u'{source_ip}

FORMAT_STRING_SHORT_PIECES = [u'{http_method}', u'{requested_uri_stem}', u'[', u'{sour

SOURCE_LONG = u'IIS Log'

SOURCE_SHORT = u'LOG'

plaso.formatters.imessage module

The iMessage chat.db (OSX) and sms.db (iOS)database event formatter.

class plaso.formatters.imessage.**IMessageFormatter**

Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an iMessage and SMS event.

DATA_TYPE = u'imessage:event:chat'

FORMAT_STRING_PIECES = [u'Row ID: {identifier}', u'iMessage ID: {imessage_id}', u'Read

FORMAT_STRING_SHORT_PIECES = [u'{text}']

GetMessages(*unused_formatter_mediator*, *event*)

Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Apple iMessage Application'

SOURCE_SHORT = u'iMessage'

plaso.formatters.interface module

This file contains the event formatters interface classes.

The l2t_csv and other formats are dependent on a message field, referred to as description_long and description_short in l2t_csv.

Plaso no longer stores these field explicitly.

A formatter, with a format string definition, is used to convert the event object values into a formatted string that is similar to the description_long and description_short field.

class plaso.formatters.interface.**ConditionalEventFormatter**

Bases: *plaso.formatters.interface.EventFormatter*

Base class to conditionally format event data using format string pieces.

Define the (long) format string and the short format string by defining FORMAT_STRING_PIECES and FORMAT_STRING_SHORT_PIECES. The syntax of the format strings pieces is similar to of the event formatter (EventFormatter). Every format string piece should contain a single attribute name or none.

FORMAT_STRING_SEPARATOR is used to control the string which the separate string pieces should be joined. It contains a space by default.

FORMAT_STRING_PIECES = [u'']

FORMAT_STRING_SEPARATOR = u' '

FORMAT_STRING_SHORT_PIECES = [u'']

GetFormatStringAttributeNames()
Retrieves the attribute names in the format string.

Returns attribute names.

Return type set(str)

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

class plaso.formatters.interface.**EventFormatter**
Bases: object

Base class to format event type specific data using a format string.

Define the (long) format string and the short format string by defining FORMAT_STRING and FORMAT_STRING_SHORT. The syntax of the format strings is similar to that of format() where the place holder for a certain event object attribute is defined as {attribute_name}.

DATA_TYPE = u'internal'

FORMAT_STRING = u''

FORMAT_STRING_SHORT = u''

GetFormatStringAttributeNames()
Retrieves the attribute names in the format string.

Returns attribute names.

Return type set(str)

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.

- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

GetSources(*event*)
Determines the the short and long source for an event object.

Parameters **event** (EventObject) – event.

Returns short and long source string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u''

SOURCE_SHORT = u'LOG'

plaso.formatters.ipod module

The iPod device event formatter.

class plaso.formatters.ipod.**IPodDeviceFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an iPod device event.

DATA_TYPE = u'ipod:device:entry'

FORMAT_STRING_PIECES = [u'Device ID: {device_id}', u'Type: {device_class}', u'[{famil

SOURCE_LONG = u'iPod Connections'

SOURCE_SHORT = u'LOG'

plaso.formatters.java_idx module

The Java WebStart Cache IDX event formatter.

class plaso.formatters.java_idx.**JavaIDXFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an Java WebStart Cache IDX download event.

DATA_TYPE = u'java:download:idx'

FORMAT_STRING_PIECES = [u'IDX Version: {idx_version}', u'Host IP address: ({ip_addre

SOURCE_LONG = u'Java Cache IDX'

SOURCE_SHORT = u'JAVA_IDX'

plaso.formatters.kik_ios module

The Kik kik.sqlite iOS database event formatter.

class plaso.formatters.kik_ios.**KikIOSMessageFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an iOS Kik message event.

DATA_TYPE = u'ios:kik:messaging'

FORMAT_STRING_PIECES = [u'Username: {username}', u'Displayname: {displayname}', u'St

FORMAT_STRING_SHORT_PIECES = [u'{body}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Kik iOS messages'

SOURCE_SHORT = u'Kik iOS'

plaso.formatters.ls_quarantine module

The MacOS launch services (LS) quarantine event formatter.

class plaso.formatters.ls_quarantine.**LSQuarantineFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a launch services (LS) quarantine history event.

DATA_TYPE = u'macosx:lsquarantine'

FORMAT_STRING_PIECES = [u'[{agent}]', u'Downloaded: {url}', u'<{data}>']

FORMAT_STRING_SHORT_PIECES = [u'{url}']

SOURCE_LONG = u'LS Quarantine Event'

SOURCE_SHORT = u'LOG'

plaso.formatters.mac_appfirewall module

The MacOS appfirewall.log file event formatter.

class plaso.formatters.mac_appfirewall.**MacAppFirewallLogFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for MacOS appfirewall.log file event.

DATA_TYPE = u'mac:appfirewall:line'

FORMAT_STRING_PIECES = [u'Computer: {computer_name}', u'Agent: {agent}', u'Status:

FORMAT_STRING_SHORT_PIECES = [u'Process name: {process_name}', u'Status: {status}']

SOURCE_LONG = u'Mac AppFirewall Log'

SOURCE_SHORT = u'LOG'

plaso.formatters.mac_document_versions module

The MacOS Document Versions files event formatter.

class plaso.formatters.mac_document_versions.**MacDocumentVersionsFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a MacOS Document Versions page visited event.

DATA_TYPE = u'mac:document_versions:file'

FORMAT_STRING_PIECES = [u'Version of [{name}]', u'({path})', u'stored in {version_path

FORMAT_STRING_SHORT_PIECES = [u'Stored a document version of [{name}]']

SOURCE_LONG = u'Document Versions'

SOURCE_SHORT = u'HISTORY'

plaso.formatters.mac_keychain module

The MacOS keychain password database file event formatter.

class plaso.formatters.mac_keychain.**KeychainApplicationRecordFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a keychain application record event.

DATA_TYPE = u'mac:keychain:application'

FORMAT_STRING_PIECES = [u'Name: {entry_name}', u'Account: {account_name}']

FORMAT_STRING_SHORT_PIECES = [u'{entry_name}']

SOURCE_LONG = u'Keychain Application password'

SOURCE_SHORT = u'LOG'

class plaso.formatters.mac_keychain.**KeychainInternetRecordFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a keychain Internet record event.

DATA_TYPE = u'mac:keychain:internet'

FORMAT_STRING_PIECES = [u'Name: {entry_name}', u'Account: {account_name}', u'Where:

FORMAT_STRING_SHORT_PIECES = [u'{entry_name}']

SOURCE_LONG = u'Keychain Internet password'

SOURCE_SHORT = u'LOG'

plaso.formatters.mac_securityd module

The MacOS securityd log file event formatter.

class plaso.formatters.mac_securityd.**MacOSSecuritydLogFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a MacOS securityd log event.

DATA_TYPE = u'mac:securityd:line'

FORMAT_STRING_PIECES = [u'Sender: {sender}', u'({sender_pid})', u'Level: {level}', u

FORMAT_STRING_SHORT_PIECES = [u'Text: {message}']

SOURCE_LONG = u'Mac Securityd Log'

SOURCE_SHORT = u'LOG'

plaso.formatters.mac_wifi module

The MacOS wifi.log file event formatter.

class plaso.formatters.mac_wifi.**MacWifiLogFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a wifi.log file event.

DATA_TYPE = u'mac:wifilog:line'

FORMAT_STRING_PIECES = [u'Action: {action}', u'Agent: {agent}', u'({function})', u'L

FORMAT_STRING_SHORT_PIECES = [u'Action: {action}']

SOURCE_LONG = u'Mac Wifi Log'

SOURCE_SHORT = u'LOG'

plaso.formatters.mackeeper_cache module

The MacKeeper Cache event formatter.

class plaso.formatters.mackeeper_cache.**MacKeeperCacheFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a MacKeeper Cache event.

DATA_TYPE = u'mackeeper:cache'

FORMAT_STRING_PIECES = [u'{description}', u'<{event_type}>', u':', u'{text}', u'[', u

FORMAT_STRING_SHORT_PIECES = [u'<{event_type}>', u'{text}']

SOURCE_LONG = u'MacKeeper Cache'

SOURCE_SHORT = u'LOG'

plaso.formatters.mactime module

The Sleuthkit (TSK) bodyfile (or mactime) event formatter.

class plaso.formatters.mactime.**MactimeFormatter**
Bases: *plaso.formatters.interface.EventFormatter*

Formatter for a mactime event.

DATA_TYPE = u'fs:mactime:line'

FORMAT_STRING = u'{filename}'

SOURCE_LONG = u'Mactime Bodyfile'

SOURCE_SHORT = u'FILE'

plaso.formatters.manager module

This file contains the event formatters manager class.

class plaso.formatters.manager.**FormattersManager**
Bases: object

Class that implements the formatters manager.

classmethod DeregisterFormatter(*formatter_class*)
Deregisters a formatter class.

The formatter classes are identified based on their lower case data type.

Parameters **formatter_class** (*type*) – class of the formatter.

Raises KeyError – if formatter class is not set for the corresponding data type.

classmethod GetFormatterObject(*data_type*)
Retrieves the formatter object for a specific data type.

Parameters **data_type** (*str*) – data type.

Returns

corresponding formatter or the default formatter if not available.

Return type *EventFormatter*

classmethod GetMessageStrings(*formatter_mediator*, *event*)
Retrieves the formatted message strings for a specific event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns long and short version of the message string.

Return type list[str, str]

classmethod GetSourceStrings(*event*)
Retrieves the formatted source strings for a specific event object.

Parameters **event** (EventObject) – event.

Returns short and long version of the source of the event.

Return type list[str, str]

classmethod RegisterFormatter(*formatter_class*)
Registers a formatter class.

The formatter classes are identified based on their lower case data type.

Parameters **formatter_class** (*type*) – class of the formatter.

Raises KeyError – if formatter class is already set for the corresponding data type.

classmethod RegisterFormatters(*formatter_classes*)
Registers formatter classes.

The formatter classes are identified based on their lower case data type.

Parameters **formatter_classes** (*list[type]*) – classes of the formatters.

Raises KeyError – if formatter class is already set for the corresponding data type.

plaso.formatters.mcafeeav module

The McAfee AV Logs file event formatter.

class plaso.formatters.mcafeeav.**McafeeAccessProtectionLogEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a McAfee Access Protection Log event.

DATA_TYPE = u'av:mcafee:accessprotectionlog'

FORMAT_STRING_PIECES = [u'File Name: {filename}', u'User: {username}', u'{trigger_lo

FORMAT_STRING_SHORT_PIECES = [u'{filename}', u'{action}']

SOURCE_LONG = u'McAfee Access Protection Log'

SOURCE_SHORT = u'LOG'

plaso.formatters.mediator module

The formatter mediator object.

class plaso.formatters.mediator.**FormatterMediator**(*data_location=None*)
Bases: object

Class that implements the formatter mediator.

DEFAULT_LANGUAGE_IDENTIFIER = u'en-US'

DEFAULT_LCID = 1033

GetWindowsEventMessage(*log_source*, *message_identifier*)
Retrieves the message string for a specific Windows Event Log source.

Parameters

- **log_source** (*str*) – Event Log source, such as “Application Error”.
- **message_identifier** (*int*) – message identifier.

Returns message string or None if not available.

Return type str

SetPreferredLanguageIdentifier(*language_identifier*)
Sets the preferred language identifier.

Parameters **language_identifier** (*str*) – language identifier string such as "en-US" for US English or "is-IS" for Icelandic.

Raises

- KeyError – if the language identifier is not defined.
- TypeError – if the language identifier is not a string type.

lcid
int – preferred Language Code identifier (LCID).

plaso.formatters.msie_webcache module

The MSIE WebCache ESE database event formatters.

class plaso.formatters.msie_webcache.**MsieWebCacheContainerEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a MSIE WebCache ESE database Container_# table record.

DATA_TYPE = u'msie:webcache:container'

FORMAT_STRING_PIECES = [u'URL: {url}', u'Redirect URL: {redirect_url}', u'Access count

FORMAT_STRING_SHORT_PIECES = [u'URL: {url}']

SOURCE_LONG = u'MSIE WebCache container record'

SOURCE_SHORT = u'WEBHIST'

class plaso.formatters.msie_webcache.**MsieWebCacheContainersEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a MSIE WebCache ESE database Containers table record.

DATA_TYPE = u'msie:webcache:containers'

FORMAT_STRING_PIECES = [u'Name: {name}', u'Directory: {directory}', u'Table: Contai

FORMAT_STRING_SHORT_PIECES = [u'Directory: {directory}']

SOURCE_LONG = u'MSIE WebCache containers record'

SOURCE_SHORT = u'WEBHIST'

class plaso.formatters.msie_webcache.**MsieWebCacheLeakFilesEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a MSIE WebCache ESE database LeakFiles table record.

DATA_TYPE = u'msie:webcache:leak_file'

FORMAT_STRING_PIECES = [u'Filename: {cached_filename}', u'Leak identifier: {leak_ide

FORMAT_STRING_SHORT_PIECES = [u'Filename: {cached_filename}']

SOURCE_LONG = u'MSIE WebCache partitions record'

SOURCE_SHORT = u'WEBHIST'

class plaso.formatters.msie_webcache.**MsieWebCachePartitionsEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a MSIE WebCache ESE database Partitions table record.

DATA_TYPE = u'msie:webcache:partitions'

FORMAT_STRING_PIECES = [u'Partition identifier: {partition_identifier}', u'Partition

FORMAT_STRING_SHORT_PIECES = [u'Directory: {directory}']

SOURCE_LONG = u'MSIE WebCache partitions record'

SOURCE_SHORT = u'WEBHIST'

plaso.formatters.msiecf module

The Microsoft Internet Explorer (MSIE) Cache Files (CF) event formatters.

class plaso.formatters.msiecf.**MsiecfItemFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a MSIECF item event.

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

class plaso.formatters.msiecf.**MsiecfLeakFormatter**
Bases: *plaso.formatters.msiecf.MsiecfItemFormatter*

Formatter for a MSIECF leak item event.

DATA_TYPE = u'msiecf:leak'

FORMAT_STRING_PIECES = [u'Cached file: {cached_file_path}', u'Cached file size: {cac

FORMAT_STRING_SHORT_PIECES = [u'Cached file: {cached_file_path}']

SOURCE_LONG = u'MSIE Cache File leak record'

SOURCE_SHORT = u'WEBHIST'

class plaso.formatters.msiecf.**MsiecfRedirectedFormatter**
Bases: *plaso.formatters.msiecf.MsiecfItemFormatter*

Formatter for a MSIECF leak redirected event.

DATA_TYPE = u'msiecf:redirected'

FORMAT_STRING_PIECES = [u'Location: {url}', u'{recovered_string}']

FORMAT_STRING_SHORT_PIECES = [u'Location: {url}']

SOURCE_LONG = u'MSIE Cache File redirected record'

SOURCE_SHORT = u'WEBHIST'

class plaso.formatters.msiecf.**MsiecfUrlFormatter**
Bases: *plaso.formatters.msiecf.MsiecfItemFormatter*

Formatter for a MSIECF URL item event.

DATA_TYPE = u'msiecf:url'

FORMAT_STRING_PIECES = [u'Location: {url}', u'Number of hits: {number_of_hits}', u'C

FORMAT_STRING_SHORT_PIECES = [u'Location: {url}', u'Cached file: {cached_file_path}'

SOURCE_LONG = u'MSIE Cache File URL record'

SOURCE_SHORT = u'WEBHIST'

plaso.formatters.officemru module

The Microsoft Office MRU Windows Registry event formatter.

class plaso.formatters.officemru.**OfficeMRUWindowsRegistryEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Microsoft Office MRU Windows Registry event.

DATA_TYPE = u'windows:registry:office_mru'

FORMAT_STRING_PIECES = [u'[{key_path}]', u'Value: {value_string}']

FORMAT_STRING_SHORT_PIECES = [u'{value_string}']

SOURCE_LONG = u'Registry Key: Microsoft Office MRU'

SOURCE_SHORT = u'REG'

plaso.formatters.olecf module

The OLE Compound File (OLECF) event formatters.

class plaso.formatters.olecf.**OLECFDestListEntryFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an OLECF DestList stream event.

DATA_TYPE = u'olecf:dest_list:entry'

FORMAT_STRING_PIECES = [u'Entry: {entry_number}', u'Pin status: {pin_status}', u'Hos

FORMAT_STRING_SHORT_PIECES = [u'Entry: {entry_number}', u'Pin status: {pin_status}',

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises `WrongFormatter` – if the event object cannot be formatted by the formatter.

class plaso.formatters.olecf.**OLECFDocumentSummaryInfoFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an OLECF Document Summary Info property set stream event.

DATA_TYPE = u'olecf:document_summary_info'

FORMAT_STRING_PIECES = [u'Number of bytes: {number_of_bytes}', u'Number of lines: {n

FORMAT_STRING_SHORT_PIECES = [u'Company: {company}']

SOURCE_LONG = u'OLECF Document Summary Info'

SOURCE_SHORT = u'OLECF'

class plaso.formatters.olecf.**OLECFItemFormatter**
Bases: *plaso.formatters.interface.EventFormatter*

Formatter for an OLECF item event.

DATA_TYPE = u'olecf:item'

FORMAT_STRING = u'Name: {name}'

FORMAT_STRING_SHORT = u'Name: {name}'

SOURCE_LONG = u'OLECF Item'

SOURCE_SHORT = u'OLECF'

class plaso.formatters.olecf.**OLECFSummaryInfoFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an OLECF Summary Info property set stream event.

DATA_TYPE = u'olecf:summary_info'

FORMAT_STRING_PIECES = [u'Title: {title}', u'Subject: {subject}', u'Author: {author

FORMAT_STRING_SHORT_PIECES = [u'Title: {title}', u'Subject: {subject}', u'Author: {

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises `WrongFormatter` – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'OLECF Summary Info'

SOURCE_SHORT = u'OLECF'

plaso.formatters.opera module

The Opera history event formatters.

class plaso.formatters.opera.**OperaGlobalHistoryFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an Opera global history event.

DATA_TYPE = u'opera:history:entry'

FORMAT_STRING_PIECES = [u'{url}', u'({title})', u'[{description}]']

SOURCE_LONG = u'Opera Browser History'

SOURCE_SHORT = u'WEBHIST'

class plaso.formatters.opera.**OperaTypedHistoryFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an Opera typed history event.

DATA_TYPE = u'opera:history:typed_entry'

FORMAT_STRING_PIECES = [u'{url}', u'({entry_selection})']

SOURCE_LONG = u'Opera Browser History'

SOURCE_SHORT = u'WEBHIST'

plaso.formatters.oxml module

The OpenXML event formatter.

class plaso.formatters.oxml.**OpenXMLParserFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an OXML event.

DATA_TYPE = u'metadata:openxml'

FORMAT_STRING_PIECES = [u'Creating App: {creating_app}', u'App version: {app_version

FORMAT_STRING_SHORT_PIECES = [u'Title: {title}', u'Subject: {subject}', u'Author: {

SOURCE_LONG = u'Open XML Metadata'

SOURCE_SHORT = u'META'

plaso.formatters.pcap module

The PCAP event formatter.

class plaso.formatters.pcap.**PCAPFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a PCAP event.

DATA_TYPE = u'metadata:pcap'

FORMAT_STRING_PIECES = [u'Source IP: {source_ip}', u'Destination IP: {dest_ip}', u'Sou

FORMAT_STRING_SHORT_PIECES = [u'Type: {stream_type}', u'First Packet ID: {first_packe

SOURCE_LONG = u'Packet Capture File (pcap)'

SOURCE_SHORT = u'PCAP'

plaso.formatters.pe module

The PE event formatter.

class plaso.formatters.pe.**PECompilationFormatter**
Bases: *plaso.formatters.pe.PEEventFormatter*

Formatter for a PE compilation event.

