
ps.herald

Release 1.4.2

Thomas Setz

Feb 20, 2021

CONTENTS:

1	ps herald :Version: 1.4.2	1
1.1	ps_herald	4
1.2	ps_bridge	4
1.3	ps_neelix	5
1.4	Install the package within a private virtualenv	5
1.5	Integrating restart and invocation of neelix/herald/bridge with a local crontab	5
2	herald package	7
2.1	Submodules	7
2.2	herald.angular_api module	7
2.3	herald.bare_html_api module	7
2.4	herald.database module	7
2.5	herald.model module	8
2.6	herald.ps_bridge module	9
2.7	herald.ps_herald module	9
2.8	herald.ps_neelix module	9
2.9	Subpackages	10
2.10	Module contents	12
3	Indices and tables	13
	Python Module Index	15
	Index	17

PS HERALD :VERSION: 1.4.2

The ps.herald package provides three tools usable to monitor the behaviour of distributed applications

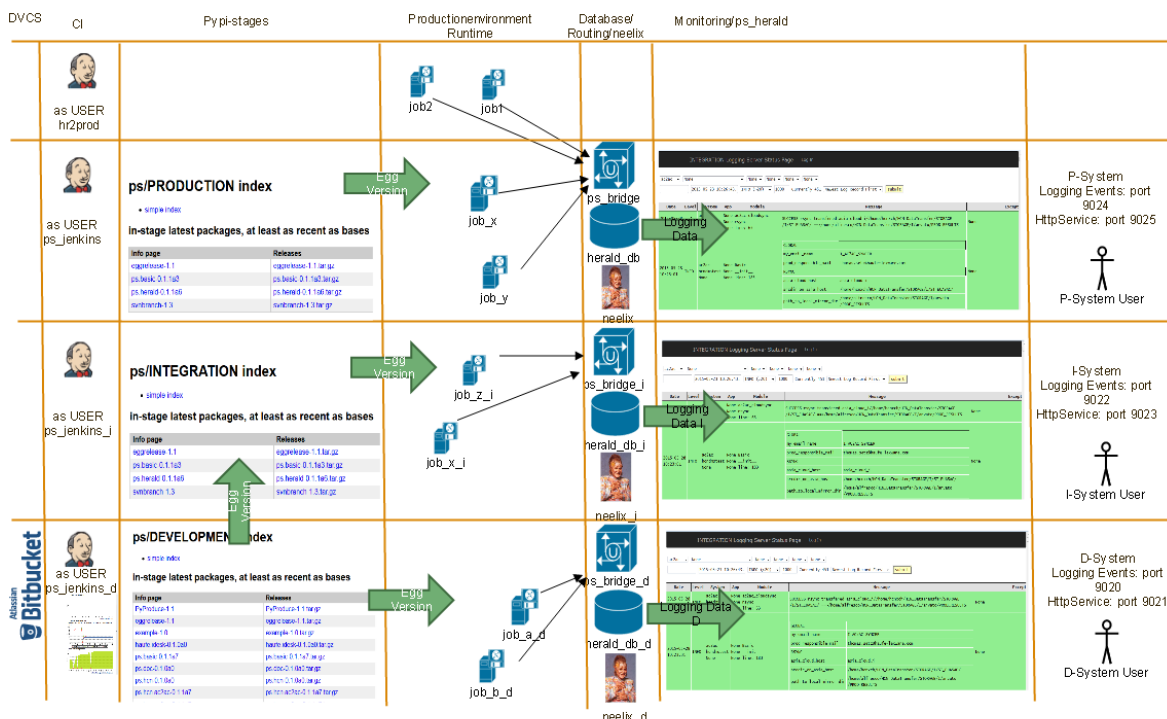
Herald's functionality is based on the Config module, defined in the Within that documentation you'll be enabled to use the ps.basic.Config mechanisms (especially the logging mechanism) within your application.

The **Config** module (among other things) enhances/uses the standard python logging package - especially the ability to log messages to a stream-socket.

