
Honeycomb Documentation

Release 0.1.1

Cymmetria

Feb 16, 2019

Contents

1	Usage	3
1.1	Running Honeycomb from command line	3
1.1.1	honeycomb	3
1.2	Running Honeycomb in a container	9
1.3	Plugin API Reference	10
1.3.1	honeycomb.servicemanager.base_service module	10
1.3.2	honeycomb.integrationmanager.integration_utils module	12
1.4	Honeycomb API Reference	13
1.4.1	honeycomb package	13
	Python Module Index	33



Honeycomb is an open-source honeypot framework created by [Cymmetria](#).

Honeycomb allows running honeypots with various integrations from a public library of plugins at https://github.com/Cymmetria/honeycomb_plugins

Writing new honeypot services and integrations for honeycomb is super easy! See the [plugins repo](#) for more info.

Full CLI documentation can be found at <http://honeycomb.cymmetria.com/en/latest/cli.html>

CHAPTER 1

Usage

Using pip:

```
$ pip install honeycomb-framework
$ honeycomb --help
```

Using Docker:

```
$ docker run -v honeycomb.yml:/usr/share/honeycomb/honeycomb.yml cymmetria/honeycomb
```

1.1 Running Honeycomb from command line

1.1.1 honeycomb

Honeycomb is a honeypot framework.

```
honeycomb [OPTIONS] COMMAND [ARGS]...
```

Options

- H, --home** <home>
Honeycomb home path [default: /home/docs/.config/honeycomb]
- iamroot**
Force run as root (NOT RECOMMENDED!)
- c, --config** <config>
Path to a honeycomb.yml file that provides instructions
- v, --verbose**
Enable verbose logging

--version

Show the version and exit.

Environment variables

DEBUG

Provide a default for *--verbose*

integration

Honeycomb integration commands.

```
honeycomb integration [OPTIONS] COMMAND [ARGS]...
```

configure

Configure an integration with default parameters.

You can still provide one-off integration arguments to *honeycomb.commands.service.run()* if required.

```
honeycomb integration configure [OPTIONS] INTEGRATION [ARGS]...
```

Options

-e, --editable

Load integration directly from unspecified path without installing (mainly for dev)

-a, --show_args

Show available integration arguments

Arguments

INTEGRATION

Required argument

ARGS

Optional argument(s)

install

Install a honeycomb integration from the online library, local path or zipfile.

```
honeycomb integration install [OPTIONS] [INTEGRATIONS]...
```

Arguments

INTEGRATIONS

Optional argument(s)

list

List integrations.

```
honeycomb integration list [OPTIONS]
```

Options

-r, --remote

Include available integrations from online repository

show

Show detailed information about a package.

```
honeycomb integration show [OPTIONS] INTEGRATION
```

Options

-r, --remote

Show information only from remote repository

Arguments

INTEGRATION

Required argument

test

Execute the integration's internal test method to verify it's working as intended.

```
honeycomb integration test [OPTIONS] [INTEGRATIONS]...
```

Options

-e, --editable

Run integration directly from specified path (main for dev)

Arguments

INTEGRATIONS

Optional argument(s)

uninstall

Uninstall a integration.

```
honeycomb integration uninstall [OPTIONS] [INTEGRATIONS]...
```

Options

-y, --yes

Don't ask for confirmation of uninstall deletions.

Arguments

INTEGRATIONS

Optional argument(s)

service

Honeycomb service commands.

```
honeycomb service [OPTIONS] COMMAND [ARGS]...
```

install

Install a honeypot service from the online library, local path or zipfile.

```
honeycomb service install [OPTIONS] [SERVICES]...
```

Arguments

SERVICES

Optional argument(s)

list

List services.

```
honeycomb service list [OPTIONS]
```

Options

-r, --remote

Include available services from online repository

logs

Show logs of daemonized service.

```
honeycomb service logs [OPTIONS] SERVICES...
```

Options

- n, --num** <num>
Number of lines to read from end of file [default: 10]
- f, --follow**
Follow log output

Arguments

SERVICES
Required argument(s)

run

Load and run a specific service.

```
honeycomb service run [OPTIONS] SERVICE [ARGS]...
```

Options

- d, --daemon**
Run service in daemon mode
- e, --editable**
Load service directly from specified path without installing (mainly for dev)
- a, --show-args**
Show available service arguments
- i, --integration** <integration>
Enable an integration

Arguments

SERVICE
Required argument

ARGS
Optional argument(s)

show

Show detailed information about a package.

```
honeycomb service show [OPTIONS] SERVICE
```

Options

-r, --remote

Show information only from remote repository

Arguments

SERVICE

Required argument

status

Show status of installed service(s).

```
honeycomb service status [OPTIONS] [SERVICES]...
```

Options

-a, --show-all

Show status for all services

Arguments

SERVICES

Optional argument(s)

stop

Stop a running service daemon.

```
honeycomb service stop [OPTIONS] SERVICE
```

Options

-e, --editable

Load service directly from specified path without installing (mainly for dev)

Arguments

SERVICE

Required argument

test

Execute the service's internal test method to verify it's working as intended.

If there's no such method, honeycomb will attempt to connect to the port listed in config.json

```
honeycomb service test [OPTIONS] [SERVICES]...
```

Options

-f, --force

Do not check if service is running before testing

-e, --editable

Run service directly from specified path (main for dev)

Arguments

SERVICES

Optional argument(s)

uninstall

Uninstall a service.

```
honeycomb service uninstall [OPTIONS] [SERVICES]...
```

Options

-y, --yes

Don't ask for confirmation of uninstall deletions.

Arguments

SERVICES

Optional argument(s)

1.2 Running Honeycomb in a container

The rationale of container support is to allow rapid configuration and deployment so that launching honeypots would be simple and easy.

Since Honeycomb is a standalone runner for services and integrations, it doesn't make sense for it to orchestrate deployment of external honeypots using docker. Instead, Honeycomb itself could be run as a container.

This means the goal is to allow simple configuration that can be passed on to Honeycomb and launch services with integrations easily.

To launch a Honeycomb service with a configured integration, the user needs to type in several commands to install a service, install an integration, configure that integration and finally run the service with optional parameters.

This actually resembles configuring a docker environment, where the user needs to type in several commands to define volumes, networks, and finally run the desired container.

A yml configuration that specifies all of the desired configurations (services, integrations, etc.) will be supplied to Honeycomb, and it will work like a state-machine to reach the desired state before finally running the service.

An example Honeycomb file can be found on *github* <<https://github.com/Cymmetria/honeycomb/blob/master/honeycomb.yml>>.

```
1 ---
2 version: 1
3
4 services:
5   simple_http:
6     parameters:
7       port: 1234
8
9 integrations:
10  syslog:
11    parameters:
12      address: "127.0.0.1"
13      port: 5514
14      protocol: tcp
```

1.3 Plugin API Reference

1.3.1 honeycomb.servicemanager.base_service module

Custom Service implementation from MazeRunner.

class `honeycomb.servicemanager.base_service.DockerService(*args, **kwargs)`
Bases: `honeycomb.servicemanager.base_service.ServerCustomService`

Provides an ability to run a Docker container that will be monitored for events.

docker_image_name
Return docker image name.

docker_params
Return a dictionary of docker run parameters.

See also:

Docker run: <https://docs.docker.com/engine/reference/run/>

Returns Dictionary, e.g., `dict(ports={80: 80})`

get_lines()
Fetch log lines from the docker service.

Returns A blocking logs generator

on_server_shutdown()
Stop the container before shutting down.

on_server_start()
Service run loop function.

Run the desired docker container with parameters and start parsing the monitored file for alerts.

parse_line (*line*)

Parse line and return dictionary if its an alert, else None / {}.

read_lines (*file_path*, *empty_lines=False*, *signal_ready=True*)

Fetch lines from file.

