
Honeycomb Documentation

Release 0.0.10

Cymmetria

Apr 27, 2018

Contents

1	Running Honeycomb from command line	3
1.1	Honeycomb	3
1.1.1	integration	4
1.1.2	service	4
2	Running honeycomb in a container	5
3	API Reference	7
3.1	honeycomb package	7
3.1.1	Subpackages	7
3.1.2	Submodules	20
3.1.3	honeycomb.cli module	20
3.1.4	honeycomb.defs module	20
3.1.5	honeycomb.error_messages module	21
3.1.6	honeycomb.exceptions module	21
	Python Module Index	23

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

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

Writing new honeypot services and integrations for honeycomb is super easy! See the plugins repo for more info.

Running Honeycomb from command line

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`

1.1.1 integration

Honeycomb integration commands.

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

1.1.2 service

Honeycomb service commands.

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

Running honeycomb in a container

The rationale of container support is to allow rapid configuration and deployment so 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 integration at ease.

To launch a honeycomb service with 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 a 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](#)

```
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
```


3.1 honeycomb package

3.1.1 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

Hooneycomb defs and constants.

```
class honeycomb.decoymanager.models.Alert(alert_type, id=NOTHING, status=2, times-  
                                         tamp=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, label, service_type)
    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.

exception honeycomb.integrationmanager.exceptions.IntegrationNoMethodImplementationError(*args,
                                                                                          **kwargs)
    Bases: honeycomb.exceptions.PluginError
    IntegrationNoMethodImplementationError.

exception honeycomb.integrationmanager.exceptions.IntegrationNotFound(*args,
                                                                        **kwargs)
    Bases: honeycomb.exceptions.PluginError
    Integration not found.
    msg_format = 'Cannot find integration named {}, try installing it?'

exception honeycomb.integrationmanager.exceptions.IntegrationOutputFormatError(*args,
                                                                                  **kwargs)
    Bases: honeycomb.exceptions.PluginError
    IntegrationOutputFormatError.

exception honeycomb.integrationmanager.exceptions.IntegrationPackageError(*args,
                                                                            **kwargs)
    Bases: honeycomb.exceptions.PluginError
    IntegrationPackageError.

exception honeycomb.integrationmanager.exceptions.IntegrationPollEventError(*args,
                                                                               **kwargs)
    Bases: honeycomb.exceptions.PluginError
    IntegrationPollEventError.

```

```
exception honeycomb.integrationmanager.exceptions.IntegrationSendEventError (*args,  
                                                                           **kwargs)  
    Bases: honeycomb.exceptions.PluginError  
    IntegrationSendEventError.  
    msg_format = 'Error sending integration event: {}'  
exception honeycomb.integrationmanager.exceptions.IntegrationTestFailed (*args,  
                                                                           **kwargs)  
    Bases: honeycomb.exceptions.PluginError  
    Integration not found.  
    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.  
    Will be overridden by output plugins.  
    format_output_data (output_data)  
        format_output_data.  
    poll_for_updates (integration_output_data)  
        poll_for_updates.  
    send_event (required_alert_fields)  
        Send event.  
    test_connection (data)  
        test_connection.
```

honeycomb.integrationmanager.models module

Honeycomb integration models.

```
class honeycomb.integrationmanager.models.ConfiguredIntegration (name, path,  
                                                                    integration,  
                                                                    send_muted=False,  
                                                                    created_at=NOTHING)  
    Bases: object  
    Configured integration model.
```



```
class honeycomb.integrationmanager.models.Integration (parameters,
                                                    display_name, required_fields,
                                                    polling_enabled, integration_type,
                                                    max_send_retries,
                                                    supported_event_types,
                                                    test_connection_enabled,
                                                    module=None, description=None,
                                                    polling_duration=0)
```

Bases: object

Integration model.

```
class honeycomb.integrationmanager.models.IntegrationAlert (alert, status, retries,
                                                                configured_integration)
```

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,
                                                                              configured_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.ServerCustomService (alert_types, ser-vice_args={})`

Bases: `multiprocessing.context.Process`

Custom Service Class.

