python-limacharlie Documentation

Release 3.13.7

Refraction Point, Inc

Oct 10, 2020

Contents

LimaCharlie.io is Security Infrastructure as a Service platform.

https://limacharlie.io

View code, installation instructions and other usage information: https://github.com/refractionpoint/python-limacharlie/

Contents:

CHAPTER 1

limacharlie package

1.1 Submodules

1.2 limacharlie.Firehose module

class limacharlie.Firehose.**Firehose**(*manager*, *listen_on*, *data_type*, *public_dest=None*, *name=None*, *ssl_cert=None*, *ssl_key=None*, *is_parse=True*, *max_buffer=1024*, *inv_id=None*, *tag=None*, *cat=None*, *sid=None*, *is_delete_on_failure=False*, *on_dropped=None*)

Bases: object

Listener object to receive data (Events, Detects or Audit) from a limacharlie.io Organization in push mode.

getDropped()
Get the number of messages dropped because queue was full.

resetDroppedCounter()
Reset the counter of dropped messages.

shutdown()
Stop receiving data and potentially unregister the Output (if created here).

1.3 limacharlie.Jobs module

class limacharlie.Jobs.**Job**(*manager*, *data*)

Bases: object

Representation of a Job created by Services.

delete()
Delete this job.

fetchDetails()
Fetch detailed activity for this job in the cloud.

isFinished()
Check if this job has terminated.

Returns True if the job is finished.

update()
Fetch any updates to the job found in the cloud.

1.4 limacharlie.Logs module

class limacharlie.Logs.**Logs**(*manager*, *accessToken=None*)
Bases: object

Helper object to upload External Logs to limacharlie.io without going through a sensor.

getOriginal(*payloadId*, *filePath=None*, *fileObj=None*, *optParams={}*, *customGetter=None*)
Download an orginal log.

Parameters

- **payloadId** (*str*) – the payload identifier to download.
- **filePath** (*str*) – optional path where to download the file to.
- **fileObj** (*file obj*) – optional file object where to write the log.

listArtifacts(*type=None*, *source=None*, *originalPath=None*, *after=None*, *before=None*, *withData=False*, *optParams={}*, *customGetter=None*)
Get the list of artifacts matching parameters.

Parameters

- **type** (*str*) – only list artifacts with type.
- **source** (*str*) – only list artifacts from this source.
- **originalPath** (*str*) – only list artifacts with this original path.
- **after** (*int*) – list artifacts after a given second epoch.
- **before** (*int*) – list artifacts before a given second epoch.
- **withData** (*bool*) – if True, artifact will be downloaded inline and the return value will be a tuple (artifactRecord, localFilePath).

upload(*filePath*, *source=None*, *hint=None*, *payloadId=None*, *allowMultipart=False*, *originalPath=None*, *nDaysRetention=30*)
Upload a log.

Parameters

- **filePath** (*str*) – path to the file to upload.
- **source** (*str*) – optional source identifier for where the log came from.
- **hint** (*str*) – optional data format hint for the log.
- **payloadId** (*str*) – optional unique payload identifier for the log, used to perform idempotent uploads.
- **allowMultipart** (*bool*) – unused, if True will perform multi-part upload for large logs.

- **nDaysRetention** (*int*) – number of days the data should be retained in the cloud.

1.5 limacharlie.Manager module

class limacharlie.Manager.**Manager**(*oid=None*, *secret_api_key=None*, *environment=None*, *inv_id=None*, *print_debug_fn=None*, *is_interactive=False*, *extra_params={}*, *jwt=None*, *uid=None*, *onRefreshAuth=None*, *isRetryQuotaErrors=False*)

Bases: object

General interface to a limacharlie.io Organization.

addApiKey(*keyName*, *permissions=[]*)
Add an API key to an organization.

Parameters

- **keyName** (*str*) – name of the key to add.
- **permissions** (*str[]*) – list of permissions for the key.

Returns the secret value of the new API key.

addUser(*email*)
Add a user to an organization.

Parameters **email** (*str*) – email of the user to add.

addUserPermission(*email*, *permission*)
Add a user to an organization.

Parameters

- **email** (*str*) – email of the user to add.
- **permission** (*str*) – permission to add to the user.

add_fp(*name*, *rule*, *isReplace=False*, *ttl=None*)
Add a False Positive rule to the Organization.

For detailed explanation and possible rules parameters see the official documentation, naming is the same as for the REST interface.

Parameters

- **name** (*str*) – name to give to the rule.
- **isReplace** (*boolean*) – if True, replace existing rule with the same name.
- **detection** (*dict*) – dictionary representing the False Positive rule content.
- **ttl** (*int*) – number of seconds before the rule should be auto-deleted.

Returns the REST API response (JSON).

add_output(*name*, *module*, *type*, ***kwargs*)
Add an Output to the Organization.