DATA_TYPE = u'pe:compilation:compilation_time'

SOURCE_LONG = u'PE Compilation time'

class plaso.formatters.pe.**PEDelayImportFormatter**
Bases: *plaso.formatters.pe.PEEventFormatter*

Formatter for a PE delay import section event.

DATA_TYPE = u'pe:delay_import:import_time'

FORMAT_STRING_PIECES = [u'DLL name: {dll_name}', u'PE Type: {pe_type}', u'Import hash

FORMAT_STRING_SHORT_PIECES = [u'{dll_name}']

SOURCE_LONG = u'PE Delay Import Time'

class plaso.formatters.pe.**PEEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Parent class for PE event formatters.

DATA_TYPE = u'pe'

FORMAT_STRING_PIECES = [u'PE Type: {pe_type}', u'Import hash: {imphash}']

FORMAT_STRING_SEPARATOR = u' '

FORMAT_STRING_SHORT_PIECES = [u'pe_type']

SOURCE_LONG = u'PE Event'

SOURCE_SHORT = u'PE'

class plaso.formatters.pe.**PEImportFormatter**
Bases: *plaso.formatters.pe.PEEventFormatter*

Formatter for a PE import section event.

DATA_TYPE = u'pe:import:import_time'

FORMAT_STRING_PIECES = [u'DLL name: {dll_name}', u'PE Type: {pe_type}', u'Import hash

FORMAT_STRING_SHORT_PIECES = [u'{dll_name}']

SOURCE_LONG = u'PE Import Time'

class plaso.formatters.pe.**PELoadConfigModificationEvent**
Bases: *plaso.formatters.pe.PEEventFormatter*

Formatter for a PE load configuration table event.

DATA_TYPE = u'pe:load_config:modification_time'

SOURCE_LONG = u'PE Load Configuration Table Time'

class plaso.formatters.pe.**PEResourceCreationFormatter**
Bases: *plaso.formatters.pe.PEEventFormatter*

Formatter for a PE resource creation event.

DATA_TYPE = u'pe:resource:creation_time'

SOURCE_LONG = u'PE Resource Creation Time'

plaso.formatters.plist module

The plist event formatter.

class plaso.formatters.plist.**PlistFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a plist key event.

DATA_TYPE = u'plist:key'

FORMAT_STRING_PIECES = [u'{root}/', u'{key}', u' {desc}']

FORMAT_STRING_SEPARATOR = u''

SOURCE_LONG = u'Plist Entry'

SOURCE_SHORT = u'PLIST'

plaso.formatters.pls_recall module

The PL/SQL Recall event formatter.

class plaso.formatters.pls_recall.**PlsRecallFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a PL/SQL Recall file container event.

DATA_TYPE = u'PLSRecall:event'

FORMAT_STRING_PIECES = [u'Sequence number: {sequence_number}', u'Username: {username

FORMAT_STRING_SHORT_PIECES = [u'{sequence_number}', u'{username}', u'{database_name}',

SOURCE_LONG = u'PL/SQL Developer Recall file'

SOURCE_SHORT = u'PLSRecall'

plaso.formatters.popcontest module

The Popularity Contest event formatters.

class plaso.formatters.popcontest.**PopularityContestLogFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Popularity Contest Log event.

DATA_TYPE = u'popularity_contest:log:event'

FORMAT_STRING_PIECES = [u'mru [{mru}]', u'package [{package}]', u'tag [{record_tag}]']

FORMAT_STRING_SHORT_PIECES = [u'{mru}']

SOURCE_LONG = u'Popularity Contest Log'

SOURCE_SHORT = u'LOG'

class plaso.formatters.popcontest.**PopularityContestSessionFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Popularity Contest Session information event.

DATA_TYPE = u'popularity_contest:session:event'

FORMAT_STRING_PIECES = [u'Session {session}', u'{status}', u'ID {hostid}', u'[{details

FORMAT_STRING_SHORT_PIECES = [u'Session {session}', u'{status}']

SOURCE_LONG = u'Popularity Contest Session'

SOURCE_SHORT = u'LOG'

plaso.formatters.recycler module

The Windows Recycler/Recycle Bin formatter.

class plaso.formatters.recycler.**WinRecyclerFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Windows Recycler/Recycle Bin file event.

DATA_TYPE = u'windows:metadata:deleted_item'

FORMAT_STRING_PIECES = [u'DC{record_index} ->', u'{original_filename}', u'[{short_file

FORMAT_STRING_SHORT_PIECES = [u'Deleted file: {original_filename}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Recycle Bin'

SOURCE_SHORT = u'RECBIN'

plaso.formatters.safari module

The Safari history event formatter.

class plaso.formatters.safari.**SafariHistoryFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Safari history event.

DATA_TYPE = u'safari:history:visit'

FORMAT_STRING_PIECES = [u'Visited: {url}', u'({title}', u'- {display_title}', u')', u

SOURCE_LONG = u'Safari History'

SOURCE_SHORT = u'WEBHIST'

class plaso.formatters.safari.**SafariHistoryFormatterSqlite**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Safari history event from Sqlite History.db

DATA_TYPE = u'safari:history:visit_sqlite'

FORMAT_STRING_PIECES = [u'URL: {url}', u'Title: ({title})', u'[count: {visit_count}]

SOURCE_LONG = u'Safari History'

SOURCE_SHORT = u'WEBHIST'

plaso.formatters.safari_cookies module

The Safari Binary cookie event formatter.

class plaso.formatters.safari_cookies.**SafaryCookieFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Safari Binary Cookie file entry event.

DATA_TYPE = u'safari:cookie:entry'

FORMAT_STRING_PIECES = [u'{url}', u'<{path}>', u'({cookie_name})', u'Flags: {flags}']

FORMAT_STRING_SHORT_PIECES = [u'{url}', u'({cookie_name})']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Safari Cookies'

SOURCE_SHORT = u'WEBHIST'

plaso.formatters.sam_users module

The SAM users Windows Registry event formatter.

class plaso.formatters.sam_users.**SAMUsersWindowsRegistryEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a SAM users Windows Registry event.

DATA_TYPE = u'windows:registry:sam_users'

FORMAT_STRING_PIECES = [u'[{key_path}]', u'Username: {username}', u'Full name: {fulln

FORMAT_STRING_SHORT_PIECES = [u'{username}', u'RID: {account_rid}', u'Login count: {lo

SOURCE_LONG = u'Registry Key: User Account Information'

SOURCE_SHORT = u'REG'

plaso.formatters.sccm module

The SCCM log formatter.

class plaso.formatters.sccm.**SCCMEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Class for SCCM event formatter.

DATA_TYPE = u'software_management:sccm:log'

FORMAT_STRING_PIECES = [u'{component}', u'{text}']

FORMAT_STRING_SEPARATOR = u' '

FORMAT_STRING_SHORT_PIECES = [u'{text}']

SOURCE_LONG = u'SCCM Event'

SOURCE_SHORT = u'LOG'

plaso.formatters.selinux module

The selinux event formatter.

class plaso.formatters.selinux.**SELinuxFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a selinux log file event.

DATA_TYPE = u'selinux:line'

FORMAT_STRING_PIECES = [u'[', u'audit_type: {audit_type}', u', pid: {pid}', u']', u'

FORMAT_STRING_SEPARATOR = u''

SOURCE_LONG = u'Audit log File'

SOURCE_SHORT = u'LOG'

plaso.formatters.shell_items module

The shell item event formatter.

class plaso.formatters.shell_items.**ShellItemFileEntryEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a shell item file entry event.

DATA_TYPE = u'windows:shell_item:file_entry'

FORMAT_STRING_PIECES = [u'Name: {name}', u'Long name: {long_name}', u'Localized name

FORMAT_STRING_SHORT_PIECES = [u'Name: {file_entry_name}', u'NTFS file reference: {fi

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'File entry shell item'

SOURCE_SHORT = u'FILE'

plaso.formatters.shutdown module

The shutdown Windows Registry event formatter.

class plaso.formatters.shutdown.**ShutdownWindowsRegistryEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a shutdown Windows Registry event.

DATA_TYPE = u'windows:registry:shutdown'

FORMAT_STRING_PIECES = [u'[{key_path}]', u'Description: {value_name}']

FORMAT_STRING_SHORT_PIECES = [u'{value_name}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Registry Key Shutdown Entry'

SOURCE_SHORT = u'REG'

plaso.formatters.skydrivelog module

The SkyDrive log event formatter.

class plaso.formatters.skydrivelog.**SkyDriveLogFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a SkyDrive log file event.

DATA_TYPE = u'skydrive:log:line'

FORMAT_STRING_PIECES = [u'[{module}', u'{source_code}', u'{log_level}]', u'{detail}']

FORMAT_STRING_SHORT_PIECES = [u'{detail}']

SOURCE_LONG = u'SkyDrive Log File'

SOURCE_SHORT = u'LOG'

class plaso.formatters.skydrivelog.**SkyDriveOldLogFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a SkyDrive old log file event.

DATA_TYPE = u'skydrive:log:old:line'

FORMAT_STRING_PIECES = [u'[{source_code}]', u'({log_level})', u'{text}']

FORMAT_STRING_SHORT_PIECES = [u'{text}']

SOURCE_LONG = u'SkyDrive Log File'

SOURCE_SHORT = u'LOG'

plaso.formatters.skype module

The Skype main database event formatter.

class plaso.formatters.skype.**SkypeAccountFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Skype account event.

DATA_TYPE = u'skype:event:account'

FORMAT_STRING_PIECES = [u'{username}', u'[{email}]', u'Country: {country}']

SOURCE_LONG = u'Skype Account'

SOURCE_SHORT = u'LOG'

class plaso.formatters.skype.**SkypeCallFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Skype call event.

DATA_TYPE = u'skype:event:call'

FORMAT_STRING_PIECES = [u'From: {src_call}', u'To: {dst_call}', u'[{call_type}]']

SOURCE_LONG = u'Skype Call'

SOURCE_SHORT = u'LOG'

class plaso.formatters.skype.**SkypeChatFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Skype chat message event.

DATA_TYPE = u'skype:event:chat'

FORMAT_STRING_PIECES = [u'From: {from_account}', u'To: {to_account}', u'[{title}]',

FORMAT_STRING_SHORT_PIECES = [u'From: {from_account}', u'To: {to_account}']

SOURCE_LONG = u'Skype Chat MSG'

SOURCE_SHORT = u'LOG'

class plaso.formatters.skype.**SkypeSMSFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Skype SMS event.

DATA_TYPE = u'skype:event:sms'

FORMAT_STRING_PIECES = [u'To: {number}', u'[{text}]']

SOURCE_LONG = u'Skype SMS'

SOURCE_SHORT = u'LOG'

class plaso.formatters.skype.**SkypeTransferFileFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Skype transfer file event.

DATA_TYPE = u'skype:event:transferfile'

FORMAT_STRING_PIECES = [u'Source: {source}', u'Destination: {destination}', u'File:

SOURCE_LONG = u'Skype Transfer Files'

SOURCE_SHORT = u'LOG'

plaso.formatters.sophos_av module

The Sophos Anti-Virus log (SAV.txt) file event formatter.

class plaso.formatters.sophos_av.**SophosAVLogFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Sophos Anti-Virus log (SAV.txt) event data.

DATA_TYPE = u'sophos:av:log'

FORMAT_STRING_PIECES = [u'{text}']

SOURCE_LONG = u'Sophos Anti-Virus log'

SOURCE_SHORT = u'LOG'

plaso.formatters.srum module

The System Resource Usage Monitor (SRUM) ESE database event formatters.

class plaso.formatters.srum.**SRUMApplicationResourceUsageEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a SRUM application resource usage event.

DATA_TYPE = u'windows:srum:application_usage'

FORMAT_STRING_PIECES = [u'Application: {application}']

FORMAT_STRING_SHORT_PIECES = [u'{application}']

class plaso.formatters.srum.**SRUMNetworkConnectivityUsageEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a SRUM network connectivity usage event.

DATA_TYPE = u'windows:srum:network_connectivity'

FORMAT_STRING_PIECES = [u'Application: {application}']

FORMAT_STRING_SHORT_PIECES = [u'{application}']

class plaso.formatters.srum.**SRUMNetworkDataUsageEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a SRUM network data usage event.

DATA_TYPE = u'windows:srum:network_usage'

FORMAT_STRING_PIECES = [u'Application: {application}', u'Bytes received: {bytes_rece

FORMAT_STRING_SHORT_PIECES = [u'{application}']

plaso.formatters.ssh module

The syslog SSH file event formatter.

class plaso.formatters.ssh.**SSHFailedConnectionEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a SSH failed connection event.

DATA_TYPE = u'syslog:ssh:failed_connection'

FORMAT_STRING_PIECES = [u'Unsuccessful connection of user: {username}', u'from {addre

FORMAT_STRING_SEPARATOR = u''

FORMAT_STRING_SHORT = u'{body}'

SOURCE_LONG = u'SSH log'

SOURCE_SHORT = u'LOG'

class plaso.formatters.ssh.**SSHLoginEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a SSH successful login event.

DATA_TYPE = u'syslog:ssh:login'

FORMAT_STRING_PIECES = [u'Successful login of user: {username}', u'from {address}:',

FORMAT_STRING_SEPARATOR = u''

FORMAT_STRING_SHORT = u'{body}'

SOURCE_LONG = u'SSH log'

SOURCE_SHORT = u'LOG'

class plaso.formatters.ssh.**SSHOpenedConnectionEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a SSH opened connection event.

DATA_TYPE = u'syslog:ssh:opened_connection'

FORMAT_STRING_PIECES = [u'Connection opened {address}:', u'{port}', u'ssh pid: {pid}

FORMAT_STRING_SEPARATOR = u''

FORMAT_STRING_SHORT = u'{body}'

SOURCE_LONG = u'SSH log'

SOURCE_SHORT = u'LOG'

plaso.formatters.symantec module

The Symantec AV log file event formatter.

class plaso.formatters.symantec.**SymantecAVFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Symantec AV log file event.

ACTION_0_NAMES = {u'11': u'Undo action in Quarantine View', u'10': u'Renamed backup

ACTION_1_2_NAMES = {u'1': u'Quarantine infected file', u'3': u'Delete infected file'

CATEGORY_NAMES = {u'1': u'GL_CAT_INFECTION', u'3': u'GL_CAT_PATTERN', u'2': u'GL_CA

DATA_TYPE = u'av:symantec:scanlog'

EVENT_NAMES = {u'56': u'GL_EVENT_CLIENT_INSTALL_FW', u'77': u'GL_EVENT_HEUR_THREAT_N

FORMAT_STRING_PIECES = [u'Event Name: {event_map}', u'Category Name: {category_map}'

FORMAT_STRING_SEPARATOR = u'; '

FORMAT_STRING_SHORT_PIECES = [u'{file}', u'{virus}', u'{action0_map}', u'{action1_map}

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Symantec AV Log'

SOURCE_SHORT = u'LOG'

plaso.formatters.syslog module

The syslog file event formatter.

class plaso.formatters.syslog.**SyslogCommentFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a syslog comment

DATA_TYPE = u'syslog:comment'

FORMAT_STRING_PIECES = [u'{body}']

FORMAT_STRING_SEPARATOR = u''

SOURCE_LONG = u'Log File'

SOURCE_SHORT = u'LOG'

class plaso.formatters.syslog.**SyslogLineFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a syslog line event.

DATA_TYPE = u'syslog:line'

FORMAT_STRING_PIECES = [u'{severity} ', u'[', u'{reporter}', u', pid: {pid}', u'] {bo

FORMAT_STRING_SEPARATOR = u''

SOURCE_LONG = u'Log File'

SOURCE_SHORT = u'LOG'

plaso.formatters.systemd_journal module

The Systemd journal file event formatter.

class plaso.formatters.systemd_journal.**SystemdJournalEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Systemd journal event.

DATA_TYPE = u'systemd:journal'

FORMAT_STRING_PIECES = [u'{hostname} ', u'[', u'{reporter}', u', pid: {pid}', u'] {bo

FORMAT_STRING_SEPARATOR = u''

SOURCE_LONG = u'systemd-journal'

SOURCE_SHORT = u'LOG'

plaso.formatters.task_scheduler module

The Task Scheduler event formatter.

class plaso.formatters.task_scheduler.**TaskCacheEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Task Scheduler Cache event.

DATA_TYPE = u'task_scheduler:task_cache:entry'

FORMAT_STRING_PIECES = [u'Task: {task_name}', u'[Identifier: {task_identifier}]']

FORMAT_STRING_SHORT_PIECES = [u'Task: {task_name}']

SOURCE_LONG = u'Task Cache'

SOURCE_SHORT = u'REG'

plaso.formatters.text module

The text file event formatter.

class plaso.formatters.text.**TextEntryFormatter**
Bases: *plaso.formatters.interface.EventFormatter*

Formatter for a text file entry event.

DATA_TYPE = u'text:entry'

FORMAT_STRING = u'{text}'

SOURCE_LONG = u'Text File'

SOURCE_SHORT = u'LOG'

plaso.formatters.trendmicroav module

The Trend Micro AV Logs file event formatter.

class plaso.formatters.trendmicroav.**OfficeScanVirusDetectionLogEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Trend Micro Office Scan Virus Detection Log event.

DATA_TYPE = u'av:trendmicro:scan'

FORMAT_STRING_PIECES = [u'Path: {path}', u'File name: {filename}', u'{threat}', u':

FORMAT_STRING_SHORT_PIECES = [u'{path}', u'{filename}', u'{action}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

If any event values have a matching formatting function in VALUE_FORMATTERS, they are run through that function; then the dictionary is passed to the superclass's formatting method.

Parameters

- **unused_formatter_mediator** (FormatterMediator) – not used.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Trend Micro Office Scan Virus Detection Log'

SOURCE_SHORT = u'LOG'

VALUE_FORMATTERS = {u'action': <function <lambda>>, u'scan_type': <function <lambda>

plaso.formatters.twitter_ios module

Twitter on iOS 8+ database formatter.

class plaso.formatters.twitter_ios.**TwitterIOSContactFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Twitter on iOS 8+ contact event formatter.

DATA_TYPE = u'twitter:ios:contact'

FORMAT_STRING_PIECES = [u'Screen name: {screen_name}', u'Profile picture URL: {profil

FORMAT_STRING_SHORT_PIECES = [u'Screen name: {screen_name}', u'Description: {descrip

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Twitter iOS Contacts'

SOURCE_SHORT = u'Twitter iOS'

class plaso.formatters.twitter_ios.**TwitterIOSStatusFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Twitter on iOS 8+ status event formatter.

DATA_TYPE = u'twitter:ios:status'

FORMAT_STRING_PIECES = [u'Name: {name}', u'User Id: {user_id}', u'Message: {text}',

FORMAT_STRING_SHORT_PIECES = [u'Name: {name}', u'Message: {text}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises `WrongFormatter` – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Twitter iOS Status'

SOURCE_SHORT = u'Twitter iOS'

plaso.formatters.userassist module

The UserAssist Windows Registry event formatter.

class plaso.formatters.userassist.**UserAssistWindowsRegistryEventFormatter**

Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an UserAssist Windows Registry event.

DATA_TYPE = u'windows:registry:userassist'

FORMAT_STRING_PIECES = [u'[{key_path}]', u'UserAssist entry: {entry_index}', u'Value

FORMAT_STRING_SHORT_PIECES = [u'{value_name}', u'Count: {number_of_executions}']

SOURCE_LONG = u'Registry Key: UserAssist'

SOURCE_SHORT = u'REG'

plaso.formatters.utmp module

The UTMP binary file event formatter.

class plaso.formatters.utmp.**UtmpSessionFormatter**

Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an UTMP session event.