In case the file handler changes (logrotate), reopen the file.

Parameters

- **file_path** – Path to file
- **empty_lines** – Return empty lines
- **signal_ready** – Report signal ready on start

```
class honeycomb.servicemanager.base_service.ServerCustomService (alert_types:  
                                                                list,          ser-  
                                                                vice_args: dict  
                                                                = {})
```

Bases: multiprocessing.context.Process

Custom Service Class.

This class provides a basic wrapper for honeycomb (and mazerunner) services.

add_alert_to_queue (*alert_dict*)

Log alert and send to integrations.

alert_types = None

List of alert types, parsed from config.json

alerts_queue = None

emit (***kwargs*)

Send alerts to logfile.

Parameters *kwargs* – Fields to pass to `honeycomb.decoymanager.models.Alert`

logger = <Logger honeycomb.servicemanager.base_service (DEBUG)>

Logger to be used by plugins and collected by main logger.

on_server_shutdown ()

Shutdown function of the server.

Override this and take care to gracefully shut down your service (e.g., close files)

on_server_start ()

Service run loop function.

The service manager will call this function in a new thread.

Note: Must call `signal_ready()` after finishing configuration

run ()

Daemon entry point.

run_service ()

Run the service and start an alert processing queue.

See also:

Use `on_server_start()` and `on_server_shutdown()` for starting and shutting down your service

```
service_args = None
    Validated dictionary of service arguments (see: honeycomb.utils.plugin_utils.
    parse_plugin_args())

signal_ready()
    Signal the service manager this service is ready for incoming connections.

thread_server = None
```

1.3.2 honeycomb.integrationmanager.integration_utils module

Honeycomb Integration Manager.

class `honeycomb.integrationmanager.integration_utils.BaseIntegration` (*integration_data*)
Bases: `object`

Base Output Integration Class.

Use `__init__()` to set up any prerequisites needed before sending events, validate paramaters, etc.

Parameters `integration_data` (*dict*) – Integration parameters

Raises `IntegrationMissingRequiredFieldError` – If a required field is missing.

format_output_data (*output_data*)

Process and format the `output_data` returned by `send_event()` before display.

This is currently only relevant for MazeRunner, if you don't return an output this should return `output_data` without change.

Parameters `output_data` – As returned by `send_event()`

Return type `dict`

Returns MazeRunner compatible UI output.

Raises `IntegrationOutputFormatError` – If there's a problem formatting the output data.

poll_for_updates (*integration_output_data*)

Poll external service for updates.

If service has enabled polling, this method will be called periodically and should act like `send_event()`

Parameters `integration_output_data` – Output data returned by previous `send_event()` or `poll_for_updates()`

Returns See `send_event()`

Raises `IntegrationPollEventError` – If there's a problem polling for updates.

send_event (*alert_dict*)

Send alert event to external integration.

Parameters `alert_dict` – A dictionary with all the alert fields.

Return type `tuple(dict(output_data), object(output_file))`

Raises

- `IntegrationSendEventError` – If there's a problem sending the event.
- `IntegrationMissingRequiredFieldError` – If a required field is missing.

Returns A tuple where the first value is a dictionary with information to display in the UI, and the second is an optional file to be attached. If polling is enabled, the returned `output_data` will be passed to `poll_for_updates()`. If your integration returns nothing, you should return `({}, None)`.

test_connection (*integration_data*)

Perform a test to ensure the integration is configured correctly.

This could include testing authentication or performing a test query.

Parameters `integration_data` – Integration arguments.

Returns *success*

Return type tuple(bool(success), str(response))

1.4 Honeycomb API Reference

1.4.1 honeycomb package

Subpackages

`honeycomb.commands.service` package

Submodules

`honeycomb.commands.service.install` module

Honeycomb service install command.

`honeycomb.commands.service.list` module

Honeycomb service list command.

`honeycomb.commands.service.logs` module

Honeycomb service logs command.

`honeycomb.commands.service.run` module

Honeycomb service run command.

`honeycomb.commands.service.show` module

Honeycomb service show command.

`honeycomb.commands.service.status` module

Honeycomb service status command.

honeycomb.commands.service.stop module

Honeycomb service stop command.

honeycomb.commands.service.test module

Honeycomb service test command.

honeycomb.commands.service.uninstall module

Honeycomb service uninstall command.

honeycomb.commands.integration package

Submodules

honeycomb.commands.integration.configure module

Honeycomb integration run command.

honeycomb.commands.integration.install module

Honeycomb integration install command.

honeycomb.commands.integration.list module

Honeycomb integration list command.

honeycomb.commands.integration.show module

Honeycomb integration show command.

honeycomb.commands.integration.test module

Honeycomb integration test command.

honeycomb.commands.integration.uninstall module

Honeycomb integration uninstall command.

honeycomb.decoymanager package

Submodules

honeycomb.decoymanager.models module

Honeycomb defs and constants.

```
class honeycomb.decoymanager.models.Alert (alert_type: honey-  
comb.decoymanager.models.AlertType, id:  
str = NOTHING, status: int = 2, timestamp:  
datetime.datetime = NOTHING)
```

Bases: object

Alert object.

```
ALERT_STATUS = ((0, 'Ignore'), (1, 'Mute'), (2, 'Alert'))
```

```
STATUS_ALERT = 2
```

```
STATUS_IGNORED = 0
```

```
STATUS_MUTED = 1
```

```
additional_fields
```

```
address
```

```
alert_type
```

```
cmd
```

```
decoy_hostname
```

```
decoy_ipv4
```

```
decoy_name
```

```
decoy_os
```

```
dest_ip
```

```
dest_port
```

```
domain
```

```
end_timestamp
```

```
event_description
```

```
event_type
```

```
file_accessed
```

```
id
```

```
image_file
```

```
image_md5
```

```
image_path
```

```
image_sha256
```

```
manufacturer
```

```
originating_hostname
```

```
originating_ip
originating_mac_address
originating_port
password
pid
ppid
request
status
timestamp
transport_protocol
uid
username
```

```
class honeycomb.decoymanager.models.AlertType(name: str, label: str, service_type: honey-
                                             comb.servicemanager.models.ServiceType)
    Bases: object
    Alert Type.
    label
    name
    service_type
```

Module contents

Honeycomb Decoy Manager.

honeycomb.integrationmanager package

Submodules

honeycomb.integrationmanager.defs module

Honeycomb integrations definitions and constants.

```
class honeycomb.integrationmanager.defs.IntegrationAlertStatuses
    Bases: honeycomb.defs.IBaseType
    Provides information about the alert status in queue.
    DONE = BaseNameLabel(name='done', label='Done')
    ERROR_MISSING_SEND_FIELDS = BaseNameLabel(name='error_missing', label='Error. Missing')
    ERROR_POLLING = BaseNameLabel(name='error_polling', label='Error polling')
    ERROR_POLLING_FORMATTING = BaseNameLabel(name='error_polling_formatting', label='Error')
    ERROR_SENDING = BaseNameLabel(name='error_sending', label='Error sending')
```

```

ERROR_SENDING_FORMATTING = BaseNameLabel(name='error_sending_formatting', label='Error
IN_POLLING = BaseNameLabel(name='in_polling', label='Polling')
PENDING = BaseNameLabel(name='pending', label='Pending')
POLLING = BaseNameLabel(name='polling', label='Polling')
class honeycomb.integrationmanager.defs.IntegrationTypes
    Bases: honeycomb.defs.IBaseType
    Integration types.
    Currently only output event is supported.
EVENT_OUTPUT = BaseNameLabel(name='event_output', label='Event output')

```

honeycomb.integrationmanager.error_messages module

Honeycomb integration error messages.

honeycomb.integrationmanager.exceptions module

Honeycomb Output Integration Exceptions.

```

exception honeycomb.integrationmanager.exceptions.IntegrationMissingRequiredFieldError(*args,
                                                                                         **kwargs)
    Bases: honeycomb.exceptions.PluginError
    IntegrationMissingRequiredFieldError.
    Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.
exception honeycomb.integrationmanager.exceptions.IntegrationNoMethodImplementationError(*args,
                                                                                          **)
    Bases: honeycomb.exceptions.PluginError
    IntegrationNoMethodImplementationError.
    Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.
exception honeycomb.integrationmanager.exceptions.IntegrationNotFound(*args,
                                                                        **kwargs)
    Bases: honeycomb.exceptions.PluginError
    Integration not found.
    Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.
msg_format = 'Cannot find integration named {}, try installing it?'
exception honeycomb.integrationmanager.exceptions.IntegrationOutputFormatError(*args,
                                                                                   **kwargs)
    Bases: honeycomb.exceptions.PluginError
    IntegrationOutputFormatError.
    Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.
exception honeycomb.integrationmanager.exceptions.IntegrationPackageError(*args,
                                                                             **kwargs)
    Bases: honeycomb.exceptions.PluginError
    IntegrationPackageError.