For detailed explanation and possible Output module parameters see the official documentation, naming is the same as for the REST interface.

Parameters

- **name** (*str*) – name to give to the Output.

- **module** (*str*) – name of the Output module to use.
- **type** (*str*) – type of Output stream.
- ****kwargs** – arguments specific to the Output module, see official doc.

Returns the REST API response (JSON).

add_rule(*name*, *detection*, *response*, *isReplace=False*, *namespace=None*, *isEnabled=True*, *ttl=None*)
Add a Rule to the Organization.

For detailed explanation and possible Rules parameters see the official documentation, naming is the same as for the REST interface.

Parameters

- **name** (*str*) – name to give to the Rule.
- **namespace** (*str*) – optional namespace to operator on, defaults to "general".
- **isReplace** (*boolean*) – if True, replace existing Rule with the same name.
- **detection** (*dict*) – dictionary representing the detection component of the Rule.
- **response** (*list*) – list representing the response component of the Rule.
- **isEnabled** (*boolean*) – if True (default), the rule is enabled.
- **ttl** (*int*) – number of seconds before the rule should be auto-deleted.

Returns the REST API response (JSON).

delIngestionKey(*name*)
Delete an Ingestion key.

Parameters **name** (*str*) – name of the Ingestion key to delete.

del_fp(*name*)
Remove a False Positive rule from the Organization.

Parameters **name** (*str*) – the name of the rule to remove.

Returns the REST API response (JSON).

del_output(*name*)
Remove an Output from the Organization.

Parameters **name** (*str*) – the name of the Output to remove.

Returns the REST API response (JSON).

del_rule(*name*, *namespace=None*)
Remove a Rule from the Organization.

Parameters

- **name** (*str*) – the name of the Rule to remove.
- **namespace** (*str*) – optional namespace to operator on, defaults to "general".

Returns the REST API response (JSON).

exportSensorList()
Perform a bulk export of the entire sensor list.

Returns a dictionary of sensors with their information and tags.

fps()
Get the list of all False Positive rules for the Organization.

Returns a list of False Positive rules (JSON).

getApiKeys()
Get the list of API keys in the organization.

getAvailableServices()
Get the list of Services currently available.

Returns List of Service names.

getBatchObjectInformation(*objects*, *isCaseSensitive=True*)
Get object prevalence information in a batch.

Parameters

- **objects** (*dict*) – dictionary of object type to list of object names to query for (objects["file_name"] = ["a.exe", "b.exe"]).
- **isCaseSensitive** (*bool*) – False to ignore case in the object name.

Returns a dict with keys as time ranges and values are maps of object types to object name lists.

getHistoricDetections(*start*, *end*, *limit=None*, *cat=None*)
Get the detections for this organization between the two times, requires Insight (retention) enabled.

Parameters

- **start** (*int*) – start unix (seconds) timestamp to fetch detects from.
- **end** (*int*) – end unix (seconds) timestamp to feth detects to.
- **limit** (*int*) – maximum number of detects to return.
- **cat** (*str*) – return dects only from this category.

Returns a generator of detects.

getIngestionKeys()
Get the Ingestion keys associated to this organization.

Returns Dictionary of the Ingestion keys.

getInsightHostCountPerPlatform()
Get the number of hosts for each platform for which we have long term Insight data.

Returns a dict with "mac", "linux" and "windows" and their count tuples [1,7,30].

getJob(*jobId*)
Get a specific job.

Parameters **jobId** (*str*) – job ID of the job to get.

Returns a Job object.

getJobs(*startTime*, *endTime*, *limit=None*, *sid=None*)
Get all the jobs in an organization in a time window.

Parameters

- **startTime** (*int*) – second epoch of the start of the time window.
- **endTime** (*int*) – second epoch of the end of the time window.
- **limit** (*int*) – optional maximum number of jobs to return.
- **sid** (*str*) – optionally only return jobs that relate to this sensor ID.

Returns a Job object.

getObjectInformation(*objType*, *objName*, *info*, *isCaseSensitive=True*, *isWithWildcards=False*, *limit=None*, *isPerObject=None*)
Get information about an object (indicator) using Insight (retention) data.

Parameters

- **objType** (*str*) – the object type to query for, one of: user, domain, ip, hash, file_path, file_name.
- **objName** (*str*) – the name of the object to query for, like "cmd.exe".
- **info** (*str*) – the type of information to query for, one of: summary, locations.
- **isCaseSensitive** (*bool*) – False to ignore case in the object name.
- **isWithWildcards** (*bool*) – True to enable use of "%" wildcards in the object name.
- **limit** (*int*) – optional maximum number of sensors/logs to report, or None for LimaCharlie default.
- **isPerObject** (*bool*) – if set, specifies if the results should be groupped per object when a wildcard is present.

Returns a dict with the requested information.

getOrgConfig(*configName*)
Get the value of a per-organization config.

Parameters **configName** (*str*) – name of the config to get.