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

Parameters `service_args` – Validated dictionary of service arguments (see: `honeycomb.Honeycomb.parse_service_args()`)

add_alert_to_queue (`alert_dict`)
Log alert and send to integrations.

emit (`**kwargs`)
Send alerts to logfile.

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

logger = `<logging.Logger object>`

on_server_shutdown ()
Shutdown function of the server.

Override this and take care of gracefully shutting down you 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

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

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 Server Manager Exception.

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

Specified service does not exist.

`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.

`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, ports, label, al-
                                                low_many, supported_os_families,
                                                alert_types=[])

    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.

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.

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

Copy a single file.

:param:src: Source name :param:dst: Destination name

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

Copy a full directory structure.

:param:src: Source path :param:dst: Destination path :param:symlinks: Copy symlinks :param:ignore: Subdirs/filenames to ignore

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

Return path to plugin.

:param:home: Path to honeycomb home :param:plugin_type: Type of plugin (`honeycomb.defs.SERVICES` or `honeycomb.defs.INTEGRATIONS`) :param:plugin_name: Name of plugin :param: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()`. :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.

:param:pkgpath: Name of plugin to be downloaded from online repo or path to plugin folder or zip file.
:param:install_path: Path where plugin will be installed. :param: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.

:param:pkgpath: Path to package to uninstall (delete) :param:force: Force uninstall without asking

honeycomb.utils.tailer module

Honeycomb service log tailer.

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

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.validators module

Hooneycomb generic validators.

honeycomb.utils.validators.**validate_ip_or_hostname** (*value*)

IP/Host parameter validator.

honeycomb.utils.validators.**validate_port** (*value*)

Validate port is in standard range.

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 retrys
- **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.

3.1.2 Submodules

3.1.3 honeycomb.cli module

Honeycomb Command Line Interface.

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

Bases: logging.Logger

Custom Logger.

```
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.

3.1.4 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.

3.1.5 honeycomb.error_messages module

Honeycomb generic error messages.

3.1.6 honeycomb.exceptions module

Honeycomb Exceptions.

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

Bases: `click.exceptions.ClickException`

Base Exception.

```
msg_format = None
```

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

Bases: `honeycomb.exceptions.ConfigValidationError`

Field is missing from config file.

```
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.

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

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

Bases: `honeycomb.exceptions.ConfigValidationError`

Error validating config field.

```
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.