```

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

```
exception honeycomb.integrationmanager.exceptions.IntegrationPollEventError(*args,  
                                                                    **kwargs)
```

Bases: *honeycomb.exceptions.PluginError*

IntegrationPollEventError.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

```
exception honeycomb.integrationmanager.exceptions.IntegrationSendEventError(*args,  
                                                                    **kwargs)
```

Bases: *honeycomb.exceptions.PluginError*

IntegrationSendEventError.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

```
msg_format = 'Error sending integration event:  {}'
```

```
exception honeycomb.integrationmanager.exceptions.IntegrationTestFailed(*args,  
                                                                    **kwargs)
```

Bases: *honeycomb.exceptions.PluginError*

Integration not found.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

```
msg_format = 'Integration test failed, details:  {}'
```

honeycomb.integrationmanager.integration_utils module

Honeycomb Integration Manager.

```
class honeycomb.integrationmanager.integration_utils.BaseIntegration(integration_data)  
    Bases: object
```

Base Output Integration Class.

Use `__init__()` to set up any prerequisites needed before sending events, validate paramaters, etc.

Parameters `integration_data` (*dict*) – Integration parameters

Raises *IntegrationMissingRequiredFieldError* – If a required field is missing.

```
format_output_data (output_data)
```

Process and format the output_data returned by `send_event()` before display.

This is currently only relevant for MazeRunner, if you don't return an output this should return output_data without change.

Parameters `output_data` – As returned by `send_event()`

Return type dict

Returns MazeRunner compatible UI output.

Raises *IntegrationOutputFormatError* – If there's a problem formatting the output data.

```
poll_for_updates (integration_output_data)
```

Poll external service for updates.

If service has enabled polling, this method will be called periodically and should act like `send_event()`

Parameters `integration_output_data` – Output data returned by previous `send_event()` or `poll_for_updates()`

Returns See `send_event()`

Raises `IntegrationPollEventError` – If there's a problem polling for updates.

send_event (`alert_dict`)

Send alert event to external integration.

Parameters `alert_dict` – A dictionary with all the alert fields.

Return type tuple(dict(output_data), object(output_file))

Raises

- `IntegrationSendEventError` – If there's a problem sending the event.
- `IntegrationMissingRequiredFieldError` – If a required field is missing.

Returns A tuple where the first value is a dictionary with information to display in the UI, and the second is an optional file to be attached. If polling is enabled, the returned `output_data` will be passed to `poll_for_updates()`. If your integration returns nothing, you should return `({}, None)`.

test_connection (`integration_data`)

Perform a test to ensure the integration is configured correctly.

This could include testing authentication or performing a test query.

Parameters `integration_data` – Integration arguments.

Returns `success`

Return type tuple(bool(success), str(response))

honeycomb.integrationmanager.models module

Honeycomb integration models.

```
class honeycomb.integrationmanager.models.ConfiguredIntegration (name: str,
                                                                path: str,
                                                                integration: honey-
                                                                comb.integrationmanager.models.Integr-
                                                                ation,
                                                                send_muted:
                                                                bool =
                                                                False, created_at: date-
                                                                time.datetime
                                                                = NOTHING)
```

Bases: `object`

Configured integration model.

```
class honeycomb.integrationmanager.models.Integration (parameters:      str,      dis-
                                                    play_name:      str,      re-
                                                    quired_fields:      list,
                                                    polling_enabled:      bool,
                                                    integration_type:      str,
                                                    max_send_retries:      int,
                                                    supported_event_types: list,
                                                    test_connection_enabled:
bool,      module=None, de-
description: str = None,
polling_duration:      date-
time.timedelta = 0)
```

Bases: object

Integration model.

```
class honeycomb.integrationmanager.models.IntegrationAlert (alert:      honey-
comb.decoymanager.models.Alert,
status:      str,      re-
tries:      int,      config-
ured_integration:
honey-
comb.integrationmanager.models.ConfiguredIn
```

Bases: object

Integration alert model.

honeycomb.integrationmanager.registration module

Honeycomb service manager.

honeycomb.integrationmanager.registration.**get_integration_module** (integration_path)
Add custom paths to sys and import integration module.

Parameters **integration_path** – Path to integration folder

honeycomb.integrationmanager.registration.**register_integration** (package_folder)
Register a honeycomb integration.

Parameters **package_folder** – Path to folder with integration to load

Returns Validated integration object

Return type honeycomb.utils.defs.Integration()

honeycomb.integrationmanager.tasks module

Honeycomb integration tasks.

honeycomb.integrationmanager.tasks.**configure_integration** (path)
Configure and enable an integration.

honeycomb.integrationmanager.tasks.**create_integration_alert_and_call_send** (alert, con-
fig-
ured_integration)
Create an IntegrationAlert object and send it to Integration.

`honeycomb.integrationmanager.tasks.get_current_datetime_utc()`

Return a datetime object localized to UTC.

`honeycomb.integrationmanager.tasks.get_valid_configured_integrations(alert)`

Return a list of integrations for alert filtered by alert_type.

Returns A list of relevant integrations

`honeycomb.integrationmanager.tasks.poll_integration_alert_data(integration_alert)`

Poll for updates on waiting IntegrationAlerts.

`honeycomb.integrationmanager.tasks.poll_integration_information_for_waiting_integration_alerts()`

poll_integration_information_for_waiting_integration_alerts.

`honeycomb.integrationmanager.tasks.send_alert_to_configured_integration(integration_alert)`

Send IntegrationAlert to configured integration.

`honeycomb.integrationmanager.tasks.send_alert_to_subscribed_integrations(alert)`

Send Alert to relevant integrations.

Module contents

Honeycomb Output Manager.

honeycomb.servicemanager package

Submodules

honeycomb.servicemanager.base_service module

Custom Service implementation from MazeRunner.

class `honeycomb.servicemanager.base_service.DockerService(*args, **kwargs)`

Bases: `honeycomb.servicemanager.base_service.ServerCustomService`

Provides an ability to run a Docker container that will be monitored for events.

docker_image_name

Return docker image name.

docker_params

Return a dictionary of docker run parameters.

See also:

Docker run: <https://docs.docker.com/engine/reference/run/>

Returns Dictionary, e.g., dict (ports={80: 80})

get_lines()

Fetch log lines from the docker service.

Returns A blocking logs generator

on_server_shutdown()

Stop the container before shutting down.

on_server_start ()

Service run loop function.

Run the desired docker container with parameters and start parsing the monitored file for alerts.

parse_line (line)

Parse line and return dictionary if its an alert, else None / {}.

read_lines (file_path, empty_lines=False, signal_ready=True)

Fetch lines from file.