Returns String value of the configuration.

getOrgURLs()
Get the URLs used by various resources in the organization.

Returns Dictionary of resource types to URLs.

getSensorsWithHostname(*hostnamePrefix*)
Get the list of sensor IDs and hostnames that match the given prefix.

Parameters **hostnamePrefix** (*str*) – a hostname prefix to search for.

Returns List of (sid, hostname).

getSensorsWithIp(*ip*, *start*, *end*)
Get the list of sensor IDs that used the given IP during the time range.

Parameters

- **ip** (*str*) – the IP address used.
- **start** (*int*) – beginning of the time range to look for.
- **end** (*int*) – end of the time range to look for.

Returns List of sid.

getSubscriptions()
Get the list of resources the organization is subscribed to.

getUserPermissions()
Get the list of users and their permissions.

getUsers()
Get the list of users in the organization.

hosts(*hostname_expr*)
Get the Sensor objects for hosts matching a hostname expression.

Parameters **hostname_expr** (*str*) – hostname prefix to look for.

Returns a list of Sensor IDs matching the hostname expression.

isInsightEnabled()
Check to see if Insight (retention) is enabled on this organization.

Returns True if Insight is enabled.

make_interactive()
Enables interactive mode on this instance if it was not created with is_interactive.

outputs()
Get the list of all Outputs configured for the Organization.

Returns a list of Output descriptions (JSON).

removeApiKey(*keyHash*)
Remove an API key from an organization.

Parameters **keyHash** (*str*) – key hash of the key to remove.

removeUser(*email*)
Remove user from an organization.

Parameters **email** (*str*) – email of the user to remove.

removeUserPermission(*email*, *permission*)
Remove user from an organization.

Parameters

- **email** (*str*) – email of the user to remove.
- **permission** (*str*) – permission to remove from the user.

rules(*namespace=None*)
Get the list of all Detection & Response rules for the Organization.

Parameters **namespace** (*str*) – optional namespace to operator on, defaults to "general".

Returns a list of D&R rules (JSON).

sensor(*sid*, *inv_id=None*)
Get a Sensor object for the specific Sensor ID.

The sensor may or may not be online.

Parameters

- **sid** (*uuid str*) – the Sensor ID to represent.
- **inv_id** (*str*) – investigation ID to add to all actions done using this object.

Returns a Sensor object.

sensors(*inv_id=None*, *is_next=False*)
Gets all Sensors in the Organization.

The sensors may or may not be online.

Parameters **inv_id** (*str*) – investigation ID to add to all actions done using these objects.

Returns a generator of Sensor objects.

sensorsWithTag(*tag*)
Get a list of sensors that have the matching tag.

Parameters **tag** (*str*) – a tag to look for.

Returns a list of Sensor objects.

serviceRequest(*serviceName*, *data*, *isAsynchronous=False*)
Issue a request to a Service.

Parameters

- **serviceName** (*str*) – the name of the Service to task.
- **data** (*dict*) – JSON data to send to the Service as a request.
- **isAsynchronous** (*bool*) – if set to False, wait for data from the Service and return it.

Returns Dict with general success, or data from Service if isSynchronous.

setIngestionKey(*name*)
Set (or reset) an Ingestion key.

Parameters **name** (*str*) – name of the Ingestion key to set.

Returns Dictionary with the key name and value.

setOrgQuota(*quota*)
Set a new sensor quota for the organization.

Parameters **quota** (*int*) – the new quota value.

shutdown()
Shut down any active mechanisms like interactivity.

subscribeToResource(*name*)
Subscribe the organization to the specific resource.

Parameters **name** (*str*) – name of the resource like lookup/test-res.

testAuth(*permissions=[]*)
Tests authentication with limacharlie.io.

Parameters **permissions** (*list*) – optional list of permissions validate we have.

Returns a boolean indicating whether authentication succeeded.

unsubscribeFromResource(*name*)
Unsubscribe the organization from the specific resource.

Parameters **name** (*str*) – name of the resource like lookup/test-res.

whoAmI()
Query the API to see which organizations we are authenticated for.

Returns A list of organizations and permissions, or a dictionary of organizations with the related permissions.

1.6 limacharlie.Payloads module

class limacharlie.Payloads.**Payloads**(*manager*)
Bases: object

Helper object to manage executable Payloads for sensors.

create(*name*, *payloadPath=None*, *payloadContent=None*)
Create a new payload.

Parameters

- **name** (*str*) – the name of the payload to create.
- **payloadPath** (*str*) – path to the file containing the payload.
- **payloadContent** (*bytes*) – content of the new payload.

delete(*name*)
Delete a payload.

Parameters **name** (*str*) – the name of the payload to delete.

get(*name*)
Get a specific payload content.

Parameters **name** (*str*) – the name of the payload to get.

list()
List all available payloads.