DATA_TYPE = u'linux:utmp:event'

FORMAT_STRING_PIECES = [u'User: {user}', u'Computer Name: {computer_name}', u'Termina

FORMAT_STRING_SHORT_PIECES = [u'User: {user}']

SOURCE_LONG = u'UTMP session'

SOURCE_SHORT = u'LOG'

plaso.formatters.utmpx module

The UTMPX binary file event formatter.

class plaso.formatters.utmpx.**UtmpxSessionFormatter**

Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for an UTMPX session event.

DATA_TYPE = u'mac:utmpx:event'

FORMAT_STRING_PIECES = [u'User: {user}', u'Status: {status}', u'Computer Name: {comp

FORMAT_STRING_SHORT_PIECES = [u'User: {user}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'UTMPX session'

SOURCE_SHORT = u'LOG'

plaso.formatters.windows module

The Windows event formatter.

class plaso.formatters.windows.**WindowsDistributedLinkTrackingCreationEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Windows distributed link creation event.

DATA_TYPE = u'windows:distributed_link_tracking:creation'

FORMAT_STRING_PIECES = [u'{uuid}', u'MAC address: {mac_address}', u'Origin: {origin}

FORMAT_STRING_SHORT_PIECES = [u'{uuid}', u'Origin: {origin}']

SOURCE_LONG = u'System'

SOURCE_SHORT = u'LOG'

class plaso.formatters.windows.**WindowsRegistryInstallationEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Windows installation event.

DATA_TYPE = u'windows:registry:installation'

FORMAT_STRING_PIECES = [u'{product_name}', u'{version}', u'{service_pack}', u'Owner:

FORMAT_STRING_SHORT_PIECES = [u'{product_name}', u'{version}', u'{service_pack}', u'Or

SOURCE_LONG = u'System'

SOURCE_SHORT = u'LOG'

class plaso.formatters.windows.**WindowsRegistryListEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Windows list event e.g. MRU or Jump list.

DATA_TYPE = u'windows:registry:list'

FORMAT_STRING_PIECES = [u'Key: {key_path}', u'Value: {value_name}', u'List: {list_na

SOURCE_LONG = u'System'

SOURCE_SHORT = u'LOG'

class plaso.formatters.windows.**WindowsRegistryNetworkEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Windows network event.

DATA_TYPE = u'windows:registry:network'

FORMAT_STRING_PIECES = [u'SSID: {ssid}', u'Description: {description}', u'Connection '

SOURCE_LONG = u'System: Network Connection'

SOURCE_SHORT = u'LOG'

class plaso.formatters.windows.**WindowsVolumeCreationEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Windows volume creation event.

DATA_TYPE = u'windows:volume:creation'

FORMAT_STRING_PIECES = [u'{device_path}', u'Serial number: 0x{serial_number:08X}', u'

FORMAT_STRING_SHORT_PIECES = [u'{device_path}', u'Origin: {origin}']

SOURCE_LONG = u'System'

SOURCE_SHORT = u'LOG'

plaso.formatters.winevt module

The Windows EventLog (EVT) file event formatter.

class plaso.formatters.winevt.**WinEVTFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Windows EventLog (EVT) record event.

DATA_TYPE = u'windows:evt:record'

FORMAT_STRING_PIECES = [u'[{event_identifier} /', u'0x{event_identifier:04x}]', u'Sour

**FORMAT_STRING_SHORT_PIECES = [u'[{event_identifier} /', u'0x{event_identifier:04x}]', **

GetEventTypeString(*event_type*)
Retrieves a string representation of the event type.

Parameters **event_type** (*int*) – event type.

Returns description of the event type.

Return type str

GetMessages(*formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

GetSeverityString(*severity*)
Retrieves a string representation of the severity.

Parameters **severity** (*int*) – severity.

Returns description of the event severity.

Return type str

SOURCE_LONG = u'WinEVT'

SOURCE_SHORT = u'EVT'

plaso.formatters.winevt_rc module

Windows Event Log resources database reader.

class plaso.formatters.winevt_rc.**Sqlite3DatabaseFile**
Bases: object

Class that defines a sqlite3 database file.

Close()
Closes the database file.

Raises RuntimeError – if the database is not opened.

GetValues(*table_names*, *column_names*, *condition*)
Retrieves values from a table.

Parameters

- **table_names** (*list[str]*) – table names.
- **column_names** (*list[str]*) – column names.
- **condition** (*str*) – query condition such as "log_source == 'Application Error'".

Yields *sqlite3.row* – row.

Raises RuntimeError – if the database is not opened.

HasTable(*table_name*)
Determines if a specific table exists.

Parameters **table_name** (*str*) – table name.

Returns True if the table exists.

Return type bool

Raises RuntimeError – if the database is not opened.

Open(*filename*, *read_only=False*)
Opens the database file.

Parameters

- **filename** (*str*) – filename of the database.
- **read_only** (*Optional[bool]*) – True if the database should be opened in read-only mode. Since sqlite3 does not support a real read-only mode we fake it by only permitting SELECT queries.

Returns True if successful.

Return type bool

Raises RuntimeError – if the database is already opened.

class plaso.formatters.winevt_rc.**Sqlite3DatabaseReader**
Bases: object

Class to represent a sqlite3 database reader.

Close()
Closes the database reader object.

Open(*filename*)
Opens the database reader object.

Parameters **filename** (*str*) – filename of the database.

Returns True if successful.

Return type bool

class plaso.formatters.winevt_rc.**WinevtResourcesSqlite3DatabaseReader**
Bases: *plaso.formatters.winevt_rc.Sqlite3DatabaseReader*

Class to represent a sqlite3 Event Log resources database reader.

GetMessage(*log_source*, *lcid*, *message_identifier*)
Retrieves a specific message for a specific Event Log source.

Parameters

- **log_source** (*str*) – Event Log source.
- **lcid** (*int*) – language code identifier (LCID).
- **message_identifier** (*int*) – message identifier.

Returns message string or None if not available.

Return type str

GetMetadataAttribute(*attribute_name*)
Retrieves the metadata attribute.

Parameters **attribute_name** (*str*) – name of the metadata attribute.

Returns the metadata attribute or None.

Return type str

Raises RuntimeError – if more than one value is found in the database.

Open(*filename*)
Opens the database reader object.

Parameters **filename** (*str*) – filename of the database.

Returns True if successful.

Return type bool

Raises RuntimeError – if the version or string format of the database is not supported.

plaso.formatters.winevtx module

The Windows XML EventLog (EVTX) file event formatter.

class plaso.formatters.winevtx.**WinEVTXFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Windows XML EventLog (EVTX) record event.

DATA_TYPE = u'windows:evtx:record'

FORMAT_STRING_PIECES = [u'[{event_identifier} /', u'0x{event_identifier:04x}]', u'Sour

FORMAT_STRING_SHORT_PIECES = [u'[{event_identifier} /', u'0x{event_identifier:04x}]',

GetMessages(*formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'WinEVTX'

SOURCE_SHORT = u'EVT'

plaso.formatters.winfirewall module

The Windows firewall log file event formatter.

class plaso.formatters.winfirewall.**WinFirewallFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Windows firewall log entry event.

DATA_TYPE = u'windows:firewall:log_entry'

FORMAT_STRING_PIECES = [u'{action}', u'[', u'{protocol}', u'{path}', u']', u'From: {s

FORMAT_STRING_SHORT_PIECES = [u'{action}', u'[{protocol}]', u'{source_ip}', u': {sour

SOURCE_LONG = u'Windows Firewall Log'

SOURCE_SHORT = u'LOG'

plaso.formatters.winjob module

The Windows Scheduled Task (job) event formatter.

class plaso.formatters.winjob.**WinJobFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Windows Scheduled Task (job) event.

DATA_TYPE = u'windows:tasks:job'

FORMAT_STRING_PIECES = [u'Application: {application}', u'{parameters}', u'Scheduled b

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Windows Scheduled Task Job'

SOURCE_SHORT = u'JOB'

plaso.formatters.winlnk module

The Windows Shortcut (LNK) event formatter.

class plaso.formatters.winlnk.**WinLnkLinkFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Windows Shortcut (LNK) link event.

DATA_TYPE = u'windows:lnk:link'

FORMAT_STRING_PIECES = [u'[{description}]', u'File size: {file_size}', u'File attribu

FORMAT_STRING_SHORT_PIECES = [u'[{description}]', u'{linked_path}', u'{command_line_ar

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Windows Shortcut'

SOURCE_SHORT = u'LNK'

plaso.formatters.winprefetch module

The Windows Prefetch event formatter.

class plaso.formatters.winprefetch.**WinPrefetchExecutionFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Windows Prefetch execution event.

DATA_TYPE = u'windows:prefetch:execution'

FORMAT_STRING_PIECES = [u'Prefetch', u'[{executable}] was executed -', u'run count {ru

FORMAT_STRING_SHORT_PIECES = [u'{executable} was run', u'{run_count} time(s)']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'WinPrefetch'

SOURCE_SHORT = u'LOG'

plaso.formatters.winreg module

The Windows Registry key or value event formatter.

class plaso.formatters.winreg.**WinRegistryGenericFormatter**
Bases: *plaso.formatters.interface.EventFormatter*

Formatter for a Windows Registry key or value event.

DATA_TYPE = u'windows:registry:key_value'

FORMAT_STRING = u'[{key_path}] {text}'

FORMAT_STRING_ALTERNATIVE = u'{text}'

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

GetSources(*event*)
Determines the the short and long source for an event object.

Parameters **event** (EventObject) – event.

Returns short and long source string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Registry Key'

SOURCE_SHORT = u'REG'

plaso.formatters.winregservice module

The Windows services event formatter.

The Windows services are derived from Windows Registry files.

class plaso.formatters.winregservice.**WinRegistryServiceFormatter**
Bases: *plaso.formatters.winreg.WinRegistryGenericFormatter*

Formatter for a Windows service event.

DATA_TYPE = u'windows:registry:service'

GetMessages(*formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

plaso.formatters.winrestore module

The Windows Restore Point (rp.log) file event formatter.

class plaso.formatters.winrestore.**RestorePointInfoFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a Windows Windows Restore Point information event.

DATA_TYPE = u'windows:restore_point:info'

FORMAT_STRING_PIECES = [u'{description}', u'Event type: {restore_point_event_type}',

FORMAT_STRING_SHORT_PIECES = [u'{description}']

GetMessages(*unused_formatter_mediator*, *event*)
Determines the formatted message strings for an event object.

Parameters

- **formatter_mediator** (FormatterMediator) – mediates the interactions between formatters and other components, such as storage and Windows EventLog resources.
- **event** (EventObject) – event.

Returns formatted message string and short message string.

Return type tuple(str, str)

Raises WrongFormatter – if the event object cannot be formatted by the formatter.

SOURCE_LONG = u'Windows Restore Point'

SOURCE_SHORT = u'RP'

plaso.formatters.xchatlog module

The XChat log file event formatter.

class plaso.formatters.xchatlog.**XChatLogFormatter**

Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a XChat log file entry event.

DATA_TYPE = u'xchat:log:line'

FORMAT_STRING_PIECES = [u'[nickname: {nickname}]', u'{text}']

SOURCE_LONG = u'XChat Log File'

SOURCE_SHORT = u'LOG'

plaso.formatters.xchatscrollback module

The XChat scrollback file event formatter.

class plaso.formatters.xchatscrollback.**XChatScrollbackFormatter**

Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Formatter for a XChat scrollback file entry event.

DATA_TYPE = u'xchat:scrollback:line'

FORMAT_STRING_PIECES = [u'[', u'nickname: {nickname}', u']', u' {text}']

FORMAT_STRING_SEPARATOR = u''

SOURCE_LONG = u'XChat Scrollback File'

SOURCE_SHORT = u'LOG'

plaso.formatters.zeitgeist module

The Zeitgeist event formatter.

class plaso.formatters.zeitgeist.**ZeitgeistFormatter**

Bases: *plaso.formatters.interface.EventFormatter*

Formatter for a Zeitgeist activity database event.

DATA_TYPE = u'zeitgeist:activity'

FORMAT_STRING = u'{subject_uri}'

SOURCE_LONG = u'Zeitgeist activity log'

SOURCE_SHORT = u'LOG'

plaso.formatters.zsh_extended_history module

The Zsh extended_history formatter.

class plaso.formatters.zsh_extended_history.**ZshExtendedHistoryEventFormatter**
Bases: *plaso.formatters.interface.ConditionalEventFormatter*

Class for the Zsh event formatter.

DATA_TYPE = u'shell:zsh:history'

FORMAT_STRING_PIECES = [u'{command}', u'Time elapsed: {elapsed_seconds} seconds']

FORMAT_STRING_SEPARATOR = u' '

FORMAT_STRING_SHORT_PIECES = [u'{command}']

SOURCE_LONG = u'Zsh Extended History'

SOURCE_SHORT = u'HIST'

Module contents

This file contains an import statement for each formatter.

plaso.lib package

Submodules

plaso.lib.binary module

This file contains a helper library to read binary files.

plaso.lib.binary.**ArrayOfUTF16StreamCopyToString**(*byte_stream*, *byte_stream_size=None*)
Copies an array of UTF-16 formatted byte streams to an array of strings.

The UTF-16 formatted byte stream should be terminated by an end-of-string character (). Otherwise the function reads up to the byte stream size.

Parameters

- **byte_stream** – The UTF-16 formatted byte stream.
- **byte_stream_size** – The byte stream size or None if the entire byte stream should be used.

Returns An array of Unicode strings.

plaso.lib.binary.**ArrayOfUTF16StreamCopyToStringTable**(*byte_stream*, *byte_stream_size=None*)
Copies an array of UTF-16 formatted byte streams to a string table.

The string table is a dict of strings with the byte offset as their key. The UTF-16 formatted byte stream should be terminated by an end-of-string character (). Otherwise the function reads up to the byte stream size.

Parameters

- **byte_stream** – The UTF-16 formatted byte stream.
- **byte_stream_size** – The byte stream size or None if the entire byte stream should be used.

Returns A dict of Unicode strings with the byte offset as their key.

plaso.lib.binary.**ByteArrayCopyToString**(*byte_array*, *codepage=u'utf-8'*)
Copies a UTF-8 encoded byte array into a Unicode string.

Parameters

- **byte_array** – A byte array containing an UTF-8 encoded string.
- **codepage** – The codepage of the byte stream.

Returns A Unicode string.

plaso.lib.binary.**ByteStreamCopyToString**(*byte_stream*, *codepage=u'utf-8'*)
Copies a UTF-8 encoded byte stream into a Unicode string.

Parameters

- **byte_stream** – A byte stream containing an UTF-8 encoded string.
- **codepage** – The codepage of the byte stream.

Returns A Unicode string.

plaso.lib.binary.**ByteStreamCopyToUTF16Stream**(*byte_stream*, *byte_stream_size=None*)
Reads an UTF-16 formatted stream from a byte stream.

The UTF-16 formatted stream should be terminated by an end-of-string character (). Otherwise the function reads up to the byte stream size.

Parameters

- **byte_stream** – The byte stream that contains the UTF-16 formatted stream.
- **byte_stream_size** – Optional byte stream size or None if the entire byte stream should be read.

Returns String containing the UTF-16 formatted stream.

plaso.lib.binary.**HexifyBuffer**(*string_buffer*)
Return a string with the hex representation of a string buffer.

plaso.lib.binary.**ReadUTF16**(*string_buffer*)
Returns a decoded UTF-16 string from a string buffer.

plaso.lib.binary.**ReadUTF16Stream**(*file_object*, *offset=None*, *byte_size=0*)
Reads an UTF-16 formatted stream from a file-like object.

Reads an UTF-16 formatted stream that's terminated by an end-of-string character () or up to the byte size.

Parameters

- **file_object** – A file-like object to read the data from.
- **offset** – An offset into the file object data, if -1 or not set the current location into the file object data is used.

- **byte_size** – Maximum number of bytes to read or 0 if the function should keep reading up to the end of file.

Returns An Unicode string.

plaso.lib.binary.**UTF16StreamCopyToString**(*byte_stream*, *byte_stream_size=None*)
Copies an UTF-16 formatted byte stream to a string.

The UTF-16 formatted byte stream should be terminated by an end-of-string character (). Otherwise the function reads up to the byte stream size.

Parameters

- **byte_stream** – The UTF-16 formatted byte stream.
- **byte_stream_size** – The byte stream size or None if the entire byte stream should be used.

Returns An Unicode string.

plaso.lib.bufferlib module

Circular buffer for storing event objects.

class plaso.lib.bufferlib.**CircularBuffer**(*size*)
Bases: object

Class that defines a circular buffer for storing event objects.

Append(*item*)
Add an item to the list.

Parameters **item** (*object*) – item.

Clear()
Removes all elements from the list.

Flush()
Returns a generator for all items and clear the buffer.

GetCurrent()
Retrieves the current item that index points to.

Returns item.

Return type object

__iter__()
Return all elements from the list.

__len__()
Return the length (the fixed size).

size
int – number of elements in the buffer.

plaso.lib.definitions module

The definitions.

plaso.lib.errors module

This file contains the error classes.

exception plaso.lib.errors.**BadConfigObject**
Bases: *plaso.lib.errors.Error*

Raised when the configuration object is of the wrong type.

exception plaso.lib.errors.**BadConfigOption**
Bases: *plaso.lib.errors.Error*

Raised when a faulty configuration option is encountered.

exception plaso.lib.errors.**ConnectionError**
Bases: *plaso.lib.errors.Error*

Class that defines errors encountered connecting to a service.

exception plaso.lib.errors.**Error**
Bases: exceptions.Exception

Base error class.

exception plaso.lib.errors.**HeapFull**
Bases: *plaso.lib.errors.Error*

Class that implements a heap full exception.

exception plaso.lib.errors.**MalformedQueryError**
Bases: *plaso.lib.errors.Error*

Raised when an objectfilter query is malformed.

exception plaso.lib.errors.**MaximumRecursionDepth**
Bases: *plaso.lib.errors.Error*

Raised when the maximum recursion depth is reached.

exception plaso.lib.errors.**NoFormatterFound**
Bases: *plaso.lib.errors.Error*

Raised when no formatter is found for a particular event object.

exception plaso.lib.errors.**ParseError**
Bases: *plaso.lib.errors.Error*

Raised when a parse error occurred.

exception plaso.lib.errors.**PreProcessFail**
Bases: *plaso.lib.errors.Error*

Raised when a preprocess module is unable to gather information.

exception plaso.lib.errors.**QueueAlreadyClosed**
Bases: *plaso.lib.errors.Error*

Raised when an attempt is made to close a queue that is already closed.

exception plaso.lib.errors.**QueueAlreadyStarted**
Bases: *plaso.lib.errors.Error*

Raised when an attempt is made to start queue that is already started.

exception plaso.lib.errors.**QueueClose**
Bases: *plaso.lib.errors.Error*

Class that implements a queue close exception.

exception plaso.lib.errors.**QueueEmpty**
Bases: *plaso.lib.errors.Error*

Class that implements a queue empty exception.

exception plaso.lib.errors.**QueueFull**
Bases: *plaso.lib.errors.Error*

Class that implements a queue full exception.

exception plaso.lib.errors.**SerializationError**
Bases: *plaso.lib.errors.Error*

Class that defines serialization errors.

exception plaso.lib.errors.**SourceScannerError**
Bases: *plaso.lib.errors.Error*

Class that defines source scanner errors.

exception plaso.lib.errors.**TaggingFileError**
Bases: *plaso.lib.errors.Error*

Raised when the tagging file is invalid.

exception plaso.lib.errors.**TimestampError**
Bases: *plaso.lib.errors.Error*

Class that defines timestamp errors.

exception plaso.lib.errors.**UnableToLoadRegistryHelper**
Bases: *plaso.lib.errors.Error*

Raised when unable to load a Registry helper object.

exception plaso.lib.errors.**UnableToParseFile**
Bases: *plaso.lib.errors.Error*

Raised when a parser is not designed to parse a file.

exception plaso.lib.errors.**UserAbort**
Bases: *plaso.lib.errors.Error*

Class that defines an user initiated abort exception.

exception plaso.lib.errors.**WrongBencodePlugin**
Bases: *plaso.lib.errors.Error*

Error reporting wrong bencode plugin used.

exception plaso.lib.errors.**WrongFormatter**
Bases: *plaso.lib.errors.Error*

Raised when the formatter is not applicable for a particular event.

exception plaso.lib.errors.**WrongPlistPlugin**
Bases: *plaso.lib.errors.Error*

Error reporting wrong plist plugin used.

exception plaso.lib.errors.**WrongPlugin**
Bases: *plaso.lib.errors.Error*

Raised when the plugin is of the wrong type.

exception plaso.lib.errors.**WrongQueueType**
Bases: *plaso.lib.errors.Error*

Raised when an unsupported operation is attempted on a queue.