```
msg_format = 'Missing file {}'
```

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

Bases: `honeycomb.exceptions.BaseHoneycombException`

Base config validation error.

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

Error validating parameter.

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

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

Specified path was not found.

msg_format = 'Cannot find path {}'

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

Plugin already installed.

msg_format = '{} is already installed'

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

Base Plugin Exception.

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

Plugin not found in online repo.

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.

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.

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

h

`honeycomb.cli`, 20

`honeycomb.commands.integration.configure`, 8

`honeycomb.commands.integration.install`, 8

`honeycomb.commands.integration.list`, 8

`honeycomb.commands.integration.show`, 8

`honeycomb.commands.integration.test`, 8

`honeycomb.commands.integration.uninstall`, 9

`honeycomb.commands.service.install`, 7

`honeycomb.commands.service.list`, 7

`honeycomb.commands.service.logs`, 7

`honeycomb.commands.service.run`, 7

`honeycomb.commands.service.show`, 8

`honeycomb.commands.service.status`, 8

`honeycomb.commands.service.stop`, 8

`honeycomb.commands.service.test`, 8

`honeycomb.commands.service.uninstall`, 8

`honeycomb.decoymanager`, 10

`honeycomb.decoymanager.models`, 9

`honeycomb.defs`, 20

`honeycomb.error_messages`, 21

`honeycomb.exceptions`, 21

`honeycomb.integrationmanager`, 14

`honeycomb.integrationmanager.defs`, 10

`honeycomb.integrationmanager.error_messages`, 11

`honeycomb.integrationmanager.exceptions`, 11

`honeycomb.integrationmanager.integration_utils`, 12

`honeycomb.integrationmanager.models`, 12

`honeycomb.integrationmanager.registration`, 13

`honeycomb.integrationmanager.tasks`, 13

`honeycomb.servicemanager`, 16

`honeycomb.servicemanager.base_service`, 14

`honeycomb.servicemanager.defs`, 15

`honeycomb.servicemanager.error_messages`, 15

`honeycomb.servicemanager.exceptions`, 15

`honeycomb.servicemanager.models`, 15

`honeycomb.servicemanager.registration`, 16

`honeycomb.utils`, 20

`honeycomb.utils.config_utils`, 16

`honeycomb.utils.daemon`, 17

`honeycomb.utils.plugin_utils`, 17

`honeycomb.utils.tailer`, 19

`honeycomb.utils.validators`, 19

`honeycomb.utils.wait`, 19

Symbols

-iamroot
 Honeycomb command line option, 3
 -version
 Honeycomb command line option, 3
 -H, --home <home>
 Honeycomb command line option, 3
 -c, --config <config>
 Honeycomb command line option, 3
 -v, --verbose
 Honeycomb command line option, 3

A

add_alert_to_queue() (honeycomb.servicemanager.base_service.ServerCustomService method), 14
 additional_fields (honeycomb.decoymanager.models.Alert attribute), 9
 address (honeycomb.decoymanager.models.Alert attribute), 9
 Alert (class in honeycomb.decoymanager.models), 9
 ALERT_STATUS (honeycomb.decoymanager.models.Alert attribute), 9
 alert_type (honeycomb.decoymanager.models.Alert attribute), 9
 AlertType (class in honeycomb.decoymanager.models), 10
 ALL (honeycomb.servicemanager.models.OSFamilies attribute), 16
 all_labels() (honeycomb.defs.IBaseType class method), 21
 all_names() (honeycomb.defs.IBaseType class method), 21
 ALLOWED_PROTOCOLS (in module honeycomb.servicemanager.defs), 15

B

BaseCollection (class in honeycomb.defs), 20
 BaseHoneycombException, 21
 BaseIntegration (class in honeycomb.integrationmanager.integration_utils), 12
 BaseNameLabel (class in honeycomb.defs), 20

C

cmd (honeycomb.decoymanager.models.Alert attribute), 9
 config_field_type() (in module honeycomb.utils.config_utils), 16
 CONFIG_FILE_NAME (in module honeycomb.defs), 20
 ConfigField (class in honeycomb.defs), 20
 ConfigFieldMissing, 21
 ConfigFieldTypeMismatch, 21
 ConfigFieldValidationError, 21
 ConfigFileNotFound, 21
 configure_integration() (in module honeycomb.integrationmanager.tasks), 13
 ConfiguredIntegration (class in honeycomb.integrationmanager.models), 12