1.7 limacharlie.Replay module

class limacharlie.Replay.**Replay**(*manager*, *maxTimeWindow=86400*, *maxConcurrent=10*, *isInteractive=False*)

Bases: object

Interface to query historical sensor data in Insight with specific D&R rules.

scanEntireOrg(*startTime*, *endTime*, *ruleName=None*, *ruleContent=None*, *isRunTrace=False*, *limitEvent=None*, *limitEval=None*, *isIgnoreState=False*)
Scan an entire organization's data with a D&R rule.

Parameters

- **startTime** (*int*) – seconds epoch to start scanning at.
- **endTime** (*int*) – seconds epoch to stop scanning at.
- **ruleName** (*str*) – the name of an existing D&R rule to use.
- **ruleContent** (*dict*) – D&R rule to use to scan, with a "detect" key and a "respond" key.
- **isRunTrace** (*bool*) – if True, generate a trace of the evaluation.
- **limitEvent** (*int*) – approximately limit the number of events evaluated.
- **limitEval** (*int*) – approximately limit the number of rule evaluations.
- **isIgnoreState** (*bool*) – if True, parallelize processing of single sensors to increase performance but limit effectiveness of stateful detection.

Returns a dict containing results of the query.

scanEvents(*events*, *ruleName=None*, *ruleContent=None*, *isRunTrace=False*, *limitEvent=None*, *limitEval=None*)
Scan the specific events with a D&R rule.

Parameters

- **events** (*list*) – list of events to scan.
- **ruleName** (*str*) – the name of an existing D&R rule to use.

- **ruleContent** (*dict*) – D&R rule to use to scan, with a "detect" key and a "respond" key.
- **isRunTrace** (*bool*) – if True, generate a trace of the evaluation.
- **limitEvent** (*int*) – approximately limit the number of events evaluated.
- **limitEval** (*int*) – approximately limit the number of rule evaluations.

Returns a dict containing results of the query.

scanHistoricalSensor(*sid*, *startTime*, *endTime*, *ruleName=None*, *ruleContent=None*, *isRunTrace=False*, *limitEvent=None*, *limitEval=None*, *isIgnoreState=False*)

Scan a specific sensor's data with a D&R rule.

Parameters

- **sid** (*str*) – sensor ID to scan.
- **startTime** (*int*) – seconds epoch to start scanning at.
- **endTime** (*int*) – seconds epoch to stop scanning at.
- **ruleName** (*str*) – the name of an existing D&R rule to use.
- **ruleContent** (*dict*) – D&R rule to use to scan, with a "detect" key and a "respond" key.
- **isRunTrace** (*bool*) – if True, generate a trace of the evaluation.
- **limitEvent** (*int*) – approximately limit the number of events evaluated.
- **limitEval** (*int*) – approximately limit the number of rule evaluations.
- **isIgnoreState** (*bool*) – if True, parallelize processing of single sensors to increase performance but limit effectiveness of stateful detection.

Returns a dict containing results of the query.

validateRule(*ruleContent=None*)

Validate a D&R rule compiles properly.

Parameters **ruleContent** (*dict*) – D&R rule to use to scan, with a "detect" key and a "respond" key.

Returns a dict containing results of the query.

1.8 limacharlie.Replicants module

class limacharlie.Replicants.**Dumper**(*manager*)

Bases: limacharlie.Replicants._Replicant

Memory dumper service object.

dump(*sid*)

Dump the full memory of a given host.

Parameters **sid** (*str*) – sensor ID to sweep.

class limacharlie.Replicants.**Exfil**(*manager*)

Bases: limacharlie.Replicants._Replicant

Exfil control service manager object.

addEventRule(*ruleName*, *events=[]*, *tags=[]*, *platforms=[]*)
Add an event rule describing events sent to the cloud in real-time.

Parameters

- **ruleName** (*str*) – name of the rule to add.
- **events** (*list of str*) – list of event names to send in real-time.
- **tags** (*list of str*) – list of tags sensors must posses for this rule to apply.
- **platforms** (*list of str*) – list of platform names this applies to.

addWatchRule(*ruleName*, *event*, *operator*, *value*, *path=[]*, *tags=[]*, *platforms=[]*)
Add a watch rule to send matching events to the cloud in real-time.

Parameters

- **ruleName** (*str*) – name of the watch rule to add.
- **event** (*str*) – name of the event this rule applies to.
- **operator** (*str*) – comparison operator name to determine match.
- **value** (*str*) – value to compare to for matching.
- **path** (*list of str*) – path within the event to compare the value of, without a leading "event".
- **tags** (*list of str*) – list of tags sensors must posses for this rule to apply.
- **platforms** (*list of str*) – list of platform names this applies to.

getRules()
Get the exfil rules in effect.

Returns Dict of rules.

removeEventRule(*ruleName*)
Remove an event rule.