For example, attempting to Pop from a Push-only queue.

plaso.lib.lexer module

An LL(1) lexer. This lexer is very tolerant of errors and can resync.

This lexer is originally copied from the GRR project: https://code.google.com/p/grr

class plaso.lib.lexer.**BinaryExpression**(*operator=u”*, *part=None*)
Bases: *plaso.lib.lexer.Expression*

An expression which takes two other expressions.

AddOperands(*lhs*, *rhs*)
Add an operand.

Compile(*filter_implementation*)
Compile the binary expression into a filter object.

PrintTree(*depth=u”*)
Print the tree.

__str__()
Return a string representation of the binary expression.

class plaso.lib.lexer.**Expression**
Bases: object

A class representing an expression.

AddArg(*arg*)
Adds a new arg to this expression.

Parameters **arg** – The argument to add (string).

Returns True if this arg is the last arg, False otherwise.

Raises ParseError – If there are too many args.

Compile(*unused_filter_implementation*)
Given a filter implementation, compile this expression.

PrintTree(*depth=u”*)
Print the tree.

SetAttribute(*attribute*)
Set the attribute.

SetOperator(*operator*)
Set the operator.

__str__()
Return a string representation of the expression.

args = None

attribute = None

number_of_args = 1

operator = None

class plaso.lib.lexer.**IdentityExpression**
Bases: *plaso.lib.lexer.Expression*

An Expression which always evaluates to True.

Compile(*filter_implementation*)
Compile the expression.

class plaso.lib.lexer.**Lexer**(*data=u''*)
Bases: object

A generic feed lexer.

Close()
A convenience function to force us to parse all the data.

Default(***kwarg*)
The default callback handler.

Empty()
Returns a boolean indicating if the buffer is empty.

Error(*message=None*, *weight=1*)
Log an error down.

Parameters

- **message** – optional error message.
- **weight** – optional error weight.

Feed(*data*)
Feed the buffer with data.

Parameters **data** – data to be processed by the lexer.

NextToken()
Fetch the next token by trying to match any of the regexes in order.

PopState(***unused_kwargs*)
Pop the previous state from the stack.

PushBack(*string=u''*, ***unused_kwargs*)
Push the match back on the stream.

Parameters **string** – optional data.

PushState(***unused_kwargs*)
Push the current state on the state stack.

tokens = []

class plaso.lib.lexer.**SearchParser**(*data*)
Bases: *plaso.lib.lexer.Lexer*

This parser can parse the mini query language and build an AST.

Examples of valid syntax: filename contains "foo" and (size > 100k or date before "2011-10") date between 2011 and 2010 files older than 1 year

BinaryOperator(*string=None*, ***unused_kwargs*)
Set the binary operator.

BracketClose(***unused_kwargs*)
Close the bracket.

BracketOpen(***unused_kwargs*)
Define an open bracket.

Error(*message=None*, *unused_weight=1*)
Raise an error message.

InsertArg(*string=u”*, ***unused_kwargs*)
Insert an arg to the current expression.

Parse()
Parse.

Reduce()
Reduce the token stack into an AST.

StoreAttribute(*string=u”*, ***unused_kwargs*)
Store the attribute.

StoreOperator(*string=u”*, ***unused_kwargs*)
Store the operator.

StringEscape(*string*, *match*, ***unused_kwargs*)
Escape backslashes found inside a string quote.

Backslashes followed by anything other than [‘“rnbt] will just be included in the string.

Parameters

- **string** – The string that matched.
- **match** – the match object (instance of re.MatchObject). Where match.group(1) contains the escaped code.

StringFinish(***unused_kwargs*)
Finish the string operation.

StringInsert(*string=u”*, ***unused_kwargs*)
Add to the string.

StringStart(***unused_kwargs*)
Initialize the string.

binary_expression_cls
alias of *BinaryExpression*

expression_cls
alias of *Expression*

tokens = [<plaso.lib.lexer.Token object>, <plaso.lib.lexer.Token object>, <plaso.lib.l

class plaso.lib.lexer.**SelfFeederMixIn**(*file_object=None*)
Bases: *plaso.lib.lexer.Lexer*

This mixin is used to make a lexer which feeds itself.

Note that self.file_object must be the file object we read from.

Feed(*size=512*)
Feed data into the buffer.

Parameters **size** – optional data size to read form the file-like object.

NextToken()
Retrieves the next token.

Returns The next token (instance of Token) or None.

class plaso.lib.lexer.**Token**(*state_regex*, *regex*, *actions*, *next_state*, *flags=2*)
Bases: object

A token action.

plaso.lib.line_reader_file module

Binary line reader file-like object.

class plaso.lib.line_reader_file.**BinaryLineReader**(*file_object*, *end_of_line='n'*)
Bases: object

Line reader for binary file-like objects.

__enter__()
Enters a with statement.

__exit__(*unused_type*, *unused_value*, *unused_traceback*)
Exits a with statement.

__iter__()
Returns a line of text.

Yields *bytes* – line of text.

readline(*size=None*)
Reads a single line of text.

The functions reads one entire line from the file-like object. A trailing end-of-line indicator (newline by default) is kept in the byte string (but may be absent when a file ends with an incomplete line). An empty byte string is returned only when end-of-file is encountered immediately.

Parameters **size** (*Optional[int]*) – maximum byte size to read. If present and non-negative, it is a maximum byte count (including the trailing end-of-line) and an incomplete line may be returned.

Returns line of text.

Return type bytes

readlines(*sizehint=None*)
Reads lines of text.

The function reads until EOF using readline() and return a list containing the lines read.

Parameters **sizehint** (*Optional[int]*) – maximum byte size to read. If present, instead of reading up to EOF, whole lines totalling sizehint bytes are read.

Returns lines of text.

Return type list[bytes]

tell()
Retrieves the current offset into the file-like object.

Returns cuffent offset into the file-like object.

Return type int

plaso.lib.loggers module

Logging related classes and functions.

class plaso.lib.loggers.**CompressedFileHandler**(*filename*, *mode=u'a'*, *encoding=None*)
Bases: logging.FileHandler

Compressed file handler for logging.

emit(*record*)
Emits a record.

Parameters **record** (*logging.LogRecord*) – log record.

plaso.lib.loggers.**ConfigureLogging**(*debug_output=False*, *filename=None*, *mode=u'w'*, *quiet_mode=False*)
Configures the logging root logger.

Parameters

- **debug_output** (*Optional[bool]*) – True if the logging should include debug output.
- **filename** (*Optional[str]*) – log filename.
- **mode** (*Optional[str]*) – log file access mode.
- **quiet_mode** (*Optional[bool]*) – True if the logging should not include information output. Note that debug_output takes precedence over quiet_mode.

plaso.lib.objectfilter module

Classes to perform filtering of objects based on their data members.

Given a list of objects and a textual filter expression, these classes allow you to determine which objects match the filter. The system has two main pieces: A parser for the supported grammar and a filter implementation.

Given any complying user-supplied grammar, it is parsed with a custom lexer based on GRR's lexer and then compiled into an actual implementation by using the filter implementation. A filter implementation simply provides actual implementations for the primitives required to perform filtering. The compiled result is always a class supporting the Filter interface.

If we define a class called Car such as:

class Car(object):

def __init__(self, code, color="white", doors=3): self.code = code self.color = color self.doors = 3

And we have two instances:

ford_ka = Car("FORDKA1", color="grey") toyota_corolla = Car("COROLLA1", color="white", doors=5) fleet = [ford_ka, toyota_corolla]

We want to find cars that are grey and have 3 or more doors. We could filter our fleet like this:

criteria = "(color is grey) and (doors >= 3)" parser = ContextFilterParser(criteria).Parse() compiled_filter = parser.Compile(LowercaseAttributeFilterImp)

for car in fleet:

if compiled_filter.Matches(car): print("Car %s matches the supplied filter." % car.code)

The filter expression contains two subexpressions joined by an AND operator: “color is grey” and “doors >= 3”

This means we want to search for objects matching these two subexpressions. Let’s analyze the first one in depth “color is grey”:

> “color”: the left operand specifies a search path to look for the data. This tells our filtering system to look for the color property on passed objects. “is”: the operator. Values retrieved for the “color” property will be checked against the right operand to see if they are equal. “grey”: the right operand. It specifies an explicit value to check for.

So each time an object is passed through the filter, it will expand the value of the color data member, and compare its value against “grey”.

Because data members of objects are often not simple datatypes but other objects, the system allows you to reference data members within other data members by separating each by a dot. Let’s see an example:

Let’s add a more complex Car class with default tyre data:

class CarWithTyres(Car):

> **def __init__(self, code, tyres=None, color=”white”, doors=3):** super(self, CarWithTyres).__init__(code, color, doors) tyres = tyres or Tyre(“Pirelli”, “PZERO”)

class Tyre(object):

> **def __init__(self, brand, code):** self.brand = brand self.code = code

And two new instances: ford_ka = CarWithTyres(“FORDKA”, color=”grey”, tyres=Tyre(“AVON”, “ZT5”)) toyota_corolla = Car(“COROLLA1”, color=”white”, doors=5) fleet = [ford_ka, toyota_corolla]

To filter a car based on the tyre brand, we would use a search path of “tyres.brand”.

Because the filter implementation provides the actual classes that perform handling of the search paths, operators, etc. customizing the behaviour of the filter is easy. Three basic filter implementations are given:

> BaseFilterImplementation: search path expansion is done on attribute names as provided (case-sensitive). LowercaseAttributeFilterImp: search path expansion is done on the lowercased attribute name, so that it only accesses attributes, not methods. DictFilterImplementation: search path expansion is done on dictionary access to the given object. So “a.b” expands the object obj to obj[“a”][“b”]

class plaso.lib.objectfilter.**AndFilter**(*arguments=None*, *value_expander=None*)
Bases: *plaso.lib.objectfilter.Filter*

Performs a boolean AND of the given Filter instances as arguments.

Note that if no conditions are passed, all objects will pass.

Matches(*obj*)

class plaso.lib.objectfilter.**AttributeValueExpander**
Bases: *plaso.lib.objectfilter.ValueExpander*

An expander that gives values based on object attribute names.

class plaso.lib.objectfilter.**BaseFilterImplementation**
Bases: object

Defines the base implementation of an object filter by its attributes.

Inherit from this class, switch any of the needed operators and pass it to the Compile method of a parsed string to obtain an executable filter.

FILTERS = {u'AndFilter': <class 'plaso.lib.objectfilter.AndFilter'>, u'IdentityFilter

OPS = {u'>=': <class 'plaso.lib.objectfilter.GreaterEqual'>, u'==': <class 'plaso.li

class plaso.lib.objectfilter.**BasicExpression**
Bases: *plaso.lib.lexer.Expression*

Basic Expression.

Compile(*filter_implementation*)

FlipBool()

class plaso.lib.objectfilter.**BinaryExpression**(*operator=u"*, *part=None*)
Bases: *plaso.lib.lexer.BinaryExpression*

Compile(*filter_implementation*)
Compile the binary expression into a filter object.

class plaso.lib.objectfilter.**BinaryOperator**(*arguments=None*, ***kwargs*)
Bases: *plaso.lib.objectfilter.Operator*

Base class for binary operators.

The left operand is always a path into the object which will be expanded for values. The right operand is a value defined at initialization and is stored at self.right_operand.

class plaso.lib.objectfilter.**Contains**(***kwargs*)
Bases: *plaso.lib.objectfilter.GenericBinaryOperator*

Whether the right operand is contained in the value.

Operation(*x*, *y*)

class plaso.lib.objectfilter.**Context**(*arguments=None*, ***kwargs*)
Bases: *plaso.lib.objectfilter.Operator*

Restricts the child operators to a specific context within the object.

Solves the context problem. The context problem is the following: Suppose you store a list of loaded DLLs within a process. Suppose that for each of these DLLs you store the number of imported functions and each of the imported functions name.

Imagine that a malicious DLL is injected into processes and its indicators are that it only imports one function and that it is RegQueryValueEx. Yo'd write your indicator like this:

AndOperator(Equal("ImportedDLLs.ImpFunctions.Name", "RegQueryValueEx"), Equal("ImportedDLLs.NumImpFunctions", "1"))

Now imagine you have these two processes on a given system.

Process1 * __ImportedDlls

- __Name: "notevil.dll"
  - __ImpFunctions
    - __Name: "CreateFileA"
  - __NumImpFunctions: 1
- __Name: "alsonotevil.dll"
  - __ImpFunctions
    - __Name: "RegQueryValueEx"
    - __Name: "CreateFileA"
  - __NumImpFunctions: 2

Process2 * __ImportedDlls

- __Name: “evil.dll”
  - __ImpFunctions
    - __Name: “RegQueryValueEx”
  - __NumImpFunctions: 1

Both Process1 and Process2 match your query, as each of the indicators are evaluated separately. While you wanted to express “find me processes that have a DLL that has both one imported function and ReqQueryValueEx is in the list of imported functions”, your indicator actually means “find processes that have at least a DLL with 1 imported functions and at least one DLL that imports the ReqQueryValueEx function”.

To write such an indicator you need to specify a context of ImportedDLLs for these two clauses. Such that you convert your indicator to:

```
Context("ImportedDLLs",
        AndOperator(
          Equal("ImpFunctions.Name", "RegQueryValueEx"),
          Equal("NumImpFunctions", "1")
        ))
```

Context will execute the filter specified as the second parameter for each of the objects under “ImportedDLLs”, thus applying the condition per DLL, not per object and returning the right result.

Matches(*obj*)

class plaso.lib.objectfilter.**ContextExpression**(*attribute=u”*, *part=None*)
Bases: *plaso.lib.lexer.Expression*

Represents the context operator.

Compile(*filter_implementation*)
Compile the expression.

SetExpression(*expression*)
Set the expression.

class plaso.lib.objectfilter.**DictValueExpander**
Bases: *plaso.lib.objectfilter.ValueExpander*

An expander that gets values from dictionary access to the object.

class plaso.lib.objectfilter.**Equals**(***kwargs*)
Bases: *plaso.lib.objectfilter.GenericBinaryOperator*

Matches objects when the right operand equals the expanded value.

Operation(*x*, *y*)

class plaso.lib.objectfilter.**Filter**(*arguments=None*, *value_expander=None*)
Bases: object

Base class for every filter.

Filter(*objects*)
Returns a list of objects that pass the filter.

Matches(*obj*)
Whether object obj matches this filter.

class plaso.lib.objectfilter.**GenericBinaryOperator**(***kwargs*)
Bases: *plaso.lib.objectfilter.BinaryOperator*

Allows easy implementations of operators.

FlipBool()

Matches(*obj*)

Operate(*values*)
Takes a list of values and if at least one matches, returns True.

Operation(*x*, *y*)
Performs the operation between two values.

plaso.lib.objectfilter.**GetUnicodeString**(*string*)
Converts the string to Unicode if necessary.

class plaso.lib.objectfilter.**Greater**(***kwargs*)
Bases: *plaso.lib.objectfilter.GenericBinaryOperator*

Whether the expanded value > right_operand.

Operation(*x*, *y*)

class plaso.lib.objectfilter.**GreaterEqual**(***kwargs*)
Bases: *plaso.lib.objectfilter.GenericBinaryOperator*

Whether the expanded value >= right_operand.

Operation(*x*, *y*)

class plaso.lib.objectfilter.**IdentityFilter**(*arguments=None*, *value_expander=None*)
Bases: *plaso.lib.objectfilter.Operator*

Matches(*_*)

class plaso.lib.objectfilter.**InSet**(***kwargs*)
Bases: *plaso.lib.objectfilter.GenericBinaryOperator*

Whether all values are contained within the right operand.

Operation(*x*, *y*)
Whether x is fully contained in y.

exception plaso.lib.objectfilter.**InvalidNumberOfOperands**
Bases: *plaso.lib.errors.Error*

The number of operands provided to this operator is wrong.

class plaso.lib.objectfilter.**Less**(***kwargs*)
Bases: *plaso.lib.objectfilter.GenericBinaryOperator*

Whether the expanded value >= right_operand.

Operation(*x*, *y*)

class plaso.lib.objectfilter.**LessEqual**(***kwargs*)
Bases: *plaso.lib.objectfilter.GenericBinaryOperator*

Whether the expanded value <= right_operand.

Operation(*x*, *y*)

class plaso.lib.objectfilter.**LowercaseAttributeValueExpander**
Bases: *plaso.lib.objectfilter.AttributeValueExpander*

An expander that lowercases all attribute names before access.

class plaso.lib.objectfilter.**NotEquals**(***kwargs*)
Bases: *plaso.lib.objectfilter.Equals*

Matches when the right operand isn't equal to the expanded value.

class plaso.lib.objectfilter.**Operator**(*arguments=None*, *value_expander=None*)
Bases: *plaso.lib.objectfilter.Filter*

Base class for all operators.

class plaso.lib.objectfilter.**OrFilter**(*arguments=None*, *value_expander=None*)
Bases: *plaso.lib.objectfilter.Filter*

Performs a boolean OR of the given Filter instances as arguments.

Note that if no conditions are passed, all objects will pass.

Matches(*obj*)

class plaso.lib.objectfilter.**Parser**(*data*)
Bases: *plaso.lib.lexer.SearchParser*

Parses and generates an AST for a query written in the described language.

Examples of valid syntax: size is 40 (name contains "Program Files" AND hash.md5 is "123abc") @imported_modules (num_symbols = 14 AND symbol.name is "FindWindow")

ContextOperator(*string=u"*, ***unused_kwargs*)

Error(*message=None*, *_=None*)

FlipAllowed()
Raise an error if the not keyword is used where it is not allowed.

FlipLogic(***unused_kwargs*)
Flip the boolean logic of the expression.

If an expression is configured to return True when the condition is met this logic will flip that to False, and vice versa.

HexEscape(*string*, *match*, ***unused_kwargs*)
Converts a hex escaped string.

InsertArg(*string=u"*, ***unused_kwargs*)
Insert an arg to the current expression.

InsertFloatArg(*string=u"*, ***unused_kwargs*)
Inserts a Float argument.

InsertInt16Arg(*string=u"*, ***unused_kwargs*)
Inserts an Integer in base16 argument.

InsertIntArg(*string=u"*, ***unused_kwargs*)
Inserts an Integer argument.

Reduce()
Reduce the token stack into an AST.