 ConfigValidationError, 21
 copy_file() (in module honeycomb.utils.plugin_utils), 17
 copy_tree() (in module honeycomb.utils.plugin_utils), 17
 create_integration_alert_and_call_send() (in module honeycomb.integrationmanager.tasks), 13
 CTErrors, 17

D

decoy_hostname (honeycomb.decoymanager.models.Alert attribute), 9
 decoy_ip4 (honeycomb.decoymanager.models.Alert attribute), 9
 decoy_name (honeycomb.decoymanager.models.Alert attribute), 9
 decoy_os (honeycomb.decoymanager.models.Alert attribute), 9

- dest_ip (honeycomb.decoymanager.models.Alert attribute), 9
- dest_port (honeycomb.decoymanager.models.Alert attribute), 9
- domain (honeycomb.decoymanager.models.Alert attribute), 9
- DONE (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 10
- ## E
- emit() (honeycomb.servicemanager.base_service.ServerCustomService method), 14
- end_timestamp (honeycomb.decoymanager.models.Alert attribute), 9
- environment variable
DEBUG, 3
- ERROR_MISSING_SEND_FIELDS (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 10
- ERROR_POLLING (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 10
- ERROR_POLLING_FORMATTING (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 10
- ERROR_SENDING (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 10
- ERROR_SENDING_FORMATTING (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 11
- event_description (honeycomb.decoymanager.models.Alert attribute), 9
- EVENT_OUTPUT (honeycomb.integrationmanager.defs.IntegrationTypes attribute), 11
- event_type (honeycomb.decoymanager.models.Alert attribute), 9
- ## F
- file_accessed (honeycomb.decoymanager.models.Alert attribute), 9
- follow_file() (honeycomb.utils.tailer.Tailer method), 19
- format_output_data() (honeycomb.integrationmanager.integration_utils.BaseIntegrationUtils method), 12
- ## G
- get_config_parameters() (in module honeycomb.utils.config_utils), 16
- get_current_datetime_utc() (in module honeycomb.integrationmanager.tasks), 13
- get_integration_module() (in module honeycomb.integrationmanager.registration), 13
- get_plugin_path() (in module honeycomb.utils.plugin_utils), 17
- get_select_items() (in module honeycomb.utils.plugin_utils), 18
- get_service_module() (in module honeycomb.servicemanager.registration), 16
- get_truetype() (in module honeycomb.utils.config_utils), 16
- get_valid_configured_integrations() (in module honeycomb.integrationmanager.tasks), 13
- GITHUB_RAW_URL (in module honeycomb.defs), 21
- ## H
- Honeycomb command line option
-iamroot, 3
-version, 3
-H, -home <home>, 3
-c, -config <config>, 3
-v, -verbose, 3
- honeycomb.cli (module), 20
- honeycomb.commands.integration.configure (module), 8
- honeycomb.commands.integration.install (module), 8
- honeycomb.commands.integration.list (module), 8
- honeycomb.commands.integration.show (module), 8
- honeycomb.commands.integration.test (module), 8
- honeycomb.commands.integration.uninstall (module), 9
- honeycomb.commands.service.install (module), 7
- honeycomb.commands.service.list (module), 7
- honeycomb.commands.service.logs (module), 7
- honeycomb.commands.service.run (module), 7
- honeycomb.commands.service.show (module), 8
- honeycomb.commands.service.status (module), 8
- honeycomb.commands.service.stop (module), 8
- honeycomb.commands.service.test (module), 8
- honeycomb.commands.service.uninstall (module), 8
- honeycomb.decoymanager (module), 10
- honeycomb.decoymanager.models (module), 9
- honeycomb.defs (module), 20
- honeycomb.error_messages (module), 21
- honeycomb.exceptions (module), 21
- honeycomb.integrationmanager (module), 14
- honeycomb.integrationmanager.defs (module), 10
- honeycomb.integrationmanager.error_messages (module), 11
- honeycomb.integrationmanager.exceptions (module), 11
- honeycomb.integrationmanager.integration_utils (module), 12
- honeycomb.integrationmanager.models (module), 12
- honeycomb.integrationmanager.registration (module), 13
- honeycomb.integrationmanager.tasks (module), 13
- honeycomb.servicemanager (module), 16
- honeycomb.servicemanager.base_service (module), 14

[honeycomb.servicemanager.defs \(module\)](#), 15
[honeycomb.servicemanager.error_messages \(module\)](#), 15