Parameters **ruleName** (*str*) – name of the rule to remove.

removeWatchRule(*ruleName*)
Remove a watch rule.

Parameters **ruleName** (*str*) – name of the rule to remove.

class limacharlie.Replicants.**Integrity**(*manager*)
Bases: limacharlie.Replicants._Replicant

File and Registry Integrity Monitoring (FIM) service manager object.

addRule(*ruleName*, *patterns=[]*, *tags=[]*, *platforms=[]*)
Add an FIM rule.

Parameters

- **ruleName** (*str*) – name of the rule to add.
- **patterns** (*list of str*) – list of file/registry patterns to monitor.
- **tags** (*list of str*) – list of tags sensors must posses for this rule to apply.
- **platforms** (*list of str*) – list of platform names this rule applies to.

getRules()
Get FIM rules in effect.

Returns Dict of rules.

removeRule(*ruleName*)
Remove an FIM rule.

Parameters **ruleName** (*str*) – name of the rule to remove.

class limacharlie.Replicants.**Logging**(*manager*)
Bases: limacharlie.Replicants._Replicant

Logging service manager object.

addRule(*ruleName*, *patterns=[]*, *tags=[]*, *platforms=[]*, *isDeleteAfter=False*, *isIgnoreCert=False*)
Add a Log collection rule.

Parameters

- **ruleName** (*str*) – name of the rule to add.
- **patterns** (*list of str*) – list of file patterns describing Logs to monitor and retrieve.
- **tags** (*list of str*) – list of tags sensors must posses for this rule to apply.
- **platforms** (*list of str*) – list of platform names this rule applies to.
- **isDeleteAfter** (*bool*) – if True, delete the Log after retrieval.
- **isIgnoreCert** (*bool*) – if True, sensor ignores SSL cert errors during log upload.

getRules()
Get the Log collection rules in effect.

removeRule(*ruleName*)
Remove a Log collection rule.

Parameters **ruleName** (*str*) – name of the rule to remove.

class limacharlie.Replicants.**ReliableTasking**(*manager*)
Bases: limacharlie.Replicants._Replicant

Reliable Tasking service object.

getTasks(*sid=None*, *tag=None*)
Issue a task for a set of sensors even if offline.

Parameters

- **sid** (*str*) – optional sensor ID to get the tasks for or '*' for all.
- **tag** (*str*) – optional tag to select sensors to get the tasks for.

task(*task*, *sid=None*, *tag=None*, *ttl=None*)
Issue a task for a set of sensors even if offline.

Parameters

- **task** (*str*) – actual task command line to send.
- **sid** (*str*) – optional sensor ID to task or '*' for all.
- **tag** (*str*) – optional tag to select sensors to send the task to.
- **ttl** (*int*) – optional number of seconds before unsent tasks expire, defaults to a week.

class limacharlie.Replicants.**Replay**(*manager*)
Bases: limacharlie.Replicants._Replicant

Replay service manager object.

runJob(*startTime*, *endTime*, *sid=None*, *ruleName=None*, *ruleContent=None*)
Run a Replay service job.

Parameters

- **startTime** (*int*) – epoch start time to replay.
- **endTime** (*int*) – epoch end time to replay.
- **sid** (*str*) – sensor ID to replay the data from.
- **ruleName** (*str*) – optional name of an existing D&R rule to replay.
- **ruleContent** (*dict*) – optional content of a D&R rule to replay.

class limacharlie.Replicants.**Responder**(*manager*)
Bases: limacharlie.Replicants._Replicant

Responder service manager object.

sweep(*sid*)
Perform a sweep of a given host.

Parameters **sid** (*str*) – sensor ID to sweep.

class limacharlie.Replicants.**Yara**(*manager*)
Bases: limacharlie.Replicants._Replicant

Yara service manager object.

addRule(*ruleName*, *sources=[]*, *tags=[]*, *platforms=[]*)
Add a constant Yara scanning rule.

Parameters

- **ruleName** (*str*) – name of the rule to add.
- **sources** (*list of str*) – list of sources this rule should scan with.
- **tags** (*list of str*) – list of tags sensors must posses for this rule to apply.
- **platforms** (*str of str*) – list of platform names this rule applies to.

addSource(*sourceName*, *source*)
Add a Yara signature source.

Parameters

- **sourceName** (*str*) – name of the source to add.
- **source** (*str*) – source URL for the Yara signature(s).

getRules()
Get the constant Yara scanning rules in effect.

Returns Dict of rules.

getSources()
Get the Yara signature sources.

Returns Dict of sources.

removeRule(*ruleName*)
Remove a constant Yara scanning rule.

Parameters **ruleName** (*str*) – name of the rule to remove.

removeSource(*sourceName*)
Remove a Yara rule source.

Parameters **sourceName** (*str*) – name of the source to remove.

scan(*sid*, *sources*)
Perform an ad-hoc scan of a sensor with Yara signatures.