StoreAttribute(*string=u"*, ***kwargs*)

StringEscape(*string*, *match*, ***unused_kwargs*)
Escape backslashes found inside a string quote.

Backslashes followed by anything other than ['"rnbt.ws] will raise an Error.

Parameters

- **string** – The string that matched.

- **match** – the match object (instance of re.MatchObject). Where match.group(1) contains the escaped code.

Raises ParseError – When the escaped string is not one of [‘“rnbt]

StringFinish(***unused_kwargs*)

binary_expression_cls
alias of *BinaryExpression*

context_cls
alias of *ContextExpression*

expression_cls
alias of *BasicExpression*

tokens = [<plaso.lib.lexer.Token object>, <plaso.lib.lexer.Token object>, <plaso.lib.l

class plaso.lib.objectfilter.**Regexp**(**children*, ***kwargs*)
Bases: *plaso.lib.objectfilter.GenericBinaryOperator*

Whether the value matches the regexp in the right operand.

Operation(*x*, *unused_y*)

class plaso.lib.objectfilter.**RegexpInsensitive**(**children*, ***kwargs*)
Bases: *plaso.lib.objectfilter.Regexp*

Whether the value matches the regexp in the right operand.

class plaso.lib.objectfilter.**UnaryOperator**(*operand*, ***kwargs*)
Bases: *plaso.lib.objectfilter.Operator*

Base class for unary operators.

class plaso.lib.objectfilter.**ValueExpander**
Bases: object

Encapsulates the logic to expand values available in an object.

Once instantiated and called, this class returns all the values that follow a given field path.

Expand(*obj*, *path*)
Returns a list of all the values for the given path in the object obj.

Given a path such as [“sub1”, “sub2”] it returns all the values available in obj.sub1.sub2 as a list. sub1 and sub2 must be data attributes or properties.

If sub1 returns a list of objects, or a generator, Expand aggregates the values for the remaining path for each of the objects, thus returning a list of all the values under the given path for the input object.

Parameters

- **obj** – An object that will be traversed for the given path
- **path** – A list of strings

Yields The values once the object is traversed.

FIELD_SEPARATOR = u'.'

plaso.lib.pfilter module

plaso.lib.plist module

The plist file object.

class plaso.lib.plist.**PlistFile**
Bases: object

Class that defines a plist file.

root_key
the plist root key (instance of plistlib._InternalDict).

GetValueByPath(*path_segments*)
Retrieves a plist value by path.

Parameters **path_segments** – a list of path segments strings relative from the root of the plist.

Returns The value of the key specified by the path or None.

Read(*file_object*)
Reads a plist from a file-like object.

Parameters **file_object** – the file-like object.

Raises IOError – if the plist file-like object cannot be read.

plaso.lib.py2to3 module

The Python 2 and 3 compatible type definitions.

plaso.lib.specification module

The format specification classes.

class plaso.lib.specification.**FormatSpecification**(*identifier*)
Bases: object

The format specification.

AddNewSignature(*pattern*, *offset=None*)
Adds a signature.

Parameters

- **pattern** (*bytes*) – pattern of the signature.
- **offset** (*int*) – offset of the signature. None is used to indicate the signature has no offset. A positive offset is relative from the start of the data a negative offset is relative from the end of the data.

class plaso.lib.specification.**FormatSpecificationStore**
Bases: object

The store for format specifications.

AddNewSpecification(*identifier*)
Adds a new format specification.

Parameters **identifier** (*str*) – format identifier, which should be unique for the store.

Returns format specification.

Return type *FormatSpecification*

Raises KeyError – if the store already contains a specification with the same identifier.

AddSpecification(*specification*)
Adds a format specification.

Parameters **specification** (FormatSpecification) – format specification.

Raises KeyError – if the store already contains a specification with the same identifier.

GetSpecificationBySignature(*signature_identifier*)
Retrieves a specification mapped to a signature identifier.

Parameters **identifier** (*str*) – unique signature identifier for a specification store.

Returns

format specification or None if the signature identifier does not exist within the specification store.

Return type *FormatSpecification*

specifications
iterator – specifications iterator.

class plaso.lib.specification.**Signature**(*pattern*, *offset=None*)
Bases: object

The format specification signature.

The signature consists of a byte string pattern, an optional offset relative to the start of the data, and a value to indicate if the pattern is bound to the offset.

SetIdentifier(*identifier*)
Sets the identifier of the signature in the specification store.

Parameters **identifier** (*str*) – unique signature identifier for a specification store.

plaso.lib.timelib module

Time manipulation functions and variables.

This module contain common methods that can be used to convert timestamps from various formats into number of micro seconds since January 1, 1970, 00:00:00 UTC that is used internally to store timestamps.

It also contains various functions to represent timestamps in a more human readable form.

plaso.lib.timelib.**GetCurrentYear**()
Determines the current year.

plaso.lib.timelib.**GetYearFromPosixTime**(*posix_time*, *timezone=<Mock id='140451374027984'>*)
Gets the year from a POSIX timestamp

The POSIX time is the number of seconds since 1970-01-01 00:00:00 UTC.

Parameters

- **posix_time** – An integer containing the number of seconds since 1970-01-01 00:00:00 UTC.

- **timezone** – Optional timezone of the POSIX timestamp.

Returns The year of the POSIX timestamp.

Raises `ValueError` – If the posix timestamp is out of the range of supported values.

class plaso.lib.timelib.**Timestamp**
Bases: `object`

Class for converting timestamps to Plaso timestamps.

The Plaso timestamp is a 64-bit signed timestamp value containing: micro seconds since 1970-01-01 00:00:00.

The timestamp is not necessarily in UTC.

classmethod CopyFromString(*time_string*)
Copies a timestamp from a string containing a date and time value.

Parameters **time_string** – A string containing a date and time value formatted as: YYYY-MM-DD hh:mm:ss.######[+-]##:## Where # are numeric digits ranging from 0 to 9 and the seconds fraction can be either 3 or 6 digits. The time of day, seconds fraction and timezone offset are optional. The default timezone is UTC.

Returns The timestamp which is an integer containing the number of micro seconds since January 1, 1970, 00:00:00 UTC.

Raises `ValueError` – if the time string is invalid or not supported.

classmethod CopyToDatetime(*timestamp*, *timezone*, *raise_error=False*)
Copies the timestamp to a datetime object.

Parameters

- **timestamp** – The timestamp which is an integer containing the number of micro seconds since January 1, 1970, 00:00:00 UTC.
- **timezone** – The timezone (pytz.timezone) object.
- **raise_error** – Boolean that if set to True will not absorb an OverflowError if the timestamp is out of bounds. By default there will be no error raised.

Returns A datetime object (instance of datetime.datetime). A datetime object of January 1, 1970 00:00:00 UTC is returned on error if raises_error is not set.

Raises

- `OverflowError` – If raises_error is set to True and an overflow error occurs.
- `ValueError` – If raises_error is set to True and no timestamp value is provided.

classmethod CopyToIsoFormat(*timestamp*, *timezone=<Mock id='140451374027856'>*, *raise_error=False*)
Copies the timestamp to an ISO 8601 formatted string.

Parameters

- **timestamp** – The timestamp which is an integer containing the number of micro seconds since January 1, 1970, 00:00:00 UTC.
- **timezone** – Optional timezone (instance of pytz.timezone).
- **raise_error** – Boolean that if set to True will not absorb an OverflowError if the timestamp is out of bounds. By default there will be no error raised.

Returns A string containing an ISO 8601 formatted date and time.

classmethod CopyToPosix(*timestamp*)
Converts microsecond timestamps to POSIX timestamps.

Parameters **timestamp** – The timestamp which is an integer containing the number of micro seconds since January 1, 1970, 00:00:00 UTC.

Returns The timestamp which is an integer containing the number of seconds since January 1, 1970, 00:00:00 UTC.

classmethod FromPosixTime(*posix_time*)
Converts a POSIX timestamp into a timestamp.

The POSIX time is a signed 32-bit or 64-bit value containing: seconds since 1970-01-01 00:00:00

Parameters **posix_time** – The POSIX timestamp.

Returns The timestamp which is an integer containing the number of micro seconds since January 1, 1970, 00:00:00 UTC or 0 on error.

classmethod FromPythonDatetime(*datetime_object*)
Converts a Python datetime object into a timestamp.

Parameters **datetime_object** – The datetime object (instance of datetime.datetime).

Returns The timestamp which is an integer containing the number of micro seconds since January 1, 1970, 00:00:00 UTC or 0 on error.

classmethod FromTimeString(*time_string*, *dayfirst=False*, *gmt_as_timezone=True*, *timezone=<Mock id='140451374027920'>*)
Converts a string containing a date and time value into a timestamp.

Parameters

- **time_string** – String that contains a date and time value.
- **dayfirst** – An optional boolean argument. If set to true then the parser will change the precedence in which it parses timestamps from MM-DD-YYYY to DD-MM-YYYY (and YYYY-MM-DD will be YYYY-DD-MM, etc).
- **gmt_as_timezone** – Sometimes the dateutil parser will interpret GMT and UTC the same way, that is not make a distinction. By default this is set to true, that is GMT can be interpreted differently than UTC. If that is not the expected result this attribute can be set to false.
- **timezone** – Optional timezone object (instance of pytz.timezone) that the data and time value in the string represents. This value is used when the timezone cannot be determined from the string.

Returns The timestamp which is an integer containing the number of micro seconds since January 1, 1970, 00:00:00 UTC or 0 on error.

Raises `TimestampError` – if the time string could not be parsed.

classmethod GetNow()
Retrieves the current time (now) as a timestamp in UTC.

Returns The timestamp which is an integer containing the number of micro seconds since January 1, 1970, 00:00:00 UTC.

classmethod LocaltimeToUTC(*timestamp*, *timezone*, *is_dst=False*)
Converts the timestamp in localtime of the timezone to UTC.

Parameters

- **timestamp** – The timestamp which is an integer containing the number of micro seconds since January 1, 1970, 00:00:00 UTC.
- **timezone** – The timezone (pytz.timezone) object.
- **is_dst** – A boolean to indicate the timestamp is corrected for daylight savings time (DST) only used for the DST transition period.

Returns The timestamp which is an integer containing the number of micro seconds since January 1, 1970, 00:00:00 UTC or 0 on error.

MICROSECONDS_PER_MINUTE = 60000000

MICRO_SECONDS_PER_SECOND = 1000000

MILLI_SECONDS_TO_MICRO_SECONDS = 1000

NONE_TIMESTAMP = 0

classmethod RoundToSeconds(*timestamp*)
Takes a timestamp value and rounds it to a second precision.

TIMESTAMP_MAX_MICRO_SECONDS = 9223372036854775807L

TIMESTAMP_MAX_SECONDS = 9223372036854L

TIMESTAMP_MIN_MICRO_SECONDS = -9223372036854775807L

TIMESTAMP_MIN_SECONDS = -9223372036854L

plaso.lib.utils module

This file contains utility functions.

plaso.lib.utils.**IsText**(*bytes_in*, *encoding=None*)
Examine the bytes in and determine if they are indicative of a text.

Parsers need quick and at least semi reliable method of discovering whether or not a particular byte stream is a text or resembles text or not. This can be used in text parsers to determine if a file is a text file or not for instance.

The method assumes the byte sequence is either ASCII, UTF-8, UTF-16 or method supplied character encoding. Otherwise it will make the assumption the byte sequence is not text, but a byte sequence.

Parameters

- **bytes_in** (*bytes*) – byte stream to examine.
- **encoding** (*Optional[str]*) – encoding to test, if not defined ASCII and UTF-8 are tried.

Returns True if the bytes stream contains text.

Return type bool

Module contents

plaso.multi_processing package

Submodules

plaso.multi_processing.analysis_process module

The multi-process analysis process.

class plaso.multi_processing.analysis_process.**AnalysisProcess**(*event_queue*, *storage_writer*, *knowledge_base*, *analysis_plugin*, *data_location=None*, *event_filter_expression=None*, ***kwargs*)

Bases: *plaso.multi_processing.base_process.MultiProcessBaseProcess*

Multi-processing analysis process.

SignalAbort()
Signals the process to abort.

plaso.multi_processing.base_process module

Base class for a process used in multi-processing.

class plaso.multi_processing.base_process.**MultiProcessBaseProcess**(*enable_sigsegv_handler=False*, ***kwargs*)

Bases: multiprocessing.process.Process

Class that defines the multi-processing process interface.

rpc_port
int – port number of the process status RPC server.

SignalAbort()
Signals the process to abort.

name
str – process name.

run()
Runs the process.

plaso.multi_processing.engine module

plaso.multi_processing.multi_process_queue module

A multiprocessing-backed queue.

class plaso.multi_processing.multi_process_queue.**MultiProcessingQueue**(*maximum_number_of_queued*, *time-out=None*)

Bases: *plaso.engine.plaso_queue.Queue*

Multi-processing queue.

Close(*abort=False*)
Closes the queue.

This needs to be called from any process or thread putting items onto the queue.

Parameters **abort** (*Optional[bool]*) – True if the close was issued on abort.

Empty()
Empties the queue.

IsEmpty()
Determines if the queue is empty.

Open()
Opens the queue.

PopItem()
Pops an item off the queue.

Raises

- QueueClose – if the queue has already been closed.
- QueueEmpty – if no item could be retrieved from the queue within the specified timeout.

PushItem(*item*, *block=True*)
Pushes an item onto the queue.

Parameters

- **item** (*object*) – item to add.
- **block** (*Optional[bool]*) – True to block the process when the queue is full.

Raises QueueFull – if the item could not be pushed the queue because it's full.

plaso.multi_processing.plaso_xmlrpc module

XML RPC proxy server and client.

class plaso.multi_processing.plaso_xmlrpc.**ThreadedXMLRPCServer**(*callback*)
Bases: *plaso.multi_processing.rpc.RPCServer*

Class that defines the threaded XML RPC server.

Start(*hostname*, *port*)
Starts the process status RPC server.

Parameters

- **hostname** – the hostname or IP address to connect to for requests.
- **port** – the port to connect to for requests.

Returns A boolean indicating if the RPC server was successfully started.

Stop()
Stops the process status RPC server.

class plaso.multi_processing.plaso_xmlrpc.**XMLProcessStatusRPCClient**
Bases: *plaso.multi_processing.plaso_xmlrpc.XMLRPCClient*

Class that defines a XML process status RPC client.

class plaso.multi_processing.plaso_xmlrpc.**XMLProcessStatusRPCServer**(*callback*)
Bases: *plaso.multi_processing.plaso_xmlrpc.ThreadedXMLRPCServer*

Class that defines a XML process status RPC server.

class plaso.multi_processing.plaso_xmlrpc.**XMLRPCClient**
Bases: *plaso.multi_processing.rpc.RPCClient*

Class that defines the XML RPC client.

CallFunction()
Calls the function via RPC.

Close()
Closes the RPC communication channel to the server.

Open(*hostname*, *port*)
Opens a RPC communication channel to the server.

Parameters

- **hostname** – the hostname or IP address to connect to for requests.
- **port** – the port to connect to for requests.

Returns A boolean indicating if the communication channel was established.

plaso.multi_processing.psort module

plaso.multi_processing.rpc module

The RPC client and server interface.

class plaso.multi_processing.rpc.**RPCClient**
Bases: object

RPC client interface.

CallFunction()
Calls the function via RPC.

Close()
Closes the RPC communication channel to the server.

Open(*hostname*, *port*)
Opens a RPC communication channel to the server.

Parameters

- **hostname** (*str*) – hostname or IP address to connect to for requests.
- **port** (*int*) – port to connect to for requests.

Returns True if the communication channel was established.

Return type bool

class plaso.multi_processing.rpc.**RPCServer**(*callback*)
Bases: object

RPC server interface.

Start(*hostname*, *port*)
Starts the RPC server.

Parameters

- **hostname** (*str*) – hostname or IP address to connect to for requests.
- **port** (*int*) – port to connect to for requests.

Returns True if the RPC server was successfully started.

Return type bool

Stop()
Stops the RPC server.

plaso.multi_processing.task_engine module

plaso.multi_processing.task_manager module

The task manager.

class plaso.multi_processing.task_manager.**TaskManager**
Bases: object

Manages tasks and tracks their completion and status.

A task being tracked by the manager must be in exactly one of the following states:

- **abandoned: no status information has been recently received from a worker** about the task, and is assumed to be abandoned.
- **queued: the task is waiting for a worker to start processing it. It's also** possible that a worker has already completed the task, but no status update was collected from the worker while it processed the task.
- processing: a worker is processing the task.
- **pending_merge: a worker has completed processing the task and the** results are ready to be merged with the session storage.
- merging: tasks that are being merged by the engine.

Once the engine reports that a task is completely merged, it is removed from the task manager.

Tasks that are not abandoned, or abandoned, but need to be retried are considered "pending", as there is more work that needs to be done to complete them.

CompleteTask(*task*)
Completes a task.

The task is complete and can be removed from the task manager.

Parameters **task** (Task) – task.

CreateTask(*session_identifier*)
Creates a task.

Parameters **session_identifier** (*str*) – the identifier of the session the task is part of.

Returns task attribute container.

Return type *Task*

GetAbandonedTasks()
Retrieves all abandoned tasks.

Returns tasks.

Return type list[*Task*]

GetRetryTask()
Creates a task that is an attempt to retry an abandoned task.

Returns

a task that is a retry of an existing task, or None if there are no tasks that need to be retried.

Return type *Task*

GetStatusInformation()
Retrieves status information about the tasks.

Returns tasks status information.

Return type *TasksStatus*

GetTaskPendingMerge(*current_task*)
Retrieves the first task that is pending merge or has a higher priority.

This function will check if there is a task with a higher merge priority than the current_task being merged. If so, that task with the higher priority is returned.

Parameters **current_task** (Task) – current task being merged or None if no such task.

Returns

the next task to merge or None if there is no task pending merge or with a higher priority.

Return type *Task*

GetTasksCheckMerge()
Retrieves the tasks that need to be checked if they are ready for merge.

Returns

tasks that are being processed by workers or that have been abandoned.

Return type list[*Task*]

HasPendingTasks()
Determines if there are tasks running, or in need of retrying.

Returns

True if there are tasks that are active, ready to be merged, or need to be retried.

Return type bool

UpdateTaskAsPendingMerge(*task*)
Updates the task manager to reflect the task is ready to be merged.

Parameters **task** (Task) – task.

Raises KeyError – if the task was not processing or abandoned.

UpdateTaskAsProcessingByIdentifier(*task_identifier*)
Updates the task manager to reflect the task is processing.

Parameters **task_identifier** (*str*) – unique identifier of the task.

Raises KeyError – if the task is not known to the task manager.

plaso.multi_processing.worker_process module

Module contents

plaso.output package

Submodules

plaso.output.dynamic module

Contains a formatter for a dynamic output module for plaso.

class plaso.output.dynamic.**DynamicFieldsHelper**(*output_mediator*)
Bases: object

Helper for outputting a dynamic selection of fields.

GetFormattedField(*event*, *field_name*)
Formats the specified field.

Parameters

- **event** (EventObject) – event.
- **field_name** (*str*) – name of the field.

Returns value of the field.

Return type str

class plaso.output.dynamic.**DynamicOutputModule**(*output_mediator*)
Bases: *plaso.output.interface.LinearOutputModule*

Dynamic selection of fields for a separated value output format.

DESCRIPTION = u'Dynamic selection of fields for a separated value output format.'

NAME = u'dynamic'

SetFieldDelimiter(*field_delimiter*)
Sets the field delimiter.

Parameters **field_delimiter** (*str*) – field delimiter.

SetFields(*fields*)
Sets the fields to output.

Parameters **fields** (*list[str]*) – names of the fields to output.

WriteEventBody(*event*)
Writes the body of an event to the output.