[honeycomb.servicemanager.exceptions \(module\)](#), 15
[honeycomb.servicemanager.models \(module\)](#), 15
[honeycomb.servicemanager.registration \(module\)](#), 16
[honeycomb.utils \(module\)](#), 20
[honeycomb.utils.config_utils \(module\)](#), 16
[honeycomb.utils.daemon \(module\)](#), 17
[honeycomb.utils.plugin_utils \(module\)](#), 17
[honeycomb.utils.tailer \(module\)](#), 19
[honeycomb.utils.validators \(module\)](#), 19
[honeycomb.utils.wait \(module\)](#), 19

I

[IBaseType \(class in honeycomb.defs\)](#), 21
[id \(honeycomb.decoymanager.models.Alert attribute\)](#), 9
[image_file \(honeycomb.decoymanager.models.Alert attribute\)](#), 9
[image_md5 \(honeycomb.decoymanager.models.Alert attribute\)](#), 9
[image_path \(honeycomb.decoymanager.models.Alert attribute\)](#), 9
[image_sha256 \(honeycomb.decoymanager.models.Alert attribute\)](#), 9
[IN_POLLING \(honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute\)](#), 11
[install_deps\(\) \(in module honeycomb.utils.plugin_utils\)](#), 18
[install_dir\(\) \(in module honeycomb.utils.plugin_utils\)](#), 18
[install_from_repo\(\) \(in module honeycomb.utils.plugin_utils\)](#), 18
[install_from_zip\(\) \(in module honeycomb.utils.plugin_utils\)](#), 18
[install_plugin\(\) \(in module honeycomb.utils.plugin_utils\)](#), 18
[Integration \(class in honeycomb.integrationmanager.models\)](#), 12
[IntegrationAlert \(class in honeycomb.integrationmanager.models\)](#), 13
[IntegrationAlertStatuses \(class in honeycomb.integrationmanager.defs\)](#), 10
[IntegrationMissingRequiredFieldError](#), 11
[IntegrationNoMethodImplementationError](#), 11
[IntegrationNotFound](#), 11
[IntegrationOutputFormatError](#), 11
[IntegrationPackageError](#), 11
[IntegrationPollEventError](#), 11
[IntegrationSendEventError](#), 12
[IntegrationTestFailed](#), 12
[IntegrationTypes \(class in honeycomb.integrationmanager.defs\)](#), 11
[is_valid_field_name\(\) \(in module honeycomb.utils.config_utils\)](#), 17

L

[label \(honeycomb.decoymanager.models.AlertType attribute\)](#), 10
[LINUX \(honeycomb.servicemanager.models.OSFamilies attribute\)](#), 16
[list_local_plugins\(\) \(in module honeycomb.utils.plugin_utils\)](#), 18
[list_remote_plugins\(\) \(in module honeycomb.utils.plugin_utils\)](#), 18
[logger \(honeycomb.servicemanager.base_service.ServerCustomService attribute\)](#), 14

M

[MACOS \(honeycomb.servicemanager.models.OSFamilies attribute\)](#), 16
[makeRecord\(\) \(honeycomb.cli.MyLogger method\)](#), 20
[manufacturer \(honeycomb.decoymanager.models.Alert attribute\)](#), 9
[msg_format \(honeycomb.exceptions.BaseHoneycombException attribute\)](#), 21
[msg_format \(honeycomb.exceptions.ConfigFieldMissing attribute\)](#), 21
[msg_format \(honeycomb.exceptions.ConfigFieldTypeMismatch attribute\)](#), 21
[msg_format \(honeycomb.exceptions.ConfigFieldValidationError attribute\)](#), 21
[msg_format \(honeycomb.exceptions.ConfigFileNotFound attribute\)](#), 21
[msg_format \(honeycomb.exceptions.ParametersFieldError attribute\)](#), 22
[msg_format \(honeycomb.exceptions.PathNotFound attribute\)](#), 22
[msg_format \(honeycomb.exceptions.PluginAlreadyInstalled attribute\)](#), 22
[msg_format \(honeycomb.exceptions.PluginNotFoundInOnlineRepo attribute\)](#), 22
[msg_format \(honeycomb.exceptions.PluginRepoConnectionError attribute\)](#), 22
[msg_format \(honeycomb.exceptions.RequiredFieldMissing attribute\)](#), 22
[msg_format \(honeycomb.integrationmanager.exceptions.IntegrationNotFound attribute\)](#), 11
[msg_format \(honeycomb.integrationmanager.exceptions.IntegrationSendEvent attribute\)](#), 12
[msg_format \(honeycomb.integrationmanager.exceptions.IntegrationTestFailed attribute\)](#), 12
[msg_format \(honeycomb.servicemanager.exceptions.ServiceNotFound attribute\)](#), 15
[msg_format \(honeycomb.servicemanager.exceptions.UnsupportedOS attribute\)](#), 15
[MyLogger \(class in honeycomb.cli\)](#), 20
[myRunner \(class in honeycomb.utils.daemon\)](#), 17

N

name (honeycomb.decoymanager.models.AlertType attribute), 10

O

on_server_shutdown() (honeycomb.servicemanager.base_service.ServerCustomService method), 14