In case the file handler changes (logrotate), reopen the file.

Parameters

- **file_path** – Path to file
- **empty_lines** – Return empty lines
- **signal_ready** – Report signal ready on start

```
class honeycomb.servicemanager.base_service.ServerCustomService (alert_types:  
                                                                list,          ser-  
                                                                vice_args: dict  
                                                                = {})
```

Bases: multiprocessing.context.Process

Custom Service Class.

This class provides a basic wrapper for honeycomb (and mazerunner) services.

add_alert_to_queue (alert_dict)

Log alert and send to integrations.

alert_types = None

List of alert types, parsed from config.json

alerts_queue = None

emit (kwargs)**

Send alerts to logfile.

Parameters **kwargs** – Fields to pass to `honeycomb.decoymanager.models.Alert`

logger = <Logger honeycomb.servicemanager.base_service (DEBUG)>

Logger to be used by plugins and collected by main logger.

on_server_shutdown ()

Shutdown function of the server.

Override this and take care to gracefully shut down your service (e.g., close files)

on_server_start ()

Service run loop function.

The service manager will call this function in a new thread.

Note: Must call `signal_ready()` after finishing configuration

run ()

Daemon entry point.

run_service()

Run the service and start an alert processing queue.

See also:

Use `on_server_start()` and `on_server_shutdown()` for starting and shutting down your service

service_args = None

Validated dictionary of service arguments (see: `honeycomb.utils.plugin_utils.parse_plugin_args()`)

signal_ready()

Signal the service manager this service is ready for incoming connections.

thread_server = None

honeycomb.servicemanager.defs module

Honeycomb services definitions and constants.

`honeycomb.servicemanager.defs.ALLOWED_PROTOCOLS = ['TCP', 'UDP']`

Parameters.

`honeycomb.servicemanager.defs.STDERRLOG = 'stderr.log'`

Service section.

honeycomb.servicemanager.error_messages module

Honeycomb services error messages.

honeycomb.servicemanager.exceptions module

Honeycomb Service Manager Exceptions.

exception `honeycomb.servicemanager.exceptions.ServiceManagerException(*args, **kwargs)`

Bases: `honeycomb.exceptions.PluginError`

Generic Service Manager Exception.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

exception `honeycomb.servicemanager.exceptions.ServiceNotFound(*args, **kwargs)`

Bases: `honeycomb.servicemanager.exceptions.ServiceManagerException`

Specified service does not exist.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

msg_format = 'Cannot find service named {}, try installing it?'

exception `honeycomb.servicemanager.exceptions.UnsupportedOS(*args, **kwargs)`

Bases: `honeycomb.servicemanager.exceptions.ServiceManagerException`

Specified service does not exist.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

msg_format = 'Service requires running on {} and you are using {}'

honeycomb.servicemanager.models module

Honeycomb service models.

```
class honeycomb.servicemanager.models.OSFamilies
    Bases: honeycomb.defs.IBaseType
    Defines supported platforms for services.

    ALL = BaseNameLabel(name='All', label='All')
    LINUX = BaseNameLabel(name='Linux', label='Linux')
    MACOS = BaseNameLabel(name='Darwin', label='Darwin')
    WINDOWS = BaseNameLabel(name='Windows', label='Windows')

class honeycomb.servicemanager.models.ServiceType(name: str, ports: list, label: str, allow_many: bool, supported_os_families: list, alert_types: list = [])
    Bases: object
    Holds loaded service metadata.
```

honeycomb.servicemanager.registration module

Honeycomb service manager.

```
honeycomb.servicemanager.registration.get_service_module(service_path)
    Add custom paths to sys and import service module.

    Parameters service_path – Path to service folder

honeycomb.servicemanager.registration.register_service(package_folder)
    Register a honeycomb service.

    Parameters package_folder – Path to folder with service to load

    Returns Validated service object

    Return type honeycomb.utils.defs.ServiceType()
```

Module contents

Honeycomb Service Manager.

honeycomb.utils package

Submodules

honeycomb.utils.config_utils module

Honeycomb Config Utilities.

`honeycomb.utils.config_utils.config_field_type(field, cls)`

Validate a config field against a type.

Similar functionality to `validate_field_matches_type()` but returns `honeycomb.defs.ConfigField`

`honeycomb.utils.config_utils.get_config_parameters(plugin_path)`

Return the parameters section from config.json.

`honeycomb.utils.config_utils.get_truetype(value)`

Convert a string to a pythonized parameter.

`honeycomb.utils.config_utils.is_valid_field_name(value)`

Ensure field name is valid.

`honeycomb.utils.config_utils.process_config(ctx, configfile)`

Process a yaml config with instructions.

This is a heavy method that loads lots of content, so we only run the imports if its called.

`honeycomb.utils.config_utils.validate_config(config_json, fields)`

Validate a JSON file configuration against list of `honeycomb.defs.ConfigField`.

`honeycomb.utils.config_utils.validate_config_parameters(config_json, allowed_keys, allowed_types)`

Validate parameters in config file.

`honeycomb.utils.config_utils.validate_field(field, allowed_keys, allowed_types)`

Validate field is allowed and valid.

`honeycomb.utils.config_utils.validate_field_matches_type(field, value, field_type, select_items=None, _min=None, _max=None)`

Validate a config field against a specific type.

honeycomb.utils.daemon module

Honeycomb DaemonRunner utility.

```
class honeycomb.utils.daemon.myRunner(app, pidfile=None, stdout=<_io.TextIOWrapper
    name='<stdout>' mode='w' encoding='UTF-8'>,
    stderr=<_io.TextIOWrapper
    name='<stderr>' mode='w' encoding='UTF-8'>, stdin=<_io.TextIOWrapper
    name='/dev/null' mode='rt' encoding='UTF-8'>)
```

Bases: `daemon.runner.DaemonRunner`

Overriding default runner behaviour to be simpler.

Override init to fit honeycomb needs.

We initialize app with default stdout/stderr from sys instead of file path and remove the use of `parse_args()` since it's not actually a standalone runner

honeycomb.utils.plugin_utils module

Honeycomb generic plugin install utils.

exception `honeycomb.utils.plugin_utils.CTError(errors)`

Bases: `Exception`

Copytree exception class, used to collect errors from the recursive `copy_tree` function.

Collect errors.

Parameters `errors` – Collected errors

`honeycomb.utils.plugin_utils.copy_file(src, dst)`

Copy a single file.

Parameters

- **src** – Source name
- **dst** – Destination name

`honeycomb.utils.plugin_utils.copy_tree(src, dst, symlinks=False, ignore=[])`

Copy a full directory structure.

Parameters

- **src** – Source path
- **dst** – Destination path
- **symlinks** – Copy symlinks
- **ignore** – Subdirs/filenames to ignore

`honeycomb.utils.plugin_utils.get_plugin_path(home, plugin_type, plugin_name, editable=False)`

Return path to plugin.

Parameters

- **home** – Path to honeycomb home
- **plugin_type** – Type of plugin (`honeycomb.defs.SERVICES` or `honeycomb.defs.INTEGRATIONS`)
- **plugin_name** – Name of plugin
- **editable** – Use `plugin_name` as direct path instead of loading from honeycomb home folder

`honeycomb.utils.plugin_utils.get_select_items(items)`

Return list of possible select items.

`honeycomb.utils.plugin_utils.install_deps(pkgpath)`

Install plugin dependencies using pip.

We import pip here to reduce load time for when its not needed.

`honeycomb.utils.plugin_utils.install_dir(pkgpath, install_path, register_func, delete_after_install=False)`

Install plugin from specified directory.

`install_path` and `register_func` are same as `install_plugin()`. :param `delete_after_install`: Delete `pkgpath` after install (used in `install_from_zip()`).