Parameters **event** (EventObject) – event.

WriteHeader()
Writes the header to the output.

plaso.output.elastic module

An output module that saves events to Elasticsearch.

class plaso.output.elastic.**ElasticSearchHelper**(*output_mediator*, *host*, *port*, *flush_interval*, *index_name*, *mapping*, *doc_type*, *elastic_password=None*, *elastic_user=None*)

Bases: object

Elasticsearch helper class.

AddEvent(*event_object*, *force_flush=False*)

Index event in Elasticsearch.

Parameters

- **event_object** (EventObject) – the event object.
- **force_flush** (*bool*) – Force bulk insert of events in the queue.

class plaso.output.elastic.**ElasticSearchOutputModule**(*output_mediator*)

Bases: *plaso.output.interface.OutputModule*

Output module for Elasticsearch.

Close()

Close connection to the Elasticsearch database.

Sends any remaining buffered events for indexing.

DESCRIPTION = u'Saves the events into an Elasticsearch database.'

NAME = u'elastic'

SetDocType(*doc_type*)

Set the port.

Parameters **doc_type** (*str*) – The document type to use when indexing.

SetElasticPassword(*elastic_password*)

Set the Elastic password.

Parameters **elastic_password** (*str*) – Elastic password to authenticate with.

SetElasticUser(*elastic_user*)

Set the Elastic username.

Parameters **elastic_user** (*str*) – Elastic user to authenticate with.

SetFlushInterval(*flush_interval*)

Set the flush interval.

Parameters **flush_interval** (*int*) – Number of events to buffer before bulk insert.

SetIndexName(*index_name*)

Set the index name.

Parameters **index_name** – the index name.

SetRawFields(*raw_fields*)

Set raw (not analyzed) fields.

This is used for sorting and aggregations in Elasticsearch. https://www.elastic.co/guide/en/elasticsearch/guide/current/ multi-fields.html

Parameters **raw_fields** (*bool*) – Add not-analyzed index for string fields.

SetServerInformation(*server*, *port*)

Set the Elasticsearch server information.

Parameters

- **server** (*str*) – IP address or hostname of the server.
- **port** (*int*) – Port number of the server.

WriteEventBody(*event*)
Writes the body of an event to the output.

Parameters **event** (EventObject) – event.

WriteHeader()
Setup the Elasticsearch index.

plaso.output.interface module

This file contains the output module interface classes.

class plaso.output.interface.**LinearOutputModule**(*output_mediator*)
Bases: *plaso.output.interface.OutputModule*

Linear output module.

Close()
Closes the output.

SetOutputWriter(*output_writer*)
Set the output writer.

Parameters **output_writer** (CLIOutputWriter) – output writer.

class plaso.output.interface.**OutputModule**(*output_mediator*)
Bases: object

Output module interface.

Close()
Closes the output.

DESCRIPTION = u''

GetMissingArguments()
Retrieves arguments required by the module that have not been specified.

Returns

names of argument that are required by the module and have not been specified.

Return type list[str]

NAME = u''

Open()
Opens the output.

WriteEvent(*event*)
Writes the event to the output.

Parameters **event** (EventObject) – event.

WriteEventBody(*event*)
Writes the body of an event to the output.

Parameters **event** (EventObject) – event.

WriteEventEnd()
Writes the end of an event to the output.

Can be used for post-processing or output after an individual event has been written, such as writing closing XML tags, etc.

WriteEventMACBGroup(*event_macb_group*)
Writes an event MACB group to the output.

An event MACB group is a group of events that have the same timestamp and event data (attributes and values), where the timestamp description (or usage) is one or more of MACB (modification, access, change, birth).

This function is called if the psort engine detected an event MACB group so that the output module, if supported, can represent the group as such. If not overridden this function will output every event individually.

Parameters **event_macb_group** (*list[EventObject]*) – group of events with identical timestamps, attributes and values.

WriteEventStart()
Writes the start of an event to the output.

Can be used for pre-processing or output before an individual event has been written, such as writing opening XML tags, etc.

WriteFooter()
Writes the footer to the output.

Can be used for post-processing or output after the last event is written, such as writing a file footer.

WriteHeader()
Writes the header to the output.

Can be used for pre-processing or output before the first event is written, such as writing a file header.

plaso.output.json_line module

Output module that saves data into a JSON line format.

JSON line format is a single JSON entry or event per line instead of grouping all the output into a single JSON entity.

class plaso.output.json_line.**JSONLineOutputModule**(*output_mediator*)
Bases: *plaso.output.interface.LinearOutputModule*

Output module for the JSON line format.

DESCRIPTION = u'Saves the events into a JSON line format.'

NAME = u'json_line'

WriteEventBody(*event*)
Writes the body of an event object to the output.

Parameters **event** (EventObject) – event.

plaso.output.json_out module

Output module that saves data into a JSON format.

class plaso.output.json_out.**JSONOutputModule**(*output_mediator*)
Bases: *plaso.output.interface.LinearOutputModule*

Output module for the JSON format.

DESCRIPTION = u'Saves the events into a JSON format.'

NAME = u'json'

WriteEventBody(*event*)
Writes the body of an event object to the output.

Parameters **event** (EventObject) – event.

WriteFooter()
Writes the footer to the output.

WriteHeader()
Writes the header to the output.

plaso.output.kml module

An output module that writes event with geography data to a KML XML file.

The Keyhole Markup Language (KML) is an XML notation for expressing geographic annotation and visualization within Internet-based, two-dimensional maps and three-dimensional Earth browsers.

class plaso.output.kml.**KMLOutputModule**(*output_mediator*)
Bases: *plaso.output.interface.LinearOutputModule*

Output module for a Keyhole Markup Language (KML) XML file.

DESCRIPTION = u'Saves events with geography data into a KML format.'

NAME = u'kml'

WriteEventBody(*event*)
Writes the body of an event to the output.

Parameters **event** (EventObject) – event.

WriteFooter()
Writes the footer to the output.

WriteHeader()
Writes the header to the output.

plaso.output.l2t_csv module

Output module for the log2timeline (L2T) CSV format.

For documentation on the L2T CSV format see: http://forensicswiki.org/wiki/L2T_CSV

class plaso.output.l2t_csv.**L2TCSVOutputModule**(*output_mediator*)
Bases: *plaso.output.interface.LinearOutputModule*

CSV format used by log2timeline, with 17 fixed fields.

DESCRIPTION = u'CSV format used by legacy log2timeline, with 17 fixed fields.'

NAME = u'l2tcsv'

WriteEventBody(*event*)
Writes the body of an event object to the output.

Parameters **event** (EventObject) – event.

Raises NoFormatterFound – If no event formatter can be found to match the data type in the event object.

WriteEventMACBGroup(*event_macb_group*)
Writes an event MACB group to the output.

Parameters **event_macb_group** (*list[EventObject]*) – event MACB group.

WriteHeader()
Writes the header to the output.

plaso.output.manager module

Output plugin manager.

class plaso.output.manager.**OutputManager**
Bases: object

Output module manager.

classmethod DeregisterOutput(*output_class*)
Deregisters an output class.

The output classes are identified based on their NAME attribute.

Parameters **output_class** (*type*) – output module class.

Raises KeyError – if output class is not set for the corresponding data type.

classmethod GetDisabledOutputClasses()
Retrieves the disabled output classes and its associated name.

Yields *tuple[str, type]* – output module name and class.

classmethod GetOutputClass(*name*)
Retrieves the output class for a specific name.

Parameters **name** (*str*) – name of the output module.

Returns output module class.

Return type type

Raises

- KeyError – if there is no output class found with the supplied name.
- ValueError – if name is not a string.

classmethod GetOutputClasses()
Retrieves the available output classes its associated name.

Yields *tuple[str, type]* – output class name and type object.

classmethod HasOutputClass(*name*)
Determines if a specific output class is registered with the manager.

Parameters **name** (*str*) – name of the output module.

Returns True if the output class is registered.

Return type bool

classmethod IsLinearOutputModule(*name*)
Determines if a specific output class is a linear output module.

Parameters **name** (*str*) – name of the output module.

Returns if the output module is linear.

Return type True

classmethod NewOutputModule(*name*, *output_mediator*)
Creates a new output module object for the specified output format.

Parameters

- **name** (*str*) – name of the output module.
- **output_mediator** (OutputMediator) – output mediator.

Returns output module.

Return type *OutputModule*

Raises

- KeyError – if there is no output class found with the supplied name.
- ValueError – if name is not a string.

classmethod RegisterOutput(*output_class*, *disabled=False*)
Registers an output class.

The output classes are identified based on their NAME attribute.

Parameters

- **output_class** (*type*) – output module class.
- **disabled** (*Optional[bool]*) – True if the output module is disabled due to the module not loading correctly or not.

Raises KeyError – if output class is already set for the corresponding name.

classmethod RegisterOutputs(*output_classes*, *disabled=False*)
Registers output classes.

The output classes are identified based on their NAME attribute.

Parameters

- **output_classes** (*list[type]*) – output module classes.
- **disabled** (*Optional[bool]*) – True if the output module is disabled due to the module not loading correctly or not.

Raises KeyError – if output class is already set for the corresponding name.

plaso.output.mediator module

The output mediator object.

class plaso.output.mediator.**OutputMediator**(*knowledge_base*, *formatter_mediator*, *fields_filter=None*, *preferred_encoding=u'utf-8'*)

Bases: object

Output mediator.

fields_filter
FilterObject – filter object that indicates which fields to output.

GetEventFormatter(*event*)
Retrieves the event formatter for a specific event type.

Parameters **event** (EventObject) – event.

Returns event formatter or None.

Return type *EventFormatter*

GetFormatStringAttributeNames(*event*)
Retrieves the attribute names in the format string.

Parameters **event** (EventObject) – event.

Returns A list containing the attribute names. If no event formatter to match the event can be found the function returns None.

GetFormattedMessages(*event*)
Retrieves the formatted messages related to the event.

Parameters **event** (EventObject) – event.

Returns A tuple containing the formatted message string and short message string. If no event formatter to match the event can be found the function returns a tuple of None, None.

GetFormattedSources(*event*)
Retrieves the formatted sources related to the event.

Parameters **event** (EventObject) – event.

Returns A tuple of the short and long source string. If no event formatter to match the event can be found the function returns a tuple of None, None.

GetHostname(*event*, *default_hostname=u'-'*)
Retrieves the hostname related to the event.

Parameters

- **event** (EventObject) – event.
- **default_hostname** (*Optional[str]*) – default hostname.

Returns hostname.

Return type str

GetMACBRepresentation(*event*)
Retrieves the MACB representation.

Parameters **event** (EventObject) – event.

Returns MACB representation.

Return type str

GetMACBRepresentationFromDescriptions(*timestamp_descriptions*)
Determines the MACB representation from the timestamp descriptions.

MACB representation is a shorthand for representing one or more of modification, access, change, birth timestamp descriptions as the letters "MACB" or a "." if the corresponding timestamp is not set.

Note that this is an output format shorthand and does not guarantee that the timestamps represent the same occurrence.

Parameters **timestamp_descriptions** (*list[str]*) – timestamp descriptions, which are defined in definitions.TIME_DESCRIPTIONS.

Returns MACB representation.

Return type str

GetStoredHostname()
Retrieves the stored hostname.

Returns hostname.

Return type str

GetUsername(*event*, *default_username=u'-'*)
Retrieves the username related to the event.

Parameters

- **event** (EventObject) – event.
- **default_username** (*Optional[str]*) – default username.

Returns username.

Return type str

SetTimezone(*timezone*)
Sets the timezone.

Parameters **timezone** (*str*) – timezone.

Raises ValueError – if the timezone is not supported.

encoding
str – preferred encoding.

filter_expression
str – filter expression if a filter is set, None otherwise.

timezone
The timezone.

plaso.output.mysql_4n6time module

Defines the output module for the MySQL database used by 4n6time.

class plaso.output.mysql_4n6time.**MySQL4n6TimeOutputModule**(*output_mediator*)
Bases: *plaso.output.shared_4n6time.Shared4n6TimeOutputModule*

Class defining the MySQL database output module for 4n6time.

Close()
Disconnects from the database.

This method will create the necessary indices and commit outstanding transactions before disconnecting.

DESCRIPTION = u'MySQL database output for the 4n6time tool.'

NAME = u'4n6time_mysql'

Open()
Connects to the database and creates the required tables.

Raises

- IOError – If Unable to insert into database.
- ValueError – If no database name given.

SetCredentials(*password=None*, *username=None*)
Sets the database credentials.

Parameters

- **password** (*Optional[str]*) – password to access the database.
- **username** (*Optional[str]*) – username to access the database.

SetDatabaseName(*name*)
Sets the database name.

Parameters **name** (*str*) – name of the database.

SetServerInformation(*server*, *port*)
Sets the server information.

Parameters

- **server** (*str*) – hostname or IP address of the database server.
- **port** (*int*) – port number of the database server.

WriteEventBody(*event*)
Writes the body of an event object to the output.

Parameters **event** (EventObject) – event.

plaso.output.null module

Null device output module.

class plaso.output.null.**NullOutputModule**(*output_mediator*)
Bases: *plaso.output.interface.OutputModule*

Null device output module.

DESCRIPTION = u'Output module that does not output anything.'

NAME = u'null'

WriteEventBody(*unused_event_object*)
Writes the event object to the output.

Since this is the null output module nothing is actually written.

Parameters **event_object** (EventObject) – event object.

plaso.output.rawpy module

Output module for the "raw" (or native) Python format.

class plaso.output.rawpy.**NativePythonFormatterHelper**
Bases: object

Helper for outputting as "raw" (or native) Python.

classmethod GetFormattedEventObject(*event_object*)
Retrieves a string representation of the event object.

Returns A Unicode string containing the string representation of the event object.

class plaso.output.rawpy.**NativePythonOutputModule**(*output_mediator*)
Bases: *plaso.output.interface.LinearOutputModule*

Output module for the "raw" (or native) Python output format.

DESCRIPTION = u'"raw" (or native) Python output.'

NAME = u'rawpy'

WriteEventBody(*event*)
Writes the body of an event to the output.

Parameters **event** (EventObject) – event.

plaso.output.shared_4n6time module

Defines the shared code for 4n6time output modules.

class plaso.output.shared_4n6time.**Shared4n6TimeOutputModule**(*output_mediator*)
Bases: *plaso.output.interface.OutputModule*

Class defining the base 4n6time output module.

NAME = u'4n6time_shared'

SetAppendMode(*append*)
Set the append status.

Parameters **append** (*bool*) – True if the events should be added to the database.

SetEvidence(*evidence*)
Set the evidence field.

Parameters **evidence** (*str*) – the evidence field.

SetFields(*fields*)
Set the fields that will be indexed in the database.

Parameters **fields** (*list[str]*) – a list of fields that should be indexed.

SetStatusObject(*status_object*)
Set the status object.

Parameters **status_object** – status object provided by the 4n6time tool.

plaso.output.sqlite_4n6time module

Defines the output module for the SQLite database used by 4n6time.

class plaso.output.sqlite_4n6time.**SQLite4n6TimeOutputModule**(*output_mediator*)
Bases: *plaso.output.shared_4n6time.Shared4n6TimeOutputModule*

Saves the data in a SQLite database, used by the tool 4n6time.

Close()
Disconnects from the database.

This method will create the necessary indices and commit outstanding transactions before disconnecting.

DESCRIPTION = u'Saves the data in a SQLite database, used by the tool 4n6time.'

NAME = u'4n6time_sqlite'

Open()
Connects to the database and creates the required tables.

Raises

- IOError – if the specified output file already exists.
- ValueError – if the filename is not set.

SetFilename(*filename*)
Sets the filename.

Parameters **filename** (*str*) – the filename.

WriteEventBody(*event*)
Writes the body of an event to the output.

Parameters **event** (EventObject) – event.

plaso.output.timesketch_out module

Timesketch output module.

class plaso.output.timesketch_out.**TimesketchOutputModule**(*output_mediator*)
Bases: *plaso.output.interface.OutputModule*

Output module for Timesketch.

Close()
Closes the connection to TimeSketch Elasticsearch database.

Sends the remaining events for indexing and removes the processing status on the Timesketch search index object.

DESCRIPTION = u'Create a Timesketch timeline.'

GetMissingArguments()
Return a list of arguments that are missing from the input.

Returns

names of arguments that are required by the module and have not been specified.

Return type list[str]

NAME = u'timesketch'

SetDocType(*doc_type*)
Sets the Elasticsearch document type.

Parameters **doc_type** (*str*) – document type.

SetFlushInterval(*flush_interval*)
Sets the flush interval.

Parameters **flush_interval** (*int*) – flush interval.

SetIndexName(*index_name*)
Sets the index name.

Parameters **index_name** (*str*) – index name.

SetTimelineName(*timeline_name*)
Sets the timeline name.

Parameters **timeline_name** (*str*) – timeline name.

SetUserName(*username*)
Sets the username of the user that should own the timeline.

Parameters **username** (*str*) – username.

WriteEventBody(*event*)
Writes the body of an event to the output.

Parameters **event** (EventObject) – event.

WriteHeader()
Setup the Elasticsearch index and the Timesketch database object.

Creates the Elasticsearch index with Timesketch specific settings and the Timesketch SearchIndex database object.

plaso.output.tln module

Output module for the TLN format.

For documentation on the TLN format see: http://forensicswiki.org/wiki/TLN

class plaso.output.tln.**L2TTLNOutputModule**(*output_mediator*)
Bases: *plaso.output.tln.TLNBaseOutputModule*

Output module for the log2timeline extended variant of the TLN format.

l2tTLN is an extended variant of TLN introduced log2timeline 0.65.

l2tTLN extends basic TLN to 7 | separated fields, namely: * Time - 32-bit POSIX (or Unix) epoch timestamp. * Source - The name of the parser or plugin that produced the event. * Host - The source host system. * User - The user associated with the data. * Description - Message string describing the data. * TZ - L2T 0.65 field. Timezone of the event. * Notes - L2T 0.65 field. Optional notes field or filename and inode.

DESCRIPTION = u'Extended TLN 7 field | delimited output.'

NAME = u'l2ttln'

WriteEventBody(*event*)
Writes the body of an event object to the output.

Parameters **event** (EventObject) – event.

class plaso.output.tln.**TLNBaseOutputModule**(*output_mediator*)
Bases: *plaso.output.interface.LinearOutputModule*

Base class for a TLN output module.

WriteHeader()
Writes the header to the output.

class plaso.output.tln.**TLNOutputModule**(*output_mediator*)
Bases: *plaso.output.tln.TLNBaseOutputModule*

Output module for the TLN format.

TLN defines 5 | separated fields, namely: * Time - 32-bit POSIX (or Unix) epoch timestamp. * Source - The name of the parser or plugin that produced the event. * Host - The source host system. * User - The user associated with the data. * Description - Message string describing the data.

DESCRIPTION = u'TLN 5 field | delimited output.'

NAME = u'tln'

WriteEventBody(*event*)
Writes the body of an event object to the output.

Parameters **event** (EventObject) – event.

plaso.output.xlsx module

Output module for the Excel Spreadsheet (XLSX) output format.

class plaso.output.xlsx.**XLSXOutputModule**(*output_mediator*)
Bases: *plaso.output.interface.OutputModule*

Output module for the Excel Spreadsheet (XLSX) output format.

Close()
Closes the output.

DESCRIPTION = u'Excel Spreadsheet (XLSX) output'

NAME = u'xlsx'

Open()
Creates a new workbook.

Raises

- IOError – if the specified output file already exists.
- ValueError – if the filename is not set.

SetFields(*fields*)
Sets the fields to output.

Parameters **fields** (*list[str]*) – names of the fields to output.