This package/the herald package adds mechanisms to:

- put these messages into a central database (bridge)
- route/bridge these messages to additional destinations (bridge)
- display in a web-server (herald)
- take actions on special messages (neelix)
- take actions on lost system heartbeat (neelix)

The tasks of the tools implemented within that package will be explained with the following picture .



If a program is running - let that program be named job or service (within the picture those programs are named job_[a,b,x,1,2] in column 4 (Production environment Runtime)) it emits logging-messages (via the ps.basic.Config module).

Those logging messages are :

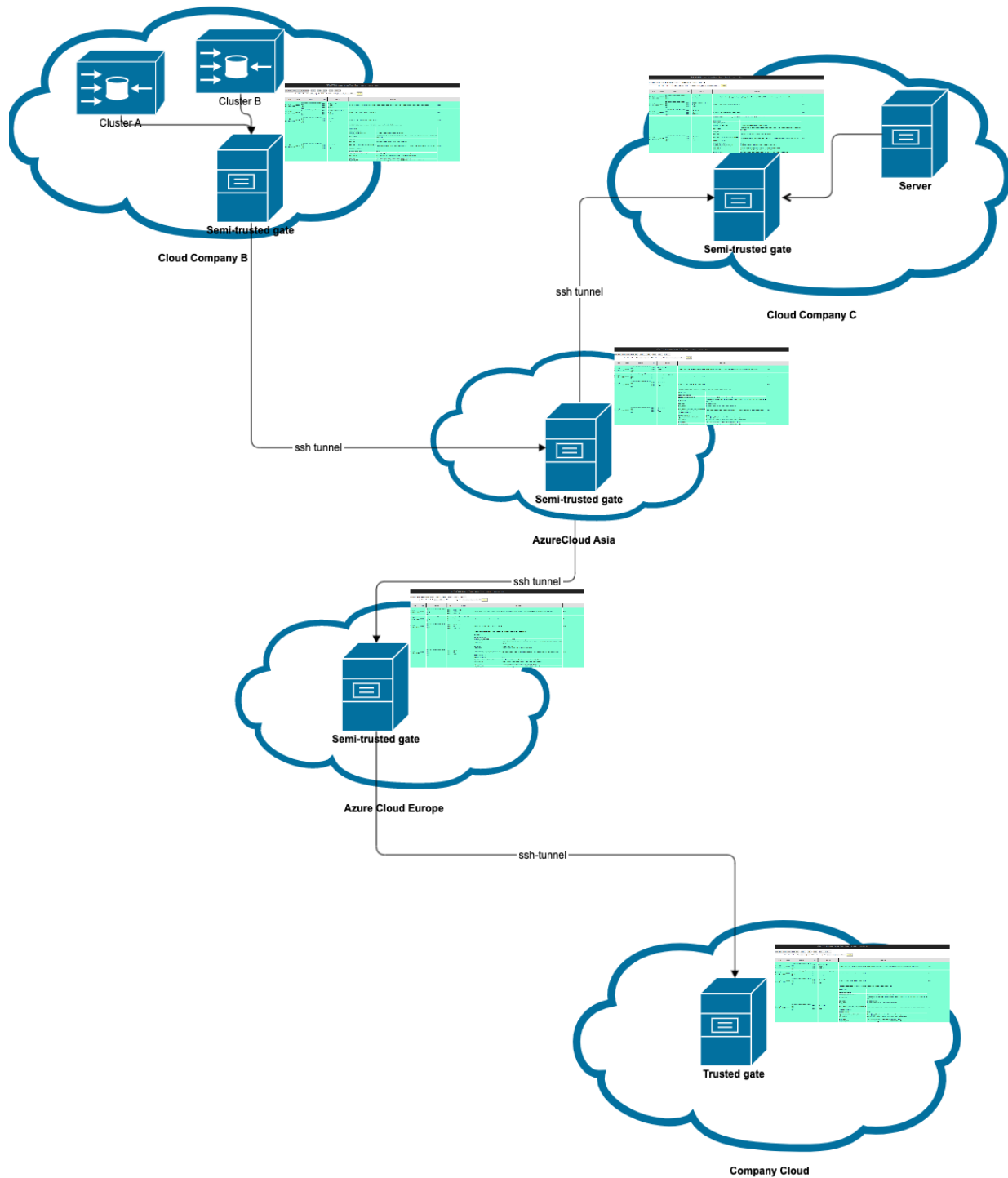
received by the ps_bridge daemon and inserted into a local sqlite database e.g. herald_db_d.