`honeycomb.utils.plugin_utils.install_from_repo(pkgname, plugin_type, install_path, register_func)`

Install plugin from online repo.

```
honeycomb.utils.plugin_utils.install_from_zip(pkgpath, install_path, register_func,
                                              delete_after_install=False)
```

Install plugin from zipfile.

```
honeycomb.utils.plugin_utils.install_plugin(pkgpath, plugin_type, install_path, register_func)
```

Install specified plugin.

Parameters

- **pkgpath** – Name of plugin to be downloaded from online repo or path to plugin folder or zip file.
- **install_path** – Path where plugin will be installed.
- **register_func** – Method used to register and validate plugin.

```
honeycomb.utils.plugin_utils.list_local_plugins(plugin_type, plugins_path, plugin_details)
```

List local plugins with details.

```
honeycomb.utils.plugin_utils.list_remote_plugins(installed_plugins, plugin_type)
```

List remote plugins from online repo.

```
honeycomb.utils.plugin_utils.parse_plugin_args(command_args, config_args)
```

Parse command line arguments based on the plugin's parameters config.

Parameters

- **command_args** – Command line arguments as provided by the user in *key=value* format.
- **config_args** – Plugin parameters parsed from config.json.

Returns Validated dictionary of parameters that will be passed to plugin class

```
honeycomb.utils.plugin_utils.print_plugin_args(plugin_path)
```

Print plugin parameters table.

```
honeycomb.utils.plugin_utils.uninstall_plugin(pkgpath, force)
```

Uninstall a plugin.

Parameters

- **pkgpath** – Path to package to uninstall (delete)
- **force** – Force uninstall without asking

honeycomb.utils.tailer module

Honeycomb service log tailer.

```
class honeycomb.utils.tailer.Tailer(name: str, filepath: str, color: str = "", nlines: int =
    10, follow: bool = False, outfile=<_io.TextIOWrapper
    name='<stdout>' mode='w' encoding='UTF-8'>,
    sleeptime: int = 0.5, show_name: bool = True,
    used_colors: list = [])
```

Bases: object

Colorized file tailer.

Print lines from a file prefixed with a colored name. Optionally continue to follow file.

```
follow_file()
```

Follow a file and send every new line to a callback.

print_log (*line*)

Print a line from a logfile.

print_named_log (*line*)

Print a line from a logfile prefixed with service name.

stop ()

Stop follow.

honeycomb.utils.wait module

Honeycomb wait utilities.

exception honeycomb.utils.wait.TimeoutException

Bases: Exception

Exception to be raised on timeout.

honeycomb.utils.wait.search_json_log (*filepath, key, value*)

Search json log file for a key=value pair.

Parameters

- **filepath** – Valid path to a json file
- **key** – key to match
- **value** – value to match

Returns First matching line in json log file, parsed by `json.loads()`

honeycomb.utils.wait.wait_until (*func, check_return_value=True, total_timeout=60, interval=0.5, exc_list=None, error_message="", *args, **kwargs*)

Run a command in a loop until desired result or timeout occurs.

Parameters

- **func** – Function to call and wait for
- **check_return_value** (*bool*) – Examine return value
- **total_timeout** (*int*) – Wait timeout,
- **interval** (*float*) – Sleep interval between retries
- **exc_list** (*list*) – Acceptable exception list
- **error_message** (*str*) – Default error messages
- **args** – args to pass to func
- **kwargs** – kwargs to pass to fun

Module contents

Honeycomb Utils.

Submodules

honeycomb.cli module

Honeycomb Command Line Interface.

class honeycomb.cli.**MyLogger** (*name, level=0*)

Bases: logging.Logger

Custom Logger.

Initialize the logger with a name and an optional level.

makeRecord (*name, level, fn, lno, msg, args, exc_info, func=None, extra=None, sinfo=None*)

Override default logger to allow overriding of internal attributes.

honeycomb.cli.**setup_logging** (*home, verbose*)

Configure logging for honeycomb.

honeycomb.defs module

Honeycomb defs and constants.

class honeycomb.defs.**BaseCollection**

Bases: object

Abstract type collection mixin, should hold BaseNameLabel attributes.

class honeycomb.defs.**BaseNameLabel** (*name, label*)

Bases: object

Generic name/label class.

honeycomb.defs.**CONFIG_FILE_NAME** = 'config.json'

Parameters constants.

class honeycomb.defs.**ConfigField** (*validator_func, get_error_message*)

Bases: object

Config Validator.

error_message is also a function to calculate the error when we ran the validator_func

honeycomb.defs.**GITHUB_RAW_URL** = 'https://raw.githubusercontent.com/Cymmetria/honeycomb_plugins'

Config constants.

class honeycomb.defs.**IBaseType**

Bases: object

Abstract type interface, provides BaseNameLabel collection methods.

classmethod **all_labels** ()

Return list of all property labels.

classmethod **all_names** ()

Return list of all property names.

honeycomb.error_messages module

Honeycomb generic error messages.

honeycomb.exceptions module

Honeycomb Exceptions.

exception `honeycomb.exceptions.BaseHoneycombException(*args, **kwargs)`

Bases: `click.exceptions.ClickException`

Base Exception.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

msg_format = `None`

exception `honeycomb.exceptions.ConfigFieldMissing(*args, **kwargs)`

Bases: `honeycomb.exceptions.ConfigValidationError`

Field is missing from config file.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

msg_format = `'field {} is missing from config file'`

exception `honeycomb.exceptions.ConfigFieldTypeMismatch(*args, **kwargs)`

Bases: `honeycomb.exceptions.ConfigValidationError`

Config field does not match specified type.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

msg_format = `'Parameters: Bad value for {}={} (must be {})'`

exception `honeycomb.exceptions.ConfigFieldValidationError(*args, **kwargs)`

Bases: `honeycomb.exceptions.ConfigValidationError`

Error validating config field.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

msg_format = `'Failed to import config. error in field {} with value {}: {}'`

exception `honeycomb.exceptions.ConfigFileNotFound(*args, **kwargs)`

Bases: `honeycomb.exceptions.PluginError`

Config file not found.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

msg_format = `'Missing file {}'`

exception `honeycomb.exceptions.ConfigValidationError(*args, **kwargs)`

Bases: `honeycomb.exceptions.BaseHoneycombException`

Base config validation error.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

exception `honeycomb.exceptions.ParametersFieldError(*args, **kwargs)`

Bases: `honeycomb.exceptions.ConfigValidationError`

Error validating parameter.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

msg_format = `"Parameters: '{}' is not a valid {}"`

exception `honeycomb.exceptions.PathNotFound(*args, **kwargs)`

Bases: `honeycomb.exceptions.BaseHoneycombException`

Specified path was not found.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

msg_format = `'Cannot find path {}'`

exception `honeycomb.exceptions.PluginAlreadyInstalled(*args, **kwargs)`

Bases: `honeycomb.exceptions.PluginError`

Plugin already installed.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

msg_format = `'{} is already installed'`

exception `honeycomb.exceptions.PluginError(*args, **kwargs)`

Bases: `honeycomb.exceptions.BaseHoneycombException`

Base Plugin Exception.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

exception `honeycomb.exceptions.PluginNotFoundInOnlineRepo(*args, **kwargs)`

Bases: `honeycomb.exceptions.PluginError`

Plugin not found in online repo.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

msg_format = `'Cannot find {} in online repository'`

exception `honeycomb.exceptions.PluginRepoConnectionError(*args, **kwargs)`

Bases: `honeycomb.exceptions.PluginError`

Connection error when trying to connect to plugin repo.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

msg_format = `'Unable to access online repository (check debug logs for detailed info)'`

exception `honeycomb.exceptions.RequiredFieldMissing(*args, **kwargs)`

Bases: `honeycomb.exceptions.PluginError`

Required parameter is missing.