Parameters

- **sid** (*str*) – sensor ID to scan.
- **sources** (*list of str*) – list of source Yara signature names to use in the scan.

1.9 limacharlie.Search module

class limacharlie.Search.**Search**(*environments=None*, *output='-'*)
Bases: object

Helper object to perform cross-organization IOC searches.

query(*iocType*, *iocName*, *info*, *isCaseInsensitive=False*, *isWithWildcards=False*, *limit=None*, *isPerIoc=False*)
Performa a search.

Parameters

- **iocType** (*str*) – type of IOC to search for.
- **iocName** (*str*) – name of the IOC to search for.
- **info** (*str*) – information type to retrieve.
- **isCaseInsensitive** (*bool*) – if True, search for IOC in a case insensitive way.
- **isWithWildcards** (*bool*) – if True, use "%" as a wildcard in the IOC name.
- **limit** (*int*) – optional maximum number of sensors/logs to report about, otherwise defaults to internal LimaCharlie limit.
- **isPerIoc** (*bool*) – if the search has wildcards, return results grouped per individual ioc.

Returns Dict of requested information.

1.10 limacharlie.Sensor module

class limacharlie.Sensor.**Sensor**(*manager*, *sid*)
Bases: object

Representation of a limacharlie.io Sensor.

delete()
Delete the sensor. It will not be able to connect to the cloud anymore, but will not be uninstalled.abs

getHistoricEvents(*start*, *end*, *limit=None*, *eventType=None*, *isForward=True*)
Get the events for this sensor between the two times, requires Insight (retention) enabled.

Parameters

- **start** (*int*) – start unix (seconds) timestamp to fetch events from.

- **end** (*int*) – end unix (seconds) timestamp to feth events to.
- **limit** (*int*) – maximum number of events to return.
- **eventType** (*str*) – return events only of this type.
- **isForward** (*bool*) – return events in ascending order.

Returns a generator of events.

getHistoricOverview(*start*, *end*)
Get a list of timestamps representing where sensor data is available in Insight (retention).

Parameters

- **start** (*int*) – start unix (seconds) timestamp to look for events from.
- **end** (*int*) – end unix (seconds) timestamp to look for events to.

Returns a list of timestamps.

getInfo()
Get basic information on the Sensor.

Returns high level information on the Sensor.

getObjectTimeline(*start*, *end*, *bucketing='day'*, *onlyTypes=None*)
Get summarized information about timeline of Objects (IOCs) for this host.

Parameters

- **start** (*int*) – start time (unix seconds epoch) of the period to search.
- **end** (*int*) – end time (unix seconds epoch) of the period to search.
- **bucketing** (*str*) – granularity of the timeline, one of "hour", "day", "week", "month".
- **onlyTypes** (*list*) – list of object types to look for, all if undefined.

Returns Dict of timelines per type and object.

getTags()
Get Tags applied to the Sensor.

Returns the list of Tags currently applied.

hostname()
Get the hostname of this sensor.

Returns a string of the hostname.

isChrome()
Checks if the sensor is on Chrome.

Returns True if the sensor is Chrome.

isChromeOS()
Checks if the sensor is on ChromeOS.

Returns True if the sensor is on ChromeOS.

isDataAvailableFor(*timestamp*)
Check if data is available in Insight for this sensor at this specific time.

Parameters **timestamp** (*int*) – time (unix seconds epoch) to check for events.

Returns True if data is available.

isIsolatedFromNetwork()
Determine if the given sensor is marked to be isolated from the network.

Returns True if isolated.

isLinux()
Checks if the sensor is a Linux OS.

Returns True if the sensor is Linux.

isMac()
Checks if the sensor is a Mac OS.

Returns True if the sensor is Mac.

isOnline()
Checks if the sensor is currently online.

Returns True if the sensor is connected to the cloud right now.

isWindows()
Checks if the sensor is a Windows OS.

Returns True if the sensor is Windows.

isolateNetwork()
Mark the sensor for network isolation (persistent).

rejoinNetwork()
Remove the sensor from network isolation (persistent).

request(*tasks*)
Send a task (or list of tasks) to the Sensor and returns a FutureResults where the results will be sent; requires Manager is_interactive.

Parameters **tasks** (*str or list of str*) – tasks to send in the command line format described in official documentation.

Returns a FutureResults object.

setInvId(*inv_id*)
Set an investigation ID to be applied to all actions done using the object.

Parameters **inv_id** (*str*) – investigation ID to propagate.

simpleRequest(*tasks*, *timeout=30*, *until_completion=False*)
Make a request to the sensor assuming a single response.

Parameters

- **tasks** (*str or list of str*) – tasks to send in the command line format described in official documentation.
- **timeout** (*int*) – number of seconds to wait for responses.
- **until_completion** (*bool or callback*) – if True, wait for completion receipts from the sensor, or callback for each response.