Those messages/events can be displayed/searched via herald.

Those messages/events can be analyzed and further events could be triggered by neelix.

Note: The ps_bridge module allows for “duplication” of event streams - meaning it is possible to insert logging messages into the local database AND to additionally send that message to another bridge (possibly residing on another machine/port). It is possible to route those message through different ssh tunnels - so that on the different levels of trust - the messages of the behind system could be analyzed. This is shown in the following picture.

Network Diagram



1.1 ps_herald

ps_herald is a flask-based (<http://flask.pocoo.org/>) web micro-framework, enabling us to display logging messages of the participating systems/services.

It uses `sqlalchemy` to access the `sqlite` database.

ps_herald listens for html-requests (normally issued by an user-interface e.g. firefox/safari/chrome ...) and returns corresponding log-messages.

The following picture shows a screenshot for the usage of herald within google-chrome . . .

PRODUCTION Logging Server Status Page
Log Out
clean > 30 days

products2intern_prodstrecke-zope [None] [None] [None] [None] [None]

2017-11-02 14:15:20:
[ALL (>0)]
1000
Currently 1000
[Newest Log Record First]
[submit]

Date	Level	System	App	Module	Message	
017-11-07 4:02:02.691521	DEBUG	products2intern_prodstrecke-zope jdev2 None	None None None	Basic 0.3.39 __exit__ line: 373	remove lock_file /home/jeeapp/ps/P/ps.deploy-products2intern_20170927_01/products2intern_prodstrecke-zope_lock	None
017-11-07 4:02:02.689840	DEBUG	products2intern_prodstrecke-zope jdev2 None	None None None	products2intern 0.0.33 main line: 773	return in main ==> exit code : with 0	None
017-11-07 4:02:02.687014	DEBUG	products2intern_prodstrecke-zope jdev2 None	None None None	fsm 0.3.39 run line: 135	Leave run, as FINISHED is a final state	None
017-11-07 4:02:02.684383	DEBUG	products2intern_prodstrecke-zope jdev2 None	None None None	fsm 0.3.39 run line: 132	PRODUCTS2INTERN.run new step: FINISHED s compute function left context as	None
					SETUPFILES	
					GLOBAL_THS_PARAMS	
					PATH_TO_CFG_EXECUTABLE	
					CONFIG_FILE	
					SYS_NAME	
					RKF_LINKDIR	
					RKF_LINKDIR_LIST_OF_ZOPE_EXECUTABLES	
					IMPORTSUCCESSLIST	
					NAME_OF_SETUP_DIR	
					FQ_PATH_OF_AURORA_CONFIG_SH	
					REGISTRY_URL	
					SETUP_DIR	
					MY_EMAIL_NAME	

Herald displays the messages of the “correlated” services in a table-format.

From left to right you see:

- date and time of insertion of the logging message into the database
- logging level
- service-name , machine-name and user_defined system-name
- Application-specific logging fields
- package name and version, function name, line number
- logging message
- stack-trace

1.2 ps_bridge

The ps_bridge executable:

- listens on a stage specific network socket
- receives logging messages
 - puts them into a local sqlite database
 - eventually (bridge-mode) forwards the message to another socket endpoint.

1.3 ps_neelix

The ps_neelix executable reads it's config file where services and 'reactions' are defined.

Next it consults it's stage specific sqlite database and checks if 'reactions' are needed e.g. email a responsible admin, that the heartbeat of a service has been lost.

