
Icservice Documentation

Release 1.9.5

Refraction Point, Inc

Jan 22, 2023

Contents

1	lcservice package	3
1.1	Subpackages	3
1.2	Submodules	3
1.3	lcservice.jobs module	3
1.4	lcservice.service module	4
1.5	lcservice.simulator module	7
1.6	Module contents	7
	Python Module Index	9
	Index	11

LimaCharlie.io External Service - Security Infrastructure as a Service platform.

<https://limacharlie.io>

View code, installation instructions and other usage information: <https://github.com/refractionpoint/lcservice/>

Contents:

1.1 Subpackages

1.1.1 lcservice.servers package

Submodules

lcservice.servers.cherrypy module

`lcservice.servers.cherrypy.ServeCherryPy` (*service*, *interface*='0.0.0.0', *port*=80, *options*={})

Serve a service using the cherrypy web server.

lcservice.servers.cloud_function module

Module contents

1.2 Submodules

1.3 lcservice.jobs module

class `lcservice.jobs.HexDump` (*caption*, *data*)
Bases: object

Abstraction for adding an attachment to a Job update to be displayed as a hex dump.

class `lcservice.jobs.Job` (*jobId*=None)
Bases: object

Abstraction for reporting Job updates to LimaCharlie.

addSensor (*sid*)

Add a sensor ID to this job, indicating it is somehow involved in the job. :param sid: sensor ID to add.

close ()

Indicate the job is now finished.

getId ()

Get the job's ID.

narrate (*message, attachments=[], isImportant=False*)

Give an update message to the job. :param message: simple message describing the update. :param attachments: optional list of attachments add along this update. :param isImportant: if True, this update will be highlighted in the job log as particularly important.

setCause (*cause*)

Set the cause for the creation of the job. :param cause: the cause string to set.

class `lcservice.jobs.JsonData` (*caption, data*)

Bases: `object`

Abstraction for adding an attachment to a Job update to be displayed as a JSON data.

class `lcservice.jobs.Table` (*caption, headers, rows=[]*)

Bases: `object`

Abstraction for adding an attachment to a Job update to be displayed as a table.

addRow (*fields*)

Add a row to the table. :param fields: list of fields representing a single table row.

length ()

Get the number of rows in the table.

class `lcservice.jobs.YamlData` (*caption, data*)

Bases: `object`

Abstraction for adding an attachment to a Job update to be displayed as YAML data.

1.4 lcservice.service module

class `lcservice.service.InteractiveService` (*serviceName, originSecret, isTraceComms=False*)

Bases: `lcservice.service.Service`

InteractiveService provide for asynchronous tasking of sensors.

Services inheriting from InteractiveService require at a minimum the following permissions: “dr.list.replicant”, “dr.del.replicant”, “dr.set.replicant” and “sensor.task”.

class `lcservice.service.Service` (*serviceName, originSecret, isTraceComms=False*)

Bases: `object`

Main class implementing core service functionality.

delay (*inDelay, func, *args, **kw_args*)

Delay the execution of a function.

Only use if your execution environment allows for asynchronous execution (like a normal container). Some environments like Cloud Functions (Lambda) or Google Cloud Run may not allow for execution outside of the processing of inbound queries.

Parameters

- **inDelay** – the number of seconds to execute into
- **func** – the function to call
- **args** – positional arguments to the function
- **kw_args** – keyword arguments to the function

every12HourGlobally (*lc, oid, request*)

Called every 12 hours once per service.

every12HourPerOrg (*lc, oid, request*)

Called every 12 hours for every organization subscribed.

every12HourPerSensor (*lc, oid, request*)

Called every 12 hours once per sensor.

every1HourGlobally (*lc, oid, request*)

Called every hour once per service.

every1HourPerOrg (*lc, oid, request*)

Called every hour for every organization subscribed.

every1HourPerSensor (*lc, oid, request*)

Called every 1 hours once per sensor.

every24HourGlobally (*lc, oid, request*)

Called every 24 hours once per service.

every24HourPerOrg (*lc, oid, request*)

Called every 24 hours for every organization subscribed.

every24HourPerSensor (*lc, oid, request*)

Called every 24 hours once per sensor.