Returns a single event (if tasks was a single task), a list of events (if tasks was a list), or None if not received.

tag(*tag*, *ttl*)
Apply a Tag to the Sensor.

Parameters

- **tag** (*str*) – Tag to apply.
- **ttl** (*int*) – number of seconds the Tag should remain applied.

Returns the REST API response (JSON).

task(*tasks*, *inv_id=None*)
Send a task (or list of tasks) to the Sensor.

Parameters

- **tasks** (*str or list of str*) – tasks to send in the command line format described in official documentation.
- **inv_id** (*str*) – investigation ID to propagate.

Returns the REST API response (JSON).

untag(*tag*)
Remove a Tag from the Sensor.

Parameters **tag** (*str*) – Tag to remove.

Returns the REST API response (JSON).

waitToComeOnline(*timeout*)
Wait for the sensor to be online.

Parameters **timeout** (*int*) – number of seconds to wait up to

Returns True if sensor is back or False if timeout

1.11 limacharlie.SpotCheck module

class limacharlie.SpotCheck.**SpotCheck**(*oid*, *secret_api_key*, *cb_check*, *cb_on_start_check=None*, *cb_on_check_done=None*, *cb_on_offline=None*, *cb_on_error=None*, *n_concurrent=1*, *n_sec_between_online_checks=60*, *extra_params={}*, *is_windows=True*, *is_linux=True*, *is_macos=True*, *is_chrome=True*, *tags=None*)
Bases: object

Representation of the process of looking for various Indicators of Compromise on the fleet.

start()
Start the SpotCheck process, returns immediately.

stop()
Stop the SpotCheck process, returns once activity has stopped.

wait(*timeout=None*)
Wait for SpotCheck to be complete, or timeout occurs.

Parameters **timeout** (*float*) – if specified, number of seconds to wait for SpotCheck to complete.

Returns True if SpotCheck is finished, False if a timeout was specified and reached before the SpotCheck is done.

1.12 limacharlie.Spout module

class limacharlie.Spout.**Spout**(*man*, *data_type*, *is_parse=True*, *max_buffer=1024*, *inv_id=None*, *tag=None*, *cat=None*, *sid=None*, *extra_params={}*)

Bases: object

Listener object to receive data (Events, Detects or Audit) from a limacharlie.io Organization in pull mode.

getDropped()
Get the number of messages dropped because queue was full.

registerFutureResults(*tracking_id*, *future*, *ttl=3600*)
Register a FutureResults to receive events coming with a specific tracking ID and investigation ID.

Parameters

- **tracking_id** (*str*) – the full value of the investigation_id field to match on, including the custom tracking after the "/".
- **future** (*limacharlie.FutureResults*) – future to receive the events.
- **ttl** (*int*) – number of seconds this future should be tracked.

resetDroppedCounter()
Reset the counter of dropped messages.

shutdown()
Stop receiving data.

1.13 limacharlie.Sync module

exception limacharlie.Sync.**LcConfigException**

Bases: exceptions.Exception

class limacharlie.Sync.**Sync**(*oid=None*, *env=None*, *manager=None*)

Bases: object

Sync object to fetch and apply configs to and from organizations.

fetch(*toConfigFile*, *isNoRules=False*, *isNoFPs=False*, *isNoOutputs=False*, *isNoIntegrity=False*, *isNoLogging=False*, *isNoExfil=False*, *isNoResources=False*)
Retrieves the effective configuration in the cloud to a local config file.

Parameters **toConfigFile** (*str, dict*) – the path to the local config file or dict where to store config.

push(*fromConfigFile*, *isForce=False*, *isDryRun=False*, *isNoRules=False*, *isNoFPs=False*, *isNoOutputs=False*, *isNoIntegrity=False*, *isNoLogging=False*, *isNoExfil=False*, *isNoResources=False*)
Apply the configuratiion in a local config file to the effective configuration in the cloud.

Users should favor using the "push<Type>()" convenience functions instead of the main "push()" function as they are safer to use in the event support for new data-types is added to the "push()" function.

Parameters

- **fromConfigFile** (*str/dict*) – the path to the config file or dict of a config file content.
- **isForce** (*boolean*) – if True will remove configurations in the cloud that are not present in the local file.
- **isDryRun** (*boolean*) – if True will only simulate the effect of a push.

Returns a generator of changes as tuple (changeType, dataType, dataName).

pushExfil (*fromConfigFile*, *isForce=False*, *isDryRun=False*)
Convenience function to push the Exfil configs in a local config file to the effective configuration in the cloud.

Parameters

- **fromConfigFile** (*str/dict*) – the path to the config file or dict of a config file content.
- **isForce** (*boolean*) – if True will remove configurations in the cloud that are not present in the local file.
- **isDryRun** (*boolean*) – if True will only simulate the effect of a push.