1.4 Install the package within a private virtualenv

To install the herald-package and it's services, we first create a new directory e.g. \$HOME/nodename/ps.herald_\$BUILD_DATE, copy the following template to that directory and than execute the skript.

```
#The Content of $HOME/.pip/pip.conf is
#
#[global]
#index_url = https://vl-pypi.haufe-ep.de/ps/${DEV_STAGE}/+simple/
#
# and so points to the staging devpi server

/bin/rm -fR oenvv
mv venv oenvv

virtualenv venv
source venv/bin/activate

pip install ps.basic
pip install ps.herald
```

Maybe we establish a link in \$HOME/nodename so that, in called scripts, we could use that link.

```
ln -s ps.herald ps.herald_$BUILD_DATE
```

1.5 Integrating restart and invocation of neelix/herald/bridge with a local crontab

Beneath an example how herald,ssh-tunnel, bridge and neelix currently are (re)started on sulu

```
SHELL=/bin/bash
MAILTO="thomas.setz@haufe-lexware.com"

# The ssh tunnel for the bridge bringing the data on port 9017 of the eu_cloud_
↪machine to the loacal bridge
01,17,24,36,45,51,10 * * * 0-6      ssh -f -oExitOnForwardFailure=yes -R_
↪9017:localhost:9024 eu_cloud -N > /home/hcn/ssh_tunnel.log 2>&1
4 */2 * * 0-7                      export DEV_STAGE=PRODUCTION; cd /home/hcn/HCN_
↪DataTransfer/u14_20170801; source venv/bin/activate && ps_neelix
↪ > neelix.log 2>&1
04,16,36,44,51,01 * * * 0-7      export DEV_STAGE=PRODUCTION; cd /home/hcn/HCN_
↪DataTransfer/u14_20170801; source venv/bin/activate && ps.herald
↪ > herald.log 2>&1
```

(continues on next page)

(continued from previous page)

```
05,15,21,34,46,50,02 * * * 0-7    export DEV_STAGE=PRODUCTION; cd /home/hcn/HCN_  
↪DataTransfer/u14_20170801; source venv/bin/activate && ps_bridge -s 1309600 -r 1024_  
↪ > bridge.log 2>&1
```

HERALD PACKAGE

2.1 Submodules

2.2 `herald.angular_api` module

```
ps.herald.angular_api.hello()  
ps.herald.angular_api.hello_to_ps_basic_logger()  
ps.herald.angular_api.index()  
ps.herald.angular_api.internal_error(exception)  
ps.herald.angular_api.list()  
ps.herald.angular_api.options()
```

2.3 `herald.bare_html_api` module

```
ps.herald.bare_html_api.hello()  
ps.herald.bare_html_api.hello_to_ps_basic_logger()  
ps.herald.bare_html_api.index()  
ps.herald.bare_html_api.isset(param1,param2)  
ps.herald.bare_html_api.rm_logs()  
ps.herald.bare_html_api.set_search_params(request)
```

2.4 `herald.database` module

```
ps.herald.database.close_engine(e=None)  
ps.herald.database.get_engine()  
    current_app.config['DATABASE'] is already mapped to the ps.basic.Config.herald_sqlite_filename  
ps.herald.database.get_session()  
ps.herald.database.init_app(app)  
ps.herald.database.init_db()
```

2.5 herald.model module

The models.

```
class ps.herald.model.HeartBeat (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base

    Store the HeartBeat of the different systems.

    id

    newest_heartbeat

    system_id
```

```
class ps.herald.model.Log (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base
```

The logging message.

The model currently is build around the data structure, the standard python logging mechanism uses to send logging Events across the **class logging.StreamHandler()**.

```
api_version
args
as_dict ()
    Return row as dict.

asctime
created
exc_info
exc_text
filename
funcname
id
levelname
levelno
lineno
message
module
msecs
name
package_version
pathname
process
processname
produkt_id
relativecreated
```

```

stack_info
sub_sub_system_id
sub_system_id
summary
system_id
thread
threadname
user_spec_1
user_spec_2

```

2.6 herald.ps_bridge module

`ps.herald.ps_bridge.client_connected_handler` (*client_reader, client_writer*)

Start a new `asyncio.Task` to handle this specific client connection

`ps.herald.ps_bridge.handle_client` (*client_reader, client_writer*)

Runs for each client connected.