Raise ClickException and log msg with relevant debugging info from the frame that raised the exception.

msg_format = `"Parameters: '{}' is missing (use --show_args to see all parameters)"`

h

`honeycomb.cli`, 29

`honeycomb.commands.integration.configure`, 14

`honeycomb.commands.integration.install`, 14

`honeycomb.commands.integration.list`, 14

`honeycomb.commands.integration.show`, 14

`honeycomb.commands.integration.test`, 14

`honeycomb.commands.integration.uninstall`, 14

`honeycomb.commands.service.install`, 13

`honeycomb.commands.service.list`, 13

`honeycomb.commands.service.logs`, 13

`honeycomb.commands.service.run`, 13

`honeycomb.commands.service.show`, 13

`honeycomb.commands.service.status`, 13

`honeycomb.commands.service.stop`, 14

`honeycomb.commands.service.test`, 14

`honeycomb.commands.service.uninstall`, 14

`honeycomb.decoymanager`, 16

`honeycomb.decoymanager.models`, 15

`honeycomb.defs`, 29

`honeycomb.error_messages`, 29

`honeycomb.exceptions`, 30

`honeycomb.integrationmanager`, 21

`honeycomb.integrationmanager.defs`, 16

`honeycomb.integrationmanager.error_messages`, 17

`honeycomb.integrationmanager.exceptions`, 17

`honeycomb.integrationmanager.integration_utils`, 18

`honeycomb.integrationmanager.models`, 19

`honeycomb.integrationmanager.registration`, 20

`honeycomb.integrationmanager.tasks`, 20

`honeycomb.servicemanager`, 24

`honeycomb.servicemanager.base_service`, 21

`honeycomb.servicemanager.defs`, 23

`honeycomb.servicemanager.error_messages`, 23

`honeycomb.servicemanager.exceptions`, 23

`honeycomb.servicemanager.models`, 24

`honeycomb.servicemanager.registration`, 24

`honeycomb.utils`, 28

`honeycomb.utils.config_utils`, 24

`honeycomb.utils.daemon`, 25

`honeycomb.utils.plugin_utils`, 25

`honeycomb.utils.tailer`, 27

`honeycomb.utils.wait`, 28

Symbols

`-iamroot`
 honeycomb command line option, 3
`-version`
 honeycomb command line option, 3
`-H, -home <home>`
 honeycomb command line option, 3
`-a, -show-all`
 honeycomb-service-status command
 line option, 8
`-a, -show-args`
 honeycomb-service-run command line
 option, 7
`-a, -show_args`
 honeycomb-integration-configure
 command line option, 4
`-c, -config <config>`
 honeycomb command line option, 3
`-d, -daemon`
 honeycomb-service-run command line
 option, 7
`-e, -editable`
 honeycomb-integration-configure
 command line option, 4
 honeycomb-integration-test command
 line option, 5
 honeycomb-service-run command line
 option, 7
 honeycomb-service-stop command
 line option, 8
 honeycomb-service-test command
 line option, 9
`-f, -follow`
 honeycomb-service-logs command
 line option, 7
`-f, -force`
 honeycomb-service-test command
 line option, 9
`-i, -integration <integration>`

 honeycomb-service-run command line
 option, 7
`-n, -num <num>`
 honeycomb-service-logs command
 line option, 7
`-r, -remote`
 honeycomb-integration-list command
 line option, 5
 honeycomb-integration-show command
 line option, 5
 honeycomb-service-list command
 line option, 6
 honeycomb-service-show command
 line option, 8
`-v, -verbose`
 honeycomb command line option, 3
`-y, -yes`
 honeycomb-integration-uninstall
 command line option, 6
 honeycomb-service-uninstall
 command line option, 9

A

`add_alert_to_queue()` (honey-
 comb.servicemanager.base_service.ServerCustomService
 method), 22
`additional_fields` (honey-
 comb.decoymanager.models.Alert attribute),
 15
`address` (honeycomb.decoymanager.models.Alert at-
 tribute), 15
`Alert` (class in honeycomb.decoymanager.models), 15
`ALERT_STATUS` (honey-
 comb.decoymanager.models.Alert attribute),
 15
`alert_type` (honeycomb.decoymanager.models.Alert
 attribute), 15
`alert_types` (honey-
 comb.servicemanager.base_service.ServerCustomService
 attribute), 22

alerts_queue (honeycomb.servicemanager.base_service.ServerCustomService attribute), 22
 AlertType (class in honeycomb.decoymanager.models), 16
 ALL (honeycomb.servicemanager.models.OSFamilies attribute), 24
 all_labels() (honeycomb.defs.IBaseType class method), 29
 all_names() (honeycomb.defs.IBaseType class method), 29
 ALLOWED_PROTOCOLS (in module honeycomb.servicemanager.defs), 23
 ARGS
 honeycomb-integration-configure command line option, 4
 honeycomb-service-run command line option, 7

B

BaseCollection (class in honeycomb.defs), 29
 BaseHoneycombException, 30
 BaseIntegration (class in honeycomb.integrationmanager.integration_utils), 18
 BaseNameLabel (class in honeycomb.defs), 29

C

cmd (honeycomb.decoymanager.models.Alert attribute), 15
 config_field_type() (in module honeycomb.utils.config_utils), 24
 CONFIG_FILE_NAME (in module honeycomb.defs), 29
 ConfigField (class in honeycomb.defs), 29
 ConfigFieldMissing, 30
 ConfigFieldTypeMismatch, 30
 ConfigFieldValidationError, 30
 ConfigFileNotFound, 30
 configure_integration() (in module honeycomb.integrationmanager.tasks), 20
 ConfiguredIntegration (class in honeycomb.integrationmanager.models), 19
 ConfigValidationError, 30
 copy_file() (in module honeycomb.utils.plugin_utils), 26
 copy_tree() (in module honeycomb.utils.plugin_utils), 26
 create_integration_alert_and_call_send() (in module honeycomb.integrationmanager.tasks), 20
 CTErrors, 25

D

decoy_hostname (honeycomb.decoymanager.models.Alert attribute), 15
 decoy_ip_v4 (honeycomb.decoymanager.models.Alert attribute), 15
 decoy_name (honeycomb.decoymanager.models.Alert attribute), 15
 decoy_os (honeycomb.decoymanager.models.Alert attribute), 15
 dest_ip (honeycomb.decoymanager.models.Alert attribute), 15
 dest_port (honeycomb.decoymanager.models.Alert attribute), 15
 docker_image_name (honeycomb.servicemanager.base_service.DockerService attribute), 21
 docker_params (honeycomb.servicemanager.base_service.DockerService attribute), 21
 DockerService (class in honeycomb.servicemanager.base_service), 21
 domain (honeycomb.decoymanager.models.Alert attribute), 15
 DONE (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 16

E

emit() (honeycomb.servicemanager.base_service.ServerCustomService method), 22
 end_timestamp (honeycomb.decoymanager.models.Alert attribute), 15
 ERROR_MISSING_SEND_FIELDS (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 16
 ERROR_POLLING (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 16
 ERROR_POLLING_FORMATTING (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 16
 ERROR_SENDING (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 16
 ERROR_SENDING_FORMATTING (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 16
 event_description (honeycomb.decoymanager.models.Alert attribute), 15
 EVENT_OUTPUT (honeycomb.integrationmanager.defs.IntegrationTypes attribute), 17
 event_type (honeycomb.decoymanager.models.Alert attribute), 15

F

`file_accessed` (*honeycomb.decoymanager.models.Alert* attribute), 15

`follow_file()` (*honeycomb.utils.tailer.Tailer* method), 27

`format_output_data()` (*honeycomb.integrationmanager.integration_utils.BaseIntegration* method), 18

G

`get_config_parameters()` (in module *honeycomb.utils.config_utils*), 25

`get_current_datetime_utc()` (in module *honeycomb.integrationmanager.tasks*), 20