SetFilename(*filename*)
Sets the filename.

Parameters **filename** (*str*) – filename.

SetTimestampFormat(*timestamp_format*)
Set the timestamp format to use for the datetime column.

Parameters **timestamp_format** (*str*) – format string of date and time values.

WriteEventBody(*event*)
Writes the body of an event object to the spreadsheet.

Parameters **event** (EventObject) – event.

WriteHeader()
Writes the header to the spreadsheet.

Module contents

Imports for the output (module) manager.

plaso.parsers package

Subpackages

plaso.parsers.bencode_plugins package

Submodules

plaso.parsers.bencode_plugins.interface module

plaso.parsers.bencode_plugins.transmission module

plaso.parsers.bencode_plugins.utorrent module

Module contents

plaso.parsers.cookie_plugins package

Submodules

plaso.parsers.cookie_plugins.ganalytics module

plaso.parsers.cookie_plugins.interface module

plaso.parsers.cookie_plugins.manager module

Module contents

plaso.parsers.esedb_plugins package

Submodules

plaso.parsers.esedb_plugins.file_history module

plaso.parsers.esedb_plugins.interface module

plaso.parsers.esedb_plugins.msie_webcache module

plaso.parsers.esedb_plugins.srum module

Module contents

plaso.parsers.olecf_plugins package

Submodules

plaso.parsers.olecf_plugins.automatic_destinations module

plaso.parsers.olecf_plugins.default module

plaso.parsers.olecf_plugins.interface module

class plaso.serializer.interface.**AttributeContainerSerializer**
Bases: object

Class that implements the attribute container serializer interface.

ReadSerialized(*serialized*)
Reads an attribute container from serialized form.

Parameters **serialized** (*object*) – serialized form.

Returns attribute container.

Return type *AttributeContainer*

WriteSerialized(*attribute_container*)
Writes an attribute container to serialized form.

Parameters **attribute_container** (AttributeContainer) – attribute container.

Returns serialized form.

Return type object

plaso.serializer.json_serializer module

The json serializer object implementation.

class plaso.serializer.json_serializer.**JSONAttributeContainerSerializer**
Bases: *plaso.serializer.interface.AttributeContainerSerializer*

Class that implements the json attribute container serializer.

classmethod ReadSerialized(*json_string*)
Reads an attribute container from serialized form.

Parameters **json_string** – a JSON string containing the serialized form.

Returns attribute container or None.

Return type *AttributeContainer*

classmethod ReadSerializedDict(*json_dict*)
Reads an attribute container from serialized dictionary form.

Parameters **json_dict** (*dict[str, object]*) – JSON serialized objects.

Returns attribute container or None.

Return type *AttributeContainer*

Raises TypeError – if the serialized dictionary does not contain an AttributeContainer.

classmethod WriteSerialized(*attribute_container*)
Writes an attribute container to serialized form.

Parameters **attribute_container** (AttributeContainer) – attribute container.

Returns A JSON string containing the serialized form.

Return type str

classmethod WriteSerializedDict(*attribute_container*)
Writes an attribute container to serialized form.

Parameters **attribute_container** (AttributeContainer) – attribute container.

Returns JSON serialized objects.

Return type dict[str, object]

Module contents

plaso.storage package

Subpackages

plaso.storage.fake package

Submodules

plaso.storage.fake.writer module

Fake storage writer for testing.

class plaso.storage.fake.writer.**FakeStorageWriter**(*session*, *storage_type=u'session'*, *task=None*)

Bases: *plaso.storage.interface.StorageWriter*

Fake storage writer object.

analysis_reports
list[AnalysisReport] – analysis reports.

session_completion
SessionCompletion – session completion attribute container.

session_start
SessionStart – session start attribute container.

task_completion
TaskCompletion – task completion attribute container.

task_start
TaskStart – task start attribute container.

AddAnalysisReport(*analysis_report*)
Adds an analysis report.

Parameters **analysis_report** (AnalysisReport) – analysis report.

Raises IOError – when the storage writer is closed.

AddError(*error*)
Adds an error.

Parameters **error** (ExtractionError) – error.

Raises IOError – when the storage writer is closed.

AddEvent(*event*)
Adds an event.

Parameters **event** (EventObject) – event.

Raises IOError – when the storage writer is closed or if the event data identifier type is not supported.

AddEventData(*event_data*)
Adds event data.

Parameters **event_data** (EventData) – event data.

Raises IOError – when the storage writer is closed.

AddEventSource(*event_source*)
Adds an event source.

Parameters **event_source** (EventSource) – event source.

Raises IOError – when the storage writer is closed.

AddEventTag(*event_tag*)
Adds an event tag.

Parameters **event_tag** (EventTag) – event tag.

Raises IOError – when the storage writer is closed.

Close()
Closes the storage writer.

Raises IOError – when the storage writer is closed.

CreateTaskStorage(*task*)
Creates a task storage.

Parameters **task** (Task) – task.

Returns storage writer.

Return type *FakeStorageWriter*

Raises IOError – if the task storage already exists.

GetErrors()
Retrieves the errors.

Returns error generator.

Return type generator(*ExtractionError*)

GetEventData()
Retrieves the event data.

Returns event data generator.

Return type generator(*EventData*)

GetEventSources()
Retrieves the event sources.

Returns event source generator.

Return type generator(*EventSource*)

GetEventTags()
Retrieves the event tags.

Returns event tag generator.

Return type generator(EventTags)

GetEvents()
Retrieves the events.

Yields *EventObject* – event.

GetFirstWrittenEventSource()
Retrieves the first event source that was written after open.

Using GetFirstWrittenEventSource and GetNextWrittenEventSource newly added event sources can be retrieved in order of addition.

Returns event source or None if there are no newly written ones.

Return type *EventSource*

Raises IOError – when the storage writer is closed.

GetNextWrittenEventSource()
Retrieves the next event source that was written after open.

Returns event source or None if there are no newly written ones.

Return type *EventSource*

Raises IOError – when the storage writer is closed.

GetSortedEvents(*time_range=None*)
Retrieves the events in increasing chronological order.

Parameters **time_range** (*Optional[TimeRange]*) – time range used to filter events that fall in a specific period.

Returns event generator.

Return type generator(*EventObject*)

Raises IOError – when the storage writer is closed.

Open()
Opens the storage writer.

Raises IOError – if the storage writer is already opened.

PrepareMergeTaskStorage(*task*)
Prepares a task storage for merging.

Parameters **task** (Task) – task.

Raises IOError – if the task storage does not exist.

ReadPreprocessingInformation(*unused_knowledge_base*)
Reads preprocessing information.

The preprocessing information contains the system configuration which contains information about various system specific configuration data, for example the user accounts.

Parameters **knowledge_base** (KnowledgeBase) – is used to store the preprocessing information.

Raises IOError – if the storage type does not support writing preprocessing information or when the storage writer is closed.

SetSerializersProfiler(*serializers_profiler*)
Sets the serializers profiler.

Parameters **serializers_profiler** (SerializersProfiler) – serializers profile.

WritePreprocessingInformation(*unused_knowledge_base*)
Writes preprocessing information.

Parameters **knowledge_base** (KnowledgeBase) – contains the preprocessing information.

Raises IOError – if the storage type does not support writing preprocessing information or when the storage writer is closed.

WriteSessionCompletion(*aborted=False*)
Writes session completion information.

Parameters **aborted** (*Optional[bool]*) – True if the session was aborted.

Raises IOError – if the storage type does not support writing a session completion or when the storage writer is closed.

WriteSessionStart()
Writes session start information.

Raises IOError – if the storage type does not support writing a session start or when the storage writer is closed.

WriteTaskCompletion(*aborted=False*)
Writes task completion information.

Parameters **aborted** (*Optional[bool]*) – True if the session was aborted.

Raises IOError – if the storage type does not support writing a task completion or when the storage writer is closed.

WriteTaskStart()
Writes task start information.

Raises IOError – if the storage type does not support writing a task start or when the storage writer is closed.

Module contents

plaso.storage.sqlite package

Submodules

plaso.storage.sqlite.merge_reader module

Merge reader for SQLite storage files.

class plaso.storage.sqlite.merge_reader.**SQLiteStorageMergeReader**(*storage_writer*, *path*)
Bases: *plaso.storage.interface.StorageFileMergeReader*

SQLite-based storage file reader for merging.

MergeAttributeContainers(*callback=None*, *maximum_number_of_containers=0*)
Reads attribute containers from a task storage file into the writer.

Parameters

- **callback** (*function[StorageWriter, AttributeContainer]*) – function to call after each attribute container is deserialized.
- **maximum_number_of_containers** (*Optional[int]*) – maximum number of containers to merge, where 0 represent no limit.

Returns True if the entire task storage file has been merged.

Return type bool

Raises `OSError` – if the task storage file cannot be deleted.

plaso.storage.sqlite.reader module

Reader for SQLite storage files.

class plaso.storage.sqlite.reader.**SQLiteStorageFileReader**(*path*)
Bases: *plaso.storage.interface.StorageFileReader*

SQLite-based storage file reader.

plaso.storage.sqlite.sqlite_file module

SQLite-based storage.

class plaso.storage.sqlite.sqlite_file.**SQLiteStorageFile**(*maximum_buffer_size=0*, *storage_type=u'session'*)
Bases: *plaso.storage.interface.BaseStorageFile*

SQLite-based storage file.

format_version
int – storage format version.

serialization_format
str – serialization format.

storage_type
str – storage type.

AddAnalysisReport(*analysis_report*)
Adds an analysis report.

Parameters **analysis_report** (`AnalysisReport`) – analysis report.

Raises `IOError` – when the storage file is closed or read-only.

AddError(*error*)
Adds an error.

Parameters **error** (`ExtractionError`) – error.

Raises `IOError` – when the storage file is closed or read-only.

AddEvent(*event*)
Adds an event.

Parameters **event** (`EventObject`) – event.

Raises `IOError` – when the storage file is closed or read-only or if the event data identifier type is not supported.

AddEventData(*event_data*)
Adds event data.

Parameters **event_data** (`EventData`) – event data.

Raises `IOError` – when the storage file is closed or read-only.

AddEventSource(*event_source*)
Adds an event source.

Parameters **event_source** (EventSource) – event source.

Raises IOError – when the storage file is closed or read-only.

AddEventTag(*event_tag*)
Adds an event tag.

Parameters **event_tag** (EventTag) – event tag.

Raises IOError – when the storage file is closed or read-only or if the event identifier type is not supported.

AddEventTags(*event_tags*)
Adds event tags.

Parameters **event_tags** (*list[EventTag]*) – event tags.

Raises IOError – when the storage file is closed or read-only or if the event tags cannot be serialized.

classmethod CheckSupportedFormat(*path*)
Checks if the storage file format is supported.

Parameters **path** (*str*) – path to the storage file.

Returns True if the format is supported.

Return type bool

Close()
Closes the storage.

Raises IOError – if the storage file is already closed.

GetAnalysisReports()
Retrieves the analysis reports.

Returns analysis report generator.

Return type generator(*AnalysisReport*)

GetErrors()
Retrieves the errors.

Returns error generator.

Return type generator(*ExtractionError*)

GetEventData()
Retrieves the event data.

Yields *generator(EventData)* – event data generator.

GetEventDataByIdentifier(*identifier*)
Retrieves specific event data.

Parameters **identifier** (SQLTableIdentifier) – event data identifier.

Returns event data or None if not available.

Return type *EventData*

GetEventSourceByIndex(*index*)
Retrieves a specific event source.

Parameters **index** (*int*) – event source index.

Returns event source or None if not available.

Return type *EventSource*

GetEventSources()
Retrieves the event sources.

Yields *generator(EventSource)* – event source generator.

GetEventTagByIdentifier(*identifier*)
Retrieves a specific event tag.

Parameters **identifier** (SQLTableIdentifier) – event tag identifier.

Returns event tag or None if not available.

Return type *EventTag*

GetEventTags()
Retrieves the event tags.

Yields *EventTag* – event tag.

GetEvents()
Retrieves the events.

Yields *EventObject* – event.

GetNumberOfAnalysisReports()
Retrieves the number analysis reports.

Returns number of analysis reports.

Return type int

GetNumberOfEventSources()
Retrieves the number event sources.

Returns number of event sources.

Return type int

GetSessions()
Retrieves the sessions.

Yields *Session* – session attribute container.

Raises IOError – if a stream is missing or there is a mismatch in session identifiers between the session start and completion attribute containers.

GetSortedEvents(*time_range=None*)
Retrieves the events in increasing chronological order.

Parameters **time_range** (*Optional[TimeRange]*) – time range used to filter events that fall in a specific period.

Yields *EventObject* – event.

HasAnalysisReports()
Determines if a store contains analysis reports.

Returns True if the store contains analysis reports.

Return type bool

HasErrors()
Determines if a store contains extraction errors.

Returns True if the store contains extraction errors.

Return type bool

HasEventTags()
Determines if a store contains event tags.

Returns True if the store contains event tags.

Return type bool

Open(*path=None*, *read_only=True*, ***unused_kwargs*)
Opens the storage.

Parameters

- **path** (*Optional[str]*) – path to the storage file.
- **read_only** (*Optional[bool]*) – True if the file should be opened in read-only mode.

Raises

- IOError – if the storage file is already opened or if the database cannot be connected.
- ValueError – if path is missing.

ReadPreprocessingInformation(*knowledge_base*)
Reads preprocessing information.

The preprocessing information contains the system configuration which contains information about various system specific configuration data, for example the user accounts.

Parameters **knowledge_base** (KnowledgeBase) – is used to store the preprocessing information.

WritePreprocessingInformation(*knowledge_base*)
Writes preprocessing information.

Parameters **knowledge_base** (KnowledgeBase) – contains the preprocessing information.

Raises IOError – if the storage type does not support writing preprocess information or the storage file is closed or read-only.

WriteSessionCompletion(*session_completion*)
Writes session completion information.

Parameters **session_completion** (SessionCompletion) – session completion information.

Raises IOError – when the storage file is closed or read-only.

WriteSessionStart(*session_start*)
Writes session start information.

Parameters **session_start** (SessionStart) – session start information.

Raises IOError – when the storage file is closed or read-only.

WriteTaskCompletion(*task_completion*)
Writes task completion information.

Parameters **task_completion** (TaskCompletion) – task completion information.

Raises IOError – when the storage file is closed or read-only.

WriteTaskStart(*task_start*)
Writes task start information.

Parameters **task_start** (TaskStart) – task start information.

Raises IOError – when the storage file is closed or read-only.

plaso.storage.sqlite.writer module

Storage writer for SQLite storage files.

class plaso.storage.sqlite.writer.**SQLiteStorageFileWriter**(*session*, *output_file*, *storage_type=u'session'*, *task=None*)
Bases: *plaso.storage.interface.StorageFileWriter*

SQLite-based storage file writer.

Module contents

Submodules

plaso.storage.event_heaps module

Heaps to sort events in chronological order.

class plaso.storage.event_heaps.**BaseEventHeap**
Bases: object

Event heap interface.

PopEvent()
Pops an event from the heap.

Returns event.

Return type *EventObject*

PopEvents()
Pops events from the heap.

Yields *EventObject* – event.

PushEvent(*event*)
Pushes an event onto the heap.

Parameters **event** (EventObject) – event.

PushEvents(*events*)
Pushes events onto the heap.

Parameters **list[EventObject]** (events) – events.

number_of_events
int – number of serialized events on the heap.

class plaso.storage.event_heaps.**EventHeap**
Bases: *plaso.storage.event_heaps.BaseEventHeap*

Event heap.

PopEvent()
Pops an event from the heap.

Returns event.

Return type *EventObject*

PushEvent(*event*)
Pushes an event onto the heap.

Parameters **event** (EventObject) – event.

class plaso.storage.event_heaps.**SerializedEventHeap**
Bases: object

Serialized event heap.

data_size
int – total data size of the serialized events on the heap.

Empty()
Empties the heap.

PopEvent()
Pops an event from the heap.

Returns

contains:

int: event timestamp or None if the heap is empty bytes: serialized event or None if the heap is empty

Return type tuple

PushEvent(*timestamp*, *event_data*)
Pushes a serialized event onto the heap.

Parameters

- **timestamp** (*int*) – event timestamp, which contains the number of micro seconds since January 1, 1970, 00:00:00 UTC.
- **event_data** (*bytes*) – serialized event.

number_of_events
int – number of serialized events on the heap.

plaso.storage.event_tag_index module

The event tag index.

class plaso.storage.event_tag_index.**EventTagIndex**
Bases: object

Event tag index.

The event tag index is used to map event tags to events.

It is necessary for the ZIP storage files since previously stored event tags cannot be altered.

GetEventTagByIdentifier(*storage_file*, *event_identifier*)
Retrieves the most recently updated event tag for an event.

Parameters

- **storage_file** (BaseStorageFile) – storage file.
- **event_identifier** (AttributeContainerIdentifier) – event attribute container identifier.

Returns event tag or None if the event has no event tag.

Return type *EventTag*

SetEventTag(*event_tag*)
Sets an event tag in the index.

Parameters **event_tag** (EventTag) – event tag.

plaso.storage.factory module

This file contains the storage factory class.

class plaso.storage.factory.**StorageFactory**
Bases: object

Storage factory.

classmethod CreateStorageFile(*storage_format*)
Creates a storage file.

Parameters **storage_format** (*str*) – storage format.

Returns

a storage file or None if the storage file cannot be opened or the storage format is not supported.

Return type StorageFile

classmethod CreateStorageFileForFile(*path*)
Creates a storage file based on the file.

Parameters **path** (*str*) – path to the storage file.

Returns

a storage file or None if the storage file cannot be opened or the storage format is not supported.

Return type StorageFile

classmethod CreateStorageReaderForFile(*path*)
Creates a storage reader based on the file.

Parameters **path** (*str*) – path to the storage file.

Returns

a storage reader or None if the storage file cannot be opened or the storage format is not supported.

Return type *StorageReader*

classmethod CreateStorageWriter(*storage_format*, *session*, *path*)
Creates a storage writer.

Parameters

- **session** (Session) – session the storage changes are part of.
- **path** (*str*) – path to the storage file.
- **storage_format** (*str*) – storage format.

Returns

a storage writer or None if the storage file cannot be opened or the storage format is not supported.

Return type *StorageWriter*

classmethod CreateStorageWriterForFile(*session*, *path*)
Creates a storage writer based on the file.

Parameters

- **session** (Session) – session the storage changes are part of.
- **path** (*str*) – path to the storage file.

Returns

a storage writer or None if the storage file cannot be opened or the storage format is not supported.

Return type *StorageWriter*

plaso.storage.identifiers module

Storage attribute container identifier objects.

class plaso.storage.identifiers.**FakeIdentifier**(*attribute_values_hash*)
Bases: *plaso.containers.interface.AttributeContainerIdentifier*

Fake attribute container identifier intended for testing.

attribute_values_hash
int – hash value of the attribute values.

CopyToString()
Copies the identifier to a string representation.

Returns unique identifier or None.

Return type str

class plaso.storage.identifiers.**SQLTableIdentifier**(*name*, *row_identifier*)
Bases: *plaso.containers.interface.AttributeContainerIdentifier*

SQL table attribute container identifier.

The identifier is used to uniquely identify attribute containers. Where for example an attribute container is stored as a JSON serialized data in a SQLite database file.

name
str – name of the table.

row_identifier
int – unique identifier of the row in the table.

CopyToString()
Copies the identifier to a string representation.

Returns unique identifier or None.

Return type str

class plaso.storage.identifiers.**SerializedStreamIdentifier**(*stream_number*, *entry_index*)
Bases: *plaso.containers.interface.AttributeContainerIdentifier*

Serialized stream attribute container identifier.