The first 4 bytes of the message define the total length of the message. Hence in the first step, the first four bytes are read and then the rest of the message. This “rest” of the message is depickled and stored in the database. If the “bridge-mode” is enabled, the “byte-buffer” (first 4 bytes plus the rest of the message is also sent to another socket - the bridge socket.

`ps.herald.ps_bridge.print_to_tunnel` (*data_p*)

Running in bridge-mode we route the incoming message to the outgoing socket.

2.7 herald.ps_herald module

2.8 herald.ps_neelix module

`ps.herald.ps_neelix.check` ()

[summary]

Go through the systemid’s found in the database:

- check if somebody needs notification
- update the heartbeat of the systemid

`ps.herald.ps_neelix.notify` (*system_id_p, starting_at_p, till_p*)

`ps.herald.ps_neelix.react` (*system_id_p, cause_p, value_p, row_p*)

- *system_id_p* Name of the system for which we react
- *cause_p* Name of the cause e.g. PATTERN or AGE
- *value_p* Value of the cause

- row_p table-row of the logging table correlated to the event

2.9 Subpackages

2.9.1 herald.graph_ql package

Submodules

herald.graph_ql.mutation module

```
class ps.herald.graph_ql.mutation.AddLog(*args, **kwargs)
    Bases: graphene.types.mutation.Mutation

    class Arguments
        Bases: object

        input = <ps.herald.graph_ql.typedefs.AddGLogFields object>
        log = <graphene.types.field.Field object>
        static mutate(self, info, input)

class ps.herald.graph_ql.mutation.Mutation(*args, **kwargs)
    Bases: graphene.types.objecttype.ObjectType

    addLog = <graphene.types.field.Field object>
```

herald.graph_ql.query module

```
class ps.herald.graph_ql.query.Query(*args, **kwargs)
    Bases: graphene.types.objecttype.ObjectType

    get_by_system_id = <graphene.types.structures.List object>
    get_logs = <graphene.types.structures.List object>
    get_search_options = <graphene.types.scalars.String object>
    goodbye = <graphene.types.scalars.String object>
    hello = <graphene.types.scalars.String object>
    static resolve_get_by_system_id(parent, info, **args)
    static resolve_get_logs(parent, info, search=None, first=None, skip=None, **args)
    static resolve_get_search_options(parent, info, **args)
    resolve_goodbye(info)
    resolve_hello(info, name)
```

herald.graph_ql.schema module

herald.graph_ql.typedefs module

```

class ps.herald.graph_ql.typedefs.AddGLogFields(*args, **kwargs)
    Bases: graphene.types.inputobjecttype.InputObjectType, ps.herald.graph_ql.
           typedefs.GLogFields

class ps.herald.graph_ql.typedefs.GHearBeatFields
    Bases: object

    id = <graphene.types.scalars.Int object>

    newest_heartbeat = <graphene.types.scalars.String object>

    system_id = <graphene.types.scalars.String object>

class ps.herald.graph_ql.typedefs.GHeartBeat(*args, **kwargs)
    Bases: graphene_sqlalchemy.types.SQLAlchemyObjectType

    connection = None

class ps.herald.graph_ql.typedefs.GLog(*args, **kwargs)
    Bases: graphene_sqlalchemy.types.SQLAlchemyObjectType

    connection = None

class ps.herald.graph_ql.typedefs.GLogFields
    Bases: object

    api_version = <graphene.types.scalars.String object>

    args = <graphene.types.scalars.String object>

    asctime = <graphene.types.scalars.String object>

    created = <graphene.types.scalars.String object>

    exc_info = <graphene.types.scalars.String object>

    exc_text = <graphene.types.scalars.String object>

    filename = <graphene.types.scalars.String object>

    funcname = <graphene.types.scalars.String object>

    id = <graphene.types.scalars.Int object>

    levelname = <graphene.types.scalars.String object>

    levelno = <graphene.types.scalars.String object>

    lineno = <graphene.types.scalars.String object>

    message = <graphene.types.scalars.String object>

    module = <graphene.types.scalars.String object>

    msecs = <graphene.types.scalars.String object>

    name = <graphene.types.scalars.String object>

    package_version = <graphene.types.scalars.String object>

    pathname = <graphene.types.scalars.String object>

    process = <graphene.types.scalars.String object>