every30DayGlobally (*lc, oid, request*)

Called every 30 days once per service.

every30DayPerOrg (*lc, oid, request*)

Called every 30 days for every organization subscribed.

every30DayPerSensor (*lc, oid, request*)

Called every 30 days once per sensor.

every3HourGlobally (*lc, oid, request*)

Called every 3 hours once per service.

every3HourPerOrg (*lc, oid, request*)

Called every 3 hours for every organization subscribed.

every3HourPerSensor (*lc, oid, request*)

Called every 3 hours once per sensor.

every7DayGlobally (*lc, oid, request*)

Called every 7 days once per service.

every7DayPerOrg (*lc, oid, request*)

Called every 7 days for every organization subscribed.

every7DayPerSensor (*lc, oid, request*)

Called every 7 days once per sensor.

log (*msg, data=None*)

Log a message to stdout.

Parameters

- **msg** – message to log.
- **data** – optional JSON data to include in log.

logCritical (*msg*)

Log a message to stderr.

Parameters **msg** – critical message to log.

onDeploymentEvent (*lc, oid, request*)

Called when a deployment event is received.

onDetection (*lc, oid, request*)

Called when a detection is received for an organization.

onLogEvent (*lc, oid, request*)

Called when a log event is received.

onNewSensor (*lc, oid, request*)

Called every 24 hours once per service.

onOrgInstalled (*lc, oid, request*)

Called when a new organization subscribes to this service.

onOrgUninstalled (*lc, oid, request*)

Called when an organization unsubscribes from this service.

onRequest (*lc, oid, request*)

Called when a request is made for the service by the organization.

onServiceError (*lc, oid, request*)

Called when LC cloud encounters an error with this service.

onShutdown ()

Called when the service is about to shut down.

onStartup ()

Called when the service is first instantiated.

parallelExec (*f, objects, timeout=None, maxConcurrent=None*)

Execute a function on a list of objects in parallel. :param f: function to apply to each object. :type f: callable :param objects: list of objects to apply the function on. :type objects: iterable :param timeout: maximum number of seconds to wait for collection of calls. :type timeout: int :param maxConcurrent: maximum number of function application to do concurrently. :type maxConcurrent: int

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

parallelExecEx (*f, objects, timeout=None, maxConcurrent=None*)

Applies a function to N objects in parallel in up to maxConcurrent threads and waits to return the generated results.

Parameters

- **f** – the function to apply
- **objects** – a dict of key names pointing to the objects to apply using f
- **timeouts** – number of seconds to wait for results, or None for indefinitely
- **maxConcurrent** – maximum number of concurrent tasks

Returns a generator of tuples(key name, f(object)), or Exception if one occurred.

publishResource (*resourceName, resourceCategory, resourceData*)

Make a resource with this name available to LimaCharlie requests.

Parameters

- **resourceName** – the name of the resource to make available.
- **resourceCategory** – the category of the resource (like “detect” or “lookup”).
- **resourceData** – the resource content.

response (*isSuccess=True, isDoRetry=False, data={}, error=None, jobs=[]*)

Generate a custom response JSON message.

Parameters

- **isSuccess** – True for success, False for failure.
- **isDoRetry** – if True indicates to LimaCharlie to retry the request.
- **data** – JSON data to include in the response.
- **error** – an error string to report to the organization.
- **jobs** – new Jobs or updates to existing Jobs.

responseNotImplemented ()

Generate a pre-made response indicating the callback is not implemented.

schedule (*delay, func, *args, **kw_args*)

Schedule a recurring function.

Only use if your execution environment allows for asynchronous execution (like a normal container). Some environments like Cloud Functions (Lambda) or Google Cloud Run may not allow for execution outside of the processing of inbound queries.

Parameters

- **delay** – the number of seconds interval between calls
- **func** – the function to call at interval
- **args** – positional arguments to the function
- **kw_args** – keyword arguments to the function

setRequestParameters (*params*)

Set the supported request parameters, with type and description.

Parameters **params** – dictionary of the parameter definitions, see official README for exact definition.

subscribeToDetect (*detectName*)

Subscribe this service to the specific detection names of all subscribed orgs.

Parameters **detectName** – name of the detection to subscribe to.