The identifier is used to uniquely identify attribute containers. Where for example an attribute container is stored as a JSON serialized data in a ZIP file.

stream_number
int – number of the serialized attribute container stream.

entry_index
int – number of the serialized event within the stream.

CopyToString()
Copies the identifier to a string representation.

Returns unique identifier or None.

Return type str

plaso.storage.interface module

The storage interface classes.

class plaso.storage.interface.**BaseStorageFile**
Bases: *plaso.storage.interface.BaseStore*

Interface for file-based stores.

SetSerializersProfiler(*serializers_profiler*)
Sets the serializers profiler.

Parameters **serializers_profiler** (SerializersProfiler) – serializers profile.

class plaso.storage.interface.**BaseStore**
Bases: object

Storage interface.

AddAnalysisReport(*analysis_report*)
Adds an analysis report.

Parameters **analysis_report** (AnalysisReport) – analysis report.

AddError(*error*)
Adds an error.

Parameters **error** (ExtractionError) – error.

AddEvent(*event*)
Adds an event.

Parameters **event** (EventObject) – event.

AddEventSource(*event_source*)
Adds an event source.

Parameters **event_source** (EventSource) – event source.

AddEventTag(*event_tag*)
Adds an event tag.

Parameters **event_tag** (EventTag) – event tag.

Close()
Closes the storage.

GetAnalysisReports()
Retrieves the analysis reports.

Yields *AnalysisReport* – analysis report.

GetErrors()
Retrieves the errors.

Yields *ExtractionError* – error.

GetEventData()
Retrieves the event data.

Yields *EventData* – event data.

GetEventDataByIdentifier(*identifier*)
Retrieves specific event data.

Parameters **identifier** (AttributeContainerIdentifier) – event data identifier.

Returns event data or None if not available.

Return type *EventData*

GetEventSources()
Retrieves the event sources.

Yields *EventSource* – event source.

GetEventTagByIdentifier(*identifier*)
Retrieves a specific event tag.

Parameters **identifier** (AttributeContainerIdentifier) – event tag identifier.

Returns event tag or None if not available.

Return type *EventTag*

GetEventTags()
Retrieves the event tags.

Yields *EventTag* – event tag.

GetEvents()
Retrieves the events.

Yields *EventObject* – event.

GetNumberOfEventSources()
Retrieves the number event sources.

Returns number of event sources.

Return type int

GetSortedEvents(*time_range=None*)
Retrieves the events in increasing chronological order.

This includes all events written to the storage including those pending being flushed (written) to the storage.

Parameters **time_range** (*Optional[TimeRange]*) – time range used to filter events that fall in a specific period.

Yields *EventObject* – event.

HasAnalysisReports()
Determines if a store contains analysis reports.

Returns True if the store contains analysis reports.

Return type bool

HasErrors()
Determines if a store contains extraction errors.

Returns True if the store contains extraction errors.

Return type bool

HasEventTags()
Determines if a store contains event tags.

Returns True if the store contains event tags.

Return type bool

Open(***kwargs*)
Opens the storage.

ReadPreprocessingInformation(*knowledge_base*)
Reads preprocessing information.

The preprocessing information contains the system configuration which contains information about various system specific configuration data, for example the user accounts.

Parameters **knowledge_base** (KnowledgeBase) – is used to store the preprocessing information.

WritePreprocessingInformation(*knowledge_base*)
Writes preprocessing information.

Parameters **knowledge_base** (KnowledgeBase) – contains the preprocessing information.

WriteSessionCompletion(*session_completion*)
Writes session completion information.

Parameters **session_completion** (SessionCompletion) – session completion information.

WriteSessionStart(*session_start*)
Writes session start information.

Parameters **session_start** (SessionStart) – session start information.

WriteTaskCompletion(*task_completion*)
Writes task completion information.

Parameters **task_completion** (TaskCompletion) – task completion information.

WriteTaskStart(*task_start*)
Writes task start information.

Parameters **task_start** (TaskStart) – task start information.

class plaso.storage.interface.**SerializedAttributeContainerList**
Bases: object

Serialized attribute container list.

The list is unsorted and pops attribute containers in the same order as pushed to preserve order.

The GetAttributeContainerByIndex method should be used to read attribute containers from the list while it being filled.

data_size
int – total data size of the serialized attribute containers on the list.

next_sequence_number
int – next attribute container sequence number.

Empty()
Empties the list.

GetAttributeContainerByIndex(*index*)
Retrieves a specific serialized attribute container from the list.

Parameters **index** (*int*) – attribute container index.

Returns serialized attribute container data or None if not available.

Return type bytes

Raises IndexError – if the index is less than zero.

PopAttributeContainer()
Pops a serialized attribute container from the list.

Returns serialized attribute container data.

Return type bytes

PushAttributeContainer(*serialized_data*)
Pushes a serialized attribute container onto the list.

Parameters **serialized_data** (*bytes*) – serialized attribute container data.

number_of_attribute_containers
int – number of serialized attribute containers on the list.

class plaso.storage.interface.**StorageFileMergeReader**(*storage_writer*)
Bases: *plaso.storage.interface.StorageMergeReader*

Storage reader interface for file-based stores merging.

class plaso.storage.interface.**StorageFileReader**(*path*)
Bases: *plaso.storage.interface.StorageReader*

File-based storage reader interface.

Close()
Closes the storage reader.

GetAnalysisReports()
Retrieves the analysis reports.

Returns analysis report generator.

Return type generator(*AnalysisReport*)

GetErrors()
Retrieves the errors.

Returns error generator.

Return type generator(*ExtractionError*)

GetEventData()
Retrieves the event data.

Returns event data generator.

Return type generator(*EventData*)

GetEventDataByIdentifier(*identifier*)
Retrieves specific event data.

Parameters **identifier** (AttributeContainerIdentifier) – event data identifier.

Returns event data or None if not available.

Return type *EventData*

GetEventSources()
Retrieves the event sources.

Returns event source generator.

Return type generator(*EventSource*)

GetEventTagByIdentifier(*identifier*)
Retrieves a specific event tag.

Parameters **identifier** (AttributeContainerIdentifier) – event tag identifier.

Returns event tag or None if not available.

Return type *EventTag*

GetEventTags()
Retrieves the event tags.

Returns event tag generator.

Return type generator(*EventTag*)

GetEvents()
Retrieves the events.

Returns event generator.

Return type generator(*EventObject*)

GetNumberOfAnalysisReports()
Retrieves the number analysis reports.

Returns number of analysis reports.

Return type int

GetSortedEvents(*time_range=None*)
Retrieves the events in increasing chronological order.

This includes all events written to the storage including those pending being flushed (written) to the storage.

Parameters **time_range** (*Optional[TimeRange]*) – time range used to filter events that fall in a specific period.

Returns event generator.

Return type generator(*EventObject*)

ReadPreprocessingInformation(*knowledge_base*)
Reads preprocessing information.

The preprocessing information contains the system configuration which contains information about various system specific configuration data, for example the user accounts.

Parameters **knowledge_base** (KnowledgeBase) – is used to store the preprocessing information.

class plaso.storage.interface.**StorageFileWriter**(*session*, *output_file*, *storage_type=u'session'*, *task=None*)
Bases: *plaso.storage.interface.StorageWriter*

Defines an interface for a file-backed storage writer.

AddAnalysisReport(*analysis_report*)
Adds an analysis report.

Parameters **analysis_report** (AnalysisReport) – analysis report.

Raises IOError – when the storage writer is closed.

AddError(*error*)
Adds an error.

Parameters **error** (*AnalysisError|ExtractionError*) – an analysis or extraction error.

Raises IOError – when the storage writer is closed.

AddEvent(*event*)
Adds an event.

Parameters **event** (EventObject) – an event.

Raises IOError – when the storage writer is closed.

AddEventData(*event_data*)
Adds event data.

Parameters **event_data** (EventData) – event data.

Raises IOError – when the storage writer is closed.

AddEventSource(*event_source*)
Adds an event source.

Parameters **event_source** (EventSource) – an event source.

Raises IOError – when the storage writer is closed.

AddEventTag(*event_tag*)
Adds an event tag.

Parameters **event_tag** (EventTag) – an event tag.

Raises IOError – when the storage writer is closed.

CheckTaskReadyForMerge(*task*)
Checks if a task is ready for merging with this session storage.

Parameters **task** (Task) – task.

Returns True if the task is ready to be merged.

Return type bool

Raises IOError – if the storage type is not supported or if the temporary path for the task storage does not exist.

Close()
Closes the storage writer.

Raises IOError – when the storage writer is closed.

CreateTaskStorage(*task*)
Creates a task storage.

The task storage is used to store attributes created by the task.

Parameters **task** (Task) – task.

Returns storage writer.

Return type *StorageWriter*

Raises IOError – if the storage type is not supported or if the temporary path for the task storage does not exist.

GetEventTagByIdentifier(*identifier*)
Retrieves a specific event tag.

Parameters **identifier** (AttributeContainerIdentifier) – event tag identifier.

Returns event tag or None if not available.

Return type *EventTag*

GetEventTags()
Retrieves the event tags.

Returns event tag generator.

Return type generator(*EventTag*)

GetEvents()
Retrieves the events.

Returns event generator.

Return type generator(*EventObject*)

Raises IOError – when the storage writer is closed.

GetFirstWrittenEventSource()
Retrieves the first event source that was written after open.

Using GetFirstWrittenEventSource and GetNextWrittenEventSource newly added event sources can be retrieved in order of addition.

Returns event source or None if there are no newly written ones.

Return type *EventSource*

Raises IOError – when the storage writer is closed.

GetNextWrittenEventSource()
Retrieves the next event source that was written after open.

Returns event source or None if there are no newly written ones.

Return type *EventSource*

Raises IOError – when the storage writer is closed.

GetSortedEvents(*time_range=None*)
Retrieves the events in increasing chronological order.

This includes all events written to the storage including those pending being flushed (written) to the storage.

Parameters **time_range** (*Optional[TimeRange]*) – time range used to filter events that fall in a specific period.

Returns event generator.

Return type generator(*EventObject*)

Raises IOError – when the storage writer is closed.

Open()
Opens the storage writer.

Raises IOError – if the storage writer is already opened.

PrepareMergeTaskStorage(*task*)
Prepares a task storage for merging.

Parameters **task** (Task) – task.

Raises IOError – if the storage type is not supported or if the temporary path for the task storage does not exist.

ReadPreprocessingInformation(*knowledge_base*)
Reads preprocessing information.

The preprocessing information contains the system configuration which contains information about various system specific configuration data, for example the user accounts.

Parameters **knowledge_base** (KnowledgeBase) – is used to store the preprocessing information.

Raises IOError – when the storage writer is closed.

SetSerializersProfiler(*serializers_profiler*)
Sets the serializers profiler.

Parameters **serializers_profiler** (SerializersProfiler) – serializers profile.

StartMergeTaskStorage(*task*)
Starts a merge of a task storage with the session storage.

Parameters **task** (Task) – task.

Returns storage merge reader of the task storage.

Return type *StorageMergeReader*

Raises IOError – if the storage file cannot be opened or if the storage type is not supported or if the temporary path for the task storage does not exist or if the temporary path for the task storage doe not refers to a file.

StartTaskStorage()
Creates a temporary path for the task storage.

Raises IOError – if the storage type is not supported or if the temporary path for the task storage already exists.

StopTaskStorage(*abort=False*)
Removes the temporary path for the task storage.

The results of tasks will be lost on abort.

Parameters **abort** (*bool*) – True to indicate the stop is issued on abort.

Raises IOError – if the storage type is not supported or if the temporary path for the task storage does not exist.

WritePreprocessingInformation(*knowledge_base*)
Writes preprocessing information.

Parameters **knowledge_base** (KnowledgeBase) – contains the preprocessing information.

Raises IOError – if the storage type does not support writing preprocessing information or when the storage writer is closed.

WriteSessionCompletion(*aborted=False*)
Writes session completion information.

Parameters **aborted** (*Optional[bool]*) – True if the session was aborted.

Raises IOError – if the storage type is not supported or when the storage writer is closed.

WriteSessionStart()
Writes session start information.

Raises IOError – if the storage type is not supported or when the storage writer is closed.

WriteTaskCompletion(*aborted=False*)
Writes task completion information.

Parameters **aborted** (*Optional[bool]*) – True if the session was aborted.

Raises IOError – if the storage type is not supported or when the storage writer is closed.

WriteTaskStart()
Writes task start information.

Raises IOError – if the storage type is not supported or when the storage writer is closed.

class plaso.storage.interface.**StorageMergeReader**(*storage_writer*)
Bases: object

Storage reader interface for merging.

MergeAttributeContainers(*callback=None*, *maximum_number_of_containers=0*)
Reads attribute containers from a task storage file into the writer.

Parameters

- **callback** (*function[StorageWriter, AttributeContainer]*) – function to call after each attribute container is deserialized.
- **maximum_number_of_containers** (*Optional[int]*) – maximum number of containers to merge, where 0 represent no limit.

Returns True if the entire task storage file has been merged.

Return type bool

class plaso.storage.interface.**StorageReader**
Bases: object

Storage reader interface.

Close()
Closes the storage reader.

GetAnalysisReports()
Retrieves the analysis reports.

Yields *AnalysisReport* – analysis report.

GetErrors()
Retrieves the errors.

Yields *ExtractionError* – error.

GetEventData()
Retrieves the event data.

Yields *EventData* – event data.

GetEventDataByIdentifier(*identifier*)
Retrieves specific event data.

Parameters **identifier** (AttributeContainerIdentifier) – event data identifier.

Returns event data or None if not available.

Return type *EventData*

GetEventSources()
Retrieves event sources.

Yields *EventSourceObject* – event source.

GetEventTagByIdentifier(*identifier*)
Retrieves a specific event tag.

Parameters **identifier** (AttributeContainerIdentifier) – event tag identifier.

Returns event tag or None if not available.

Return type *EventTag*

GetEventTags()
Retrieves the event tags.

Yields *EventTag* – event tag.

GetEvents()
Retrieves the events.

Yields *EventObject* – event.

GetNumberOfAnalysisReports()
Retrieves the number analysis reports.

Returns number of analysis reports.

Return type int

GetSortedEvents(*time_range=None*)
Retrieves the events in increasing chronological order.

This includes all events written to the storage including those pending being flushed (written) to the storage.

Parameters **time_range** (*Optional[TimeRange]*) – time range used to filter events that fall in a specific period.

Yields *EventObject* – event.

ReadPreprocessingInformation(*knowledge_base*)
Reads preprocessing information.

The preprocessing information contains the system configuration which contains information about various system specific configuration data, for example the user accounts.

Parameters **knowledge_base** (KnowledgeBase) – is used to store the preprocessing information.

__enter__()
Make usable with "with" statement.

__exit__(*unused_type*, *unused_value*, *unused_traceback*)
Make usable with "with" statement.

class plaso.storage.interface.**StorageWriter**(*session*, *storage_type=u'session'*, *task=None*)
Bases: object

Storage writer interface.

number_of_analysis_reports
int – number of analysis reports written.

number_of_errors
int – number of errors written.

number_of_event_sources
int – number of event sources written.

number_of_event_tags
int – number of event tags written.

number_of_events
int – number of events written.

AddAnalysisReport(*analysis_report*)
Adds an analysis report.

Parameters **analysis_report** (AnalysisReport) – a report.

AddError(*error*)
Adds an error.

Parameters **error** (ExtractionError) – an error.

AddEvent(*event*)
Adds an event.

Parameters **event** (EventObject) – an event.

AddEventSource(*event_source*)
Adds an event source.

Parameters **event_source** (EventSource) – an event source.

AddEventTag(*event_tag*)
Adds an event tag.

Parameters **event_tag** (EventTag) – an event tag.

Close()
Closes the storage writer.

CreateTaskStorage(*unused_task*)
Creates a task storage.

Parameters **task** (Task) – task.

Returns storage writer.

Return type *StorageWriter*

Raises NotImplementedError – since there is no implementation.

GetEvents()
Retrieves the events.

Yields *EventObject* – event.

GetFirstWrittenEventSource()
Retrieves the first event source that was written after open.

Using GetFirstWrittenEventSource and GetNextWrittenEventSource newly added event sources can be retrieved in order of addition.

Returns event source or None if there are no newly written ones.

Return type *EventSource*

GetNextWrittenEventSource()
Retrieves the next event source that was written after open.

Returns event source or None if there are no newly written ones.

Return type *EventSource*

GetSortedEvents(*time_range=None*)
Retrieves the events in increasing chronological order.

This includes all events written to the storage including those pending being flushed (written) to the storage.

Parameters **time_range** (*Optional[TimeRange]*) – time range used to filter events that fall in a specific period.

Yields *EventObject* – event.

Open()
Opens the storage writer.

PrepareMergeTaskStorage(*unused_task*)
Prepares a task storage for merging.

Parameters **task** (Task) – task.

Raises NotImplementedError – since there is no implementation.

ReadPreprocessingInformation(*knowledge_base*)
Reads preprocessing information.

The preprocessing information contains the system configuration which contains information about various system specific configuration data, for example the user accounts.

Parameters **knowledge_base** (KnowledgeBase) – is used to store the preprocessing information.

SetSerializersProfiler(*serializers_profiler*)
Sets the serializers profiler.

Parameters **serializers_profiler** (SerializersProfiler) – serializers profile.

WritePreprocessingInformation(*knowledge_base*)
Writes preprocessing information.

Parameters **knowledge_base** (KnowledgeBase) – contains the preprocessing information.

WriteSessionCompletion(*aborted=False*)
Writes session completion information.

Parameters **aborted** (*Optional[bool]*) – True if the session was aborted.

WriteSessionStart()
Writes session start information.

WriteTaskCompletion(*aborted=False*)
Writes task completion information.

Parameters **aborted** (*Optional[bool]*) – True if the session was aborted.

WriteTaskStart()
Writes task start information.

plaso.storage.time_range module

Storage time range objects.

class plaso.storage.time_range.**TimeRange**(*start_timestamp*, *end_timestamp*)
Bases: object

Date and time range.

The timestamp are integers containing the number of microseconds since January 1, 1970, 00:00:00 UTC.

duration
int – duration of the range in microseconds.

end_timestamp
int – timestamp that marks the end of the range.

start_timestamp
int – timestamp that marks the start of the range.

Module contents

plaso.unix package

Submodules

plaso.unix.bsmtoken module

This file contains the Basic Security Module definitions.

Module contents

plaso.winnt package

Submodules

plaso.winnt.human_readable_service_enums module

This file contains constants for making service keys more readable.

plaso.winnt.known_folder_ids module

This file contains the Windows NT Known Folder identifier definitions.

plaso.winnt.language_ids module

This file contains the Windows NT Language identifiers.

plaso.winnt.shell_folder_ids module

This file contains the Windows NT shell folder identifier definitions.

plaso.winnt.time_zones module

This file contains the Windows NT time zone definitions.

The Windows time zone names can be obtained from the following Windows Registry key: HKEY_LOCAL_MACHINESOFTWAREMicrosoftWindows NTCurrentVersionTime Zones

Module contents

1.1.2 Submodules

1.1.3 plaso.dependencies module

Functionality to check for the availability and version of dependencies.

`plaso.dependencies.`**`CheckDependencies`**(*verbose_output=True*)

Checks the availability of the dependencies.

Parameters **`verbose_output`** (*`Optional[bool]`*) – True if output should be verbose.

Returns True if the dependencies are available, False otherwise.

Return type bool

1.1.4 Module contents

Super timeline all the things (Plaso Langar Að Safna Öllu).

log2timeline is a tool designed to extract timestamps from various files found on a typical computer system(s) and aggregate them. Plaso is the Python rewrite of log2timeline.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z