```

```
processname = <graphene.types.scalars.String object>
produkt_id = <graphene.types.scalars.String object>
relativecreated = <graphene.types.scalars.String object>
stack_info = <graphene.types.scalars.String object>
sub_sub_system_id = <graphene.types.scalars.String object>
sub_system_id = <graphene.types.scalars.String object>
summary = <graphene.types.scalars.String object>
system_id = <graphene.types.scalars.String object>
thread = <graphene.types.scalars.String object>
threadname = <graphene.types.scalars.String object>
user_spec_1 = <graphene.types.scalars.String object>
user_spec_2 = <graphene.types.scalars.String object>
```

Module contents

2.10 Module contents

`ps.herald.create_app` (*name*, *have_config_file=False*, *test_config=None*)

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

- `ps.herald`, [12](#)
- `ps.herald.angular_api`, [7](#)
- `ps.herald.bare_html_api`, [7](#)
- `ps.herald.database`, [7](#)
- `ps.herald.graph_ql`, [12](#)
- `ps.herald.graph_ql.mutation`, [10](#)
- `ps.herald.graph_ql.query`, [10](#)
- `ps.herald.graph_ql.schema`, [11](#)
- `ps.herald.graph_ql.typedefs`, [11](#)
- `ps.herald.model`, [8](#)
- `ps.herald.ps_bridge`, [9](#)
- `ps.herald.ps_herald`, [9](#)
- `ps.herald.ps_neelix`, [9](#)

A

AddGLogFields (class *ps.herald.graph_ql.typedefs*), 11
 AddLog (class in *ps.herald.graph_ql.mutation*), 10
 addLog (*ps.herald.graph_ql.mutation.Mutation* attribute), 10
 AddLog.Arguments (class in *ps.herald.graph_ql.mutation*), 10
 api_version (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
 api_version (*ps.herald.model.Log* attribute), 8
 args (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
 args (*ps.herald.model.Log* attribute), 8
 as_dict() (*ps.herald.model.Log* method), 8
 asctime (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
 asctime (*ps.herald.model.Log* attribute), 8

C

check() (in module *ps.herald.ps_neelix*), 9
 client_connected_handler() (in module *ps.herald.ps_bridge*), 9
 close_engine() (in module *ps.herald.database*), 7
 connection (*ps.herald.graph_ql.typedefs.GHeartBeat* attribute), 11
 connection (*ps.herald.graph_ql.typedefs.GLog* attribute), 11
 create_app() (in module *ps.herald*), 12
 created (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
 created (*ps.herald.model.Log* attribute), 8

E

exc_info (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
 exc_info (*ps.herald.model.Log* attribute), 8
 exc_text (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
 exc_text (*ps.herald.model.Log* attribute), 8

F

in filename (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
 filename (*ps.herald.model.Log* attribute), 8
 funcname (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
 in funcname (*ps.herald.model.Log* attribute), 8

G

get_by_system_id (*ps.herald.graph_ql.query.Query* attribute), 10
 get_engine() (in module *ps.herald.database*), 7
 get_logs (*ps.herald.graph_ql.query.Query* attribute), 10
 get_search_options (*ps.herald.graph_ql.query.Query* attribute), 10
 get_session() (in module *ps.herald.database*), 7
 GHearBeatFields (class in *ps.herald.graph_ql.typedefs*), 11
 GHeartBeat (class in *ps.herald.graph_ql.typedefs*), 11
 GLog (class in *ps.herald.graph_ql.typedefs*), 11
 GLogFields (class in *ps.herald.graph_ql.typedefs*), 11
 goodbye (*ps.herald.graph_ql.query.Query* attribute), 10