`get_integration_module()` (in module *honeycomb.integrationmanager.registration*), 20

`get_lines()` (*honeycomb.servicemanager.base_service.DockerService* method), 21

`get_plugin_path()` (in module *honeycomb.utils.plugin_utils*), 26

`get_select_items()` (in module *honeycomb.utils.plugin_utils*), 26

`get_service_module()` (in module *honeycomb.servicemanager.registration*), 24

`get_truetype()` (in module *honeycomb.utils.config_utils*), 25

`get_valid_configured_integrations()` (in module *honeycomb.integrationmanager.tasks*), 21

`GITHUB_RAW_URL` (in module *honeycomb.defs*), 29

H

`honeycomb` command line option

- `-iamroot`, 3
- `-version`, 3
- `-H`, `-home <home>`, 3
- `-c`, `-config <config>`, 3
- `-v`, `-verbose`, 3

`honeycomb-integration-configure` command line option

- `-a`, `-show_args`, 4
- `-e`, `-editable`, 4
- ARGS, 4
- INTEGRATION, 4

`honeycomb-integration-install` command line option

- INTEGRATIONS, 4

`honeycomb-integration-list` command line option

- `-r`, `-remote`, 5

`honeycomb-integration-show` command line option

- `-r`, `-remote`, 5
- INTEGRATION, 5

`honeycomb-integration-test` command line option

- `-e`, `-editable`, 5
- INTEGRATIONS, 5

`honeycomb-integration-uninstall` command line option

- `-y`, `-yes`, 6
- INTEGRATIONS, 6

`honeycomb-service-install` command line option

- SERVICES, 6

`honeycomb-service-list` command line option

- `-r`, `-remote`, 6

`honeycomb-service-logs` command line option

- `-f`, `-follow`, 7
- `-n`, `-num <num>`, 7
- SERVICES, 7

`honeycomb-service-run` command line option

- `-a`, `-show_args`, 7
- `-d`, `-daemon`, 7
- `-e`, `-editable`, 7
- `-i`, `-integration <integration>`, 7
- ARGS, 7
- SERVICE, 7

`honeycomb-service-show` command line option

- `-r`, `-remote`, 8
- SERVICE, 8

`honeycomb-service-status` command line option

- `-a`, `-show-all`, 8
- SERVICES, 8

`honeycomb-service-stop` command line option

- `-e`, `-editable`, 8
- SERVICE, 8

`honeycomb-service-test` command line option

- `-e`, `-editable`, 9
- `-f`, `-force`, 9
- SERVICES, 9

`honeycomb-service-uninstall` command line option

- `-y`, `-yes`, 9
- SERVICES, 9

`honeycomb.cli` (module), 29

`honeycomb.commands.integration.configure` (module), 14

honeycomb.commands.integration.install (<i>module</i>), 14	honeycomb.servicemanager.exceptions (<i>module</i>), 23
honeycomb.commands.integration.list (<i>module</i>), 14	honeycomb.servicemanager.models (<i>module</i>), 24
honeycomb.commands.integration.show (<i>module</i>), 14	honeycomb.servicemanager.registration (<i>module</i>), 24
honeycomb.commands.integration.test (<i>module</i>), 14	honeycomb.utils (<i>module</i>), 28
honeycomb.commands.integration.uninstall (<i>module</i>), 14	honeycomb.utils.config_utils (<i>module</i>), 24
honeycomb.commands.service.install (<i>mod- ule</i>), 13	honeycomb.utils.daemon (<i>module</i>), 25
honeycomb.commands.service.list (<i>module</i>), 13	honeycomb.utils.plugin_utils (<i>module</i>), 25
honeycomb.commands.service.logs (<i>module</i>), 13	honeycomb.utils.tailer (<i>module</i>), 27
honeycomb.commands.service.run (<i>module</i>), 13	honeycomb.utils.wait (<i>module</i>), 28
honeycomb.commands.service.show (<i>module</i>), 13	
honeycomb.commands.service.status (<i>mod- ule</i>), 13	IBaseType (<i>class in honeycomb.defs</i>), 29
honeycomb.commands.service.stop (<i>module</i>), 14	id (<i>honeycomb.decoymanager.models.Alert attribute</i>), 15
honeycomb.commands.service.test (<i>module</i>), 14	image_file (<i>honeycomb.decoymanager.models.Alert attribute</i>), 15
honeycomb.commands.service.uninstall (<i>module</i>), 14	image_md5 (<i>honeycomb.decoymanager.models.Alert attribute</i>), 15
honeycomb.decoymanager (<i>module</i>), 16	image_path (<i>honeycomb.decoymanager.models.Alert attribute</i>), 15
honeycomb.decoymanager.models (<i>module</i>), 15	image_sha256 (<i>honey- comb.decoymanager.models.Alert attribute</i>), 15
honeycomb.defs (<i>module</i>), 29	IN_POLLING (<i>honeycomb.integrationmanager.defs.IntegrationAlertStatus attribute</i>), 17
honeycomb.error_messages (<i>module</i>), 29	install_deps() (<i>in module honey- comb.utils.plugin_utils</i>), 26
honeycomb.exceptions (<i>module</i>), 30	install_dir() (<i>in module honey- comb.utils.plugin_utils</i>), 26
honeycomb.integrationmanager (<i>module</i>), 21	install_from_repo() (<i>in module honey- comb.utils.plugin_utils</i>), 26
honeycomb.integrationmanager.defs (<i>mod- ule</i>), 16	install_from_zip() (<i>in module honey- comb.utils.plugin_utils</i>), 26
honeycomb.integrationmanager.error_messages (<i>module</i>), 17	install_plugin() (<i>in module honey- comb.utils.plugin_utils</i>), 27
honeycomb.integrationmanager.exceptions (<i>module</i>), 17	INTEGRATION
honeycomb.integrationmanager.integration_utils (<i>module</i>), 18	honeycomb-integration-configure command line option, 4
honeycomb.integrationmanager.models (<i>module</i>), 19	honeycomb-integration-show command line option, 5
honeycomb.integrationmanager.registration (<i>module</i>), 20	Integration (<i>class in honey- comb.integrationmanager.models</i>), 19
honeycomb.integrationmanager.tasks (<i>mod- ule</i>), 20	IntegrationAlert (<i>class in honey- comb.integrationmanager.models</i>), 20
honeycomb.servicemanager (<i>module</i>), 24	IntegrationAlertStatuses (<i>class in honey- comb.integrationmanager.defs</i>), 16
honeycomb.servicemanager.base_service (<i>module</i>), 21	IntegrationMissingRequiredFieldError, 17
honeycomb.servicemanager.defs (<i>module</i>), 23	IntegrationNoMethodImplementationError, 17
honeycomb.servicemanager.error_messages (<i>module</i>), 23	IntegrationNotFound, 17
	IntegrationOutputFormatError, 17

IntegrationPackageError, 17
 IntegrationPollEventError, 18

INTEGRATIONS

honeycomb-integration-install
 command line option, 4
 honeycomb-integration-test command
 line option, 5
 honeycomb-integration-uninstall
 command line option, 6

IntegrationSendEventError, 18

IntegrationTestFailed, 18

IntegrationTypes (class in *honeycomb.integrationmanager.defs*), 17

is_valid_field_name() (in module *honeycomb.utils.config_utils*), 25

L

label (*honeycomb.decoymanager.models.AlertType* attribute), 16

LINUX (*honeycomb.servicemanager.models.OSFamilies* attribute), 24

list_local_plugins() (in module *honeycomb.utils.plugin_utils*), 27

list_remote_plugins() (in module *honeycomb.utils.plugin_utils*), 27

logger (*honeycomb.servicemanager.base_service.ServerCustomService* attribute), 22