on_server_start() (honeycomb.servicemanager.base_service.ServerCustomService method), 14

originating_hostname (honeycomb.decoymanager.models.Alert attribute), 10

originating_ip (honeycomb.decoymanager.models.Alert attribute), 10

originating_mac_address (honeycomb.decoymanager.models.Alert attribute), 10

originating_port (honeycomb.decoymanager.models.Alert attribute), 10

OSFamilies (class in honeycomb.servicemanager.models), 15

P

ParametersFieldError, 21

parse_plugin_args() (in module honeycomb.utils.plugin_utils), 18

password (honeycomb.decoymanager.models.Alert attribute), 10

PathNotFound, 22

PENDING (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 11

pid (honeycomb.decoymanager.models.Alert attribute), 10

PluginAlreadyInstalled, 22

PluginError, 22

PluginNotFoundInOnlineRepo, 22

PluginRepoConnectionError, 22

poll_for_updates() (honeycomb.integrationmanager.integration_utils.BaseIntegration method), 12

poll_integration_alert_data() (in module honeycomb.integrationmanager.tasks), 13

poll_integration_information_for_waiting_integration_alerts() (in module honeycomb.integrationmanager.tasks), 14

POLLING (honeycomb.integrationmanager.defs.IntegrationAlertStatuses attribute), 11

ppid (honeycomb.decoymanager.models.Alert attribute), 10

print_log() (honeycomb.utils.tailer.Tailer method), 19

print_named_log() (honeycomb.utils.tailer.Tailer method), 19

print_plugin_args() (in module honeycomb.utils.plugin_utils), 18

process_config() (in module honeycomb.utils.config_utils), 17

R

register_integration() (in module honeycomb.integrationmanager.registration), 13

register_service() (in module honeycomb.servicemanager.registration), 16

request (honeycomb.decoymanager.models.Alert attribute), 10

RequiredFieldMissing, 22

run() (honeycomb.servicemanager.base_service.ServerCustomService method), 14

run_service() (honeycomb.servicemanager.base_service.ServerCustomService method), 15

S

search_json_log() (in module honeycomb.utils.wait), 19

send_alert_to_configured_integration() (in module honeycomb.integrationmanager.tasks), 14

send_alert_to_subscribed_integrations() (in module honeycomb.integrationmanager.tasks), 14

send_event() (honeycomb.integrationmanager.integration_utils.BaseIntegration method), 12

ServerCustomService (class in honeycomb.servicemanager.base_service), 14

service_type (honeycomb.decoymanager.models.Alert attribute), 10

ServiceManagerException, 15

ServiceNotFound, 15

ServiceType (class in honeycomb.servicemanager.models), 16

setup_logging() (in module honeycomb.cli), 20

signal_ready() (honeycomb.servicemanager.base_service.ServerCustomService method), 15

status (honeycomb.decoymanager.models.Alert attribute), 10

STATUS_ALERT (honeycomb.decoymanager.models.Alert attribute), 9

STATUS_IGNORED (honeycomb.decoymanager.models.Alert attribute), 9

STATUS_MUTED (honeycomb.decoymanager.models.Alert attribute), 9

STATUS_PENDING

STDERRLOG (in module honeycomb.servicemanager.defs), 15

stop() (honeycomb.utils.tailer.Tailer method), 19

T

Tailer (class in `honeycomb.utils.tailer`), [19](#)
`test_connection()` (`honeycomb.integrationmanager.integration_utils.BaseIntegration` method), [12](#)
`TimeoutException`, [19](#)
`timestamp` (`honeycomb.decoymanager.models.Alert` attribute), [10](#)
`transport_protocol` (`honeycomb.decoymanager.models.Alert` attribute), [10](#)

U

`uid` (`honeycomb.decoymanager.models.Alert` attribute), [10](#)
`uninstall_plugin()` (in module `honeycomb.utils.plugin_utils`), [18](#)
`UnsupportedOS`, [15](#)
`username` (`honeycomb.decoymanager.models.Alert` attribute), [10](#)

V

`validate_config()` (in module `honeycomb.utils.config_utils`), [17](#)
`validate_config_parameters()` (in module `honeycomb.utils.config_utils`), [17](#)
`validate_field()` (in module `honeycomb.utils.config_utils`), [17](#)
`validate_field_matches_type()` (in module `honeycomb.utils.config_utils`), [17](#)
`validate_ip_or_hostname()` (in module `honeycomb.utils.validators`), [19](#)
`validate_port()` (in module `honeycomb.utils.validators`), [19](#)

W

`wait_until()` (in module `honeycomb.utils.wait`), [19](#)
`WINDOWS` (`honeycomb.servicemanager.models.OSFamilies` attribute), [16](#)