1.5 Icservice.simulator module

1.6 Module contents

Reference implementation for LimaCharlie.io services.

I

- `lcservice`, 7
- `lcservice.jobs`, 3
- `lcservice.servers`, 3
- `lcservice.servers.cherrypy`, 3
- `lcservice.service`, 4
- `lcservice.simulator`, 7

A

`addRow()` (*lcservice.jobs.Table method*), 4
`addSensor()` (*lcservice.jobs.Job method*), 3

C

`close()` (*lcservice.jobs.Job method*), 4

D

`delay()` (*lcservice.service.Service method*), 4

E

`every12HourGlobally()` (*lcservice.service.Service method*), 5
`every12HourPerOrg()` (*lcservice.service.Service method*), 5
`every12HourPerSensor()` (*lcservice.service.Service method*), 5
`every1HourGlobally()` (*lcservice.service.Service method*), 5
`every1HourPerOrg()` (*lcservice.service.Service method*), 5
`every1HourPerSensor()` (*lcservice.service.Service method*), 5
`every24HourGlobally()` (*lcservice.service.Service method*), 5
`every24HourPerOrg()` (*lcservice.service.Service method*), 5
`every24HourPerSensor()` (*lcservice.service.Service method*), 5
`every30DayGlobally()` (*lcservice.service.Service method*), 5
`every30DayPerOrg()` (*lcservice.service.Service method*), 5
`every30DayPerSensor()` (*lcservice.service.Service method*), 5
`every3HourGlobally()` (*lcservice.service.Service method*), 5
`every3HourPerOrg()` (*lcservice.service.Service method*), 5

`every3HourPerSensor()` (*lcservice.service.Service method*), 5
`every7DayGlobally()` (*lcservice.service.Service method*), 5
`every7DayPerOrg()` (*lcservice.service.Service method*), 5
`every7DayPerSensor()` (*lcservice.service.Service method*), 5

G

`getId()` (*lcservice.jobs.Job method*), 4

H

`HexDump` (*class in lcservice.jobs*), 3

I

`InteractiveService` (*class in lcservice.service*), 4

J

`Job` (*class in lcservice.jobs*), 3
`JsonData` (*class in lcservice.jobs*), 4

L

`lcservice` (*module*), 7
`lcservice.jobs` (*module*), 3
`lcservice.servers` (*module*), 3
`lcservice.servers.cherrypy` (*module*), 3
`lcservice.service` (*module*), 4
`lcservice.simulator` (*module*), 7
`length()` (*lcservice.jobs.Table method*), 4
`log()` (*lcservice.service.Service method*), 5
`logCritical()` (*lcservice.service.Service method*), 6

N

`narrate()` (*lcservice.jobs.Job method*), 4

O

`onDeploymentEvent()` (*lcservice.service.Service method*), 6

`onDetection()` (*lcservice.service.Service method*), 6
`onLogEvent()` (*lcservice.service.Service method*), 6
`onNewSensor()` (*lcservice.service.Service method*), 6
`onOrgInstalled()` (*lcservice.service.Service method*), 6
`onOrgUninstalled()` (*lcservice.service.Service method*), 6
`onRequest()` (*lcservice.service.Service method*), 6
`onServiceError()` (*lcservice.service.Service method*), 6
`onShutdown()` (*lcservice.service.Service method*), 6
`onStartup()` (*lcservice.service.Service method*), 6

P

`parallelExec()` (*lcservice.service.Service method*), 6
`parallelExecEx()` (*lcservice.service.Service method*), 6
`publishResource()` (*lcservice.service.Service method*), 6

R

`response()` (*lcservice.service.Service method*), 7
`responseNotImplemented()` (*lcservice.service.Service method*), 7

S

`schedule()` (*lcservice.service.Service method*), 7
`ServeCherryPy()` (*in module lcservice.servers.cherrypy*), 3
`Service` (*class in lcservice.service*), 4
`setCause()` (*lcservice.jobs.Job method*), 4
`setRequestParameters()` (*lcservice.service.Service method*), 7
`subscribeToDetect()` (*lcservice.service.Service method*), 7

T

`Table` (*class in lcservice.jobs*), 4

Y

`YamlData` (*class in lcservice.jobs*), 4