M

MACOS (*honeycomb.servicemanager.models.OSFamilies* attribute), 24

makeRecord() (*honeycomb.cli.MyLogger* method), 29

manufacturer (*honeycomb.decoymanager.models.Alert* attribute), 15

msg_format (*honeycomb.exceptions.BaseHoneycombException* attribute), 30

msg_format (*honeycomb.exceptions.ConfigFieldMissing* attribute), 30

msg_format (*honeycomb.exceptions.ConfigFieldTypeMismatch* attribute), 30

msg_format (*honeycomb.exceptions.ConfigFieldValidationError* attribute), 30

msg_format (*honeycomb.exceptions.ConfigFileNotFound* attribute), 30

msg_format (*honeycomb.exceptions.ParametersFieldError* attribute), 30

msg_format (*honeycomb.exceptions.PathNotFound* attribute), 31

msg_format (*honeycomb.exceptions.PluginAlreadyInstalled* attribute), 31

msg_format (*honeycomb.exceptions.PluginNotFoundInOnLineRepo* attribute), 31

msg_format (*honeycomb.exceptions.PluginRepoConnectionError* attribute), 31

msg_format (*honeycomb.exceptions.RequiredFieldMissing* attribute), 31

msg_format (*honeycomb.integrationmanager.exceptions.IntegrationNotFound* attribute), 17

msg_format (*honeycomb.integrationmanager.exceptions.IntegrationSendEventError* attribute), 18

msg_format (*honeycomb.integrationmanager.exceptions.IntegrationTestFailed* attribute), 18

msg_format (*honeycomb.servicemanager.exceptions.ServiceNotFound* attribute), 23

msg_format (*honeycomb.servicemanager.exceptions.UnsupportedOS* attribute), 23

MyLogger (class in *honeycomb.cli*), 29

myRunner (class in *honeycomb.utils.daemon*), 25

N

name (*honeycomb.decoymanager.models.AlertType* attribute), 16

O

on_server_shutdown() (*honeycomb.servicemanager.base_service.DockerService* method), 21

on_server_shutdown() (*honeycomb.servicemanager.base_service.ServerCustomService* method), 22

on_server_start() (*honeycomb.servicemanager.base_service.DockerService* method), 21

on_server_start() (*honeycomb.servicemanager.base_service.ServerCustomService* method), 22

originating_hostname (*honeycomb.decoymanager.models.Alert* attribute), 15

originating_ip (*honeycomb.decoymanager.models.Alert* attribute), 15

originating_mac_address (*honeycomb.decoymanager.models.Alert* attribute), 16

originating_port (*honeycomb.decoymanager.models.Alert* attribute), 16

OSFamilies (class in *honeycomb.servicemanager.models*), 24

P

ParametersFieldError, 30

plugin_repo_line() (*honeycomb.servicemanager.base_service.DockerService* method), 22

`parse_plugin_args()` (in module `honeycomb.utils.plugin_utils`), 27
`password` (`honeycomb.decoymanager.models.Alert` attribute), 16
`PathNotFound`, 30
`PENDING` (`honeycomb.integrationmanager.defs.IntegrationAlertStatus` attribute), 17
`pid` (`honeycomb.decoymanager.models.Alert` attribute), 16
`PluginAlreadyInstalled`, 31
`PluginError`, 31
`PluginNotFoundInOnlineRepo`, 31
`PluginRepoConnectionError`, 31
`poll_for_updates()` (`honeycomb.integrationmanager.integration_utils.BaseIntegration` method), 18
`poll_integration_alert_data()` (in module `honeycomb.integrationmanager.tasks`), 21
`poll_integration_information_for_waiting_integrations()` (in module `honeycomb.integrationmanager.tasks`), 21
`POLLING` (`honeycomb.integrationmanager.defs.IntegrationAlertStatus` attribute), 17
`ppid` (`honeycomb.decoymanager.models.Alert` attribute), 16
`print_log()` (`honeycomb.utils.tailer.Tailer` method), 27
`print_named_log()` (`honeycomb.utils.tailer.Tailer` method), 28
`print_plugin_args()` (in module `honeycomb.utils.plugin_utils`), 27
`process_config()` (in module `honeycomb.utils.config_utils`), 25

R

`read_lines()` (`honeycomb.servicemanager.base_service.DockerService` method), 22
`register_integration()` (in module `honeycomb.integrationmanager.registration`), 20
`register_service()` (in module `honeycomb.servicemanager.registration`), 24
`request` (`honeycomb.decoymanager.models.Alert` attribute), 16
`RequiredFieldMissing`, 31
`run()` (`honeycomb.servicemanager.base_service.ServerCustomService` method), 22
`run_service()` (`honeycomb.servicemanager.base_service.ServerCustomService` method), 22

S

`search_json_log()` (in module `honeycomb.utils.wait`), 28
`send_alert_to_configured_integration()` (in module `honeycomb.integrationmanager.tasks`), 21
`send_alert_to_subscribed_integrations()` (in module `honeycomb.integrationmanager.tasks`), 21
`send_event()` (`honeycomb.integrationmanager.integration_utils.BaseIntegration` method), 19
`ServerCustomService` (class in `honeycomb.servicemanager.base_service`), 22
`SERVICE`
`honeycomb-service-run` command line option, 7
`honeycomb-service-show` command line option, 8
`honeycomb-service-stop` command line option, 8
`setup_logging()` (in module `honeycomb.cli`), 29
`signal_ready()` (`honeycomb.servicemanager.base_service.ServerCustomService` method), 23
`status` (`honeycomb.decoymanager.models.Alert` attribute), 16
`ServiceManagerException`, 23
`ServiceNotFound`, 23
`SERVICES`
`honeycomb-service-install` command line option, 6
`honeycomb-service-logs` command line option, 7
`honeycomb-service-status` command line option, 8
`honeycomb-service-test` command line option, 9
`honeycomb-service-uninstall` command line option, 9
`ServiceType` (class in `honeycomb.servicemanager.models`), 24
`setup_logging()` (in module `honeycomb.cli`), 29
`signal_ready()` (`honeycomb.servicemanager.base_service.ServerCustomService` method), 23
`status` (`honeycomb.decoymanager.models.Alert` attribute), 16
`STATUS_ALERT` (`honeycomb.decoymanager.models.Alert` attribute), 15
`STATUS_IGNORED` (`honeycomb.decoymanager.models.Alert` attribute), 15
`STATUS_MUTED` (`honeycomb.decoymanager.models.Alert` attribute), 15

STDERRLOG (in module *honeycomb.servicemanager.defs*), 23
 stop() (*honeycomb.utils.tailer.Tailer* method), 28

T

Tailer (class in *honeycomb.utils.tailer*), 27
 test_connection() (honeycomb.integrationmanager.integration_utils.BaseIntegration method), 19
 thread_server (honeycomb.servicemanager.base_service.ServerCustomService attribute), 23
 TimeoutException, 28
 timestamp (honeycomb.decoymanager.models.Alert attribute), 16
 transport_protocol (honeycomb.decoymanager.models.Alert attribute), 16

U

uid (honeycomb.decoymanager.models.Alert attribute), 16
 uninstall_plugin() (in module *honeycomb.utils.plugin_utils*), 27
 UnsupportedOS, 23
 username (honeycomb.decoymanager.models.Alert attribute), 16

V

validate_config() (in module *honeycomb.utils.config_utils*), 25
 validate_config_parameters() (in module *honeycomb.utils.config_utils*), 25
 validate_field() (in module *honeycomb.utils.config_utils*), 25
 validate_field_matches_type() (in module *honeycomb.utils.config_utils*), 25

W

wait_until() (in module *honeycomb.utils.wait*), 28
 WINDOWS (honeycomb.servicemanager.models.OSFamilies attribute), 24