Returns a generator of changes as tuple (changeType, dataType, dataName).

pushFPs (*fromConfigFile*, *isForce=False*, *isDryRun=False*)
Convenience function to push the FP rules in a local config file to the effective configuration in the cloud.

Parameters

- **fromConfigFile** (*str/dict*) – the path to the config file or dict of a config file content.
- **isForce** (*boolean*) – if True will remove configurations in the cloud that are not present in the local file.
- **isDryRun** (*boolean*) – if True will only simulate the effect of a push.

Returns a generator of changes as tuple (changeType, dataType, dataName).

pushIntegrity (*fromConfigFile*, *isForce=False*, *isDryRun=False*)
Convenience function to push the Integrity configs in a local config file to the effective configuration in the cloud.

Parameters

- **fromConfigFile** (*str/dict*) – the path to the config file or dict of a config file content.
- **isForce** (*boolean*) – if True will remove configurations in the cloud that are not present in the local file.
- **isDryRun** (*boolean*) – if True will only simulate the effect of a push.

Returns a generator of changes as tuple (changeType, dataType, dataName).

pushLogging (*fromConfigFile*, *isForce=False*, *isDryRun=False*)
Convenience function to push the Logging configs in a local config file to the effective configuration in the cloud.

Parameters

- **fromConfigFile** (*str/dict*) – the path to the config file or dict of a config file content.
- **isForce** (*boolean*) – if True will remove configurations in the cloud that are not present in the local file.
- **isDryRun** (*boolean*) – if True will only simulate the effect of a push.

Returns a generator of changes as tuple (changeType, dataType, dataName).

pushOutputs(*fromConfigFile*, *isForce=False*, *isDryRun=False*)
Convenience function to push the outputs in a local config file to the effective configuration in the cloud.

Parameters

- **fromConfigFile** (*str/dict*) – the path to the config file or dict of a config file content.
- **isForce** (*boolean*) – if True will remove configurations in the cloud that are not present in the local file.
- **isDryRun** (*boolean*) – if True will only simulate the effect of a push.

Returns a generator of changes as tuple (changeType, dataType, dataName).

pushResources(*fromConfigFile*, *isForce=False*, *isDryRun=False*)
Convenience function to push the Resources configs in a local config file to the effective configuration in the cloud.

Parameters

- **fromConfigFile** (*str/dict*) – the path to the config file or dict of a config file content.
- **isForce** (*boolean*) – if True will remove configurations in the cloud that are not present in the local file.
- **isDryRun** (*boolean*) – if True will only simulate the effect of a push.

Returns a generator of changes as tuple (changeType, dataType, dataName).

pushRules(*fromConfigFile*, *isForce=False*, *isDryRun=False*)
Convenience function to push the D&R rules in a local config file to the effective configuration in the cloud.

Parameters

- **fromConfigFile** (*str/dict*) – the path to the config file or dict of a config file content.
- **isForce** (*boolean*) – if True will remove configurations in the cloud that are not present in the local file.
- **isDryRun** (*boolean*) – if True will only simulate the effect of a push.

Returns a generator of changes as tuple (changeType, dataType, dataName).

1.14 limacharlie.Webhook module

class limacharlie.Webhook.**Webhook**(*secret_key*)
Bases: object

Helper class for various activities related to webhooks from limacharlie.io.

isSignatureValid(*dataFromHook*, *signature*)
Validate the signature from a webhook.

Parameters

- **dataFromHook** (*str*) – string found in the "data" element from the webhook.
- **signature** (*str*) – signature from the "Lc-Signature" header of the webhook.

Returns a boolean where True means the webhook data and signature are valid.

1.15 limacharlie.utils module

class limacharlie.utils.**FutureResults**
Bases: object

Represents a Future promise of results from a task sent to a Sensor.

getNewResponses(*timeout=None*)
Get new responses available, blocking for up to timeout seconds.

Parameters **timeout** (*float*) – number of seconds to block for new results.

Returns a list of new results, or an empty list if timeout is reached.

exception limacharlie.utils.**LcApiException**
Bases: exceptions.Exception

Exception type used for various errors in the LimaCharlie SDK.

limacharlie.utils.**enhanceEvent**(*evt*)
Wrap an event with an _enhancedDict providing utility functions getOne() and getAll().

Parameters **evt** (*dict*) – event to wrap.

Returns wrapped event.

limacharlie.utils.**parallelExec**(*f*, *objects*, *timeout=None*, *maxConcurrent=None*)
Execute a function on a list of objects in parallel.

Parameters

- **f** (*callable*) – function to apply to each object.
- **objects** (*iterable*) – list of objects to apply the function on.
- **timeout** (*int*) – maximum number of seconds to wait for collection of calls.
- **maxConcurrent** (*int*) – maximum number of function application to do concurrently.

Returns list of return values (or Exception if an exception occured).

1.16 Module contents

limacharlie API for limacharlie.io

Python Module Index

l

Index

R

S

T

U

V

W

Y