H

handle_client() (in module *ps.herald.ps_bridge*), 9
 HeartBeat (class in *ps.herald.model*), 8
 hello (*ps.herald.graph_ql.query.Query* attribute), 10
 hello() (in module *ps.herald.angular_api*), 7
 hello() (in module *ps.herald.bare_html_api*), 7
 hello_to_ps_basic_logger() (in module *ps.herald.angular_api*), 7
 hello_to_ps_basic_logger() (in module *ps.herald.bare_html_api*), 7

I

id (*ps.herald.graph_ql.typedefs.GHeartBeatFields* attribute), 11
 id (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
 id (*ps.herald.model.HeartBeat* attribute), 8
 id (*ps.herald.model.Log* attribute), 8

[index\(\)](#) (in module *ps.herald.angular_api*), 7
[index\(\)](#) (in module *ps.herald.bare_html_api*), 7
[init_app\(\)](#) (in module *ps.herald.database*), 7
[init_db\(\)](#) (in module *ps.herald.database*), 7
[input](#) (*ps.herald.graph_ql.mutation.AddLog.Arguments* attribute), 10
[internal_error\(\)](#) (in module *ps.herald.angular_api*), 7
[isset\(\)](#) (in module *ps.herald.bare_html_api*), 7

L

[levelname](#) (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
[levelname](#) (*ps.herald.model.Log* attribute), 8
[levelno](#) (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
[levelno](#) (*ps.herald.model.Log* attribute), 8
[lineno](#) (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
[lineno](#) (*ps.herald.model.Log* attribute), 8
[list\(\)](#) (in module *ps.herald.angular_api*), 7
[Log](#) (class in *ps.herald.model*), 8
[log](#) (*ps.herald.graph_ql.mutation.AddLog* attribute), 10

M

[message](#) (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
[message](#) (*ps.herald.model.Log* attribute), 8
[module](#)
 ps.herald, 12
 ps.herald.angular_api, 7
 ps.herald.bare_html_api, 7
 ps.herald.database, 7
 ps.herald.graph_ql, 12
 ps.herald.graph_ql.mutation, 10
 ps.herald.graph_ql.query, 10
 ps.herald.graph_ql.schema, 11
 ps.herald.graph_ql.typedefs, 11
 ps.herald.model, 8
 ps.herald.ps_bridge, 9
 ps.herald.ps_herald, 9
 ps.herald.ps_neelix, 9
[module](#) (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
[module](#) (*ps.herald.model.Log* attribute), 8
[msecs](#) (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
[msecs](#) (*ps.herald.model.Log* attribute), 8
[mutate\(\)](#) (*ps.herald.graph_ql.mutation.AddLog* static method), 10
[Mutation](#) (class in *ps.herald.graph_ql.mutation*), 10

N

[name](#) (*ps.herald.graph_ql.typedefs.GLogFields* at-

tribute), 11
[name](#) (*ps.herald.model.Log* attribute), 8
[newest_heartbeat](#) (*ps.herald.graph_ql.typedefs.GHearBeatFields* attribute), 11
[newest_heartbeat](#) (*ps.herald.model.HeartBeat* attribute), 8
[notify\(\)](#) (in module *ps.herald.ps_neelix*), 9

O

[options\(\)](#) (in module *ps.herald.angular_api*), 7

P

[package_version](#) (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
[package_version](#) (*ps.herald.model.Log* attribute), 8
[pathname](#) (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
[pathname](#) (*ps.herald.model.Log* attribute), 8
[print_to_tunnel\(\)](#) (in module *ps.herald.ps_bridge*), 9
[process](#) (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
[process](#) (*ps.herald.model.Log* attribute), 8
[processname](#) (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 11
[processname](#) (*ps.herald.model.Log* attribute), 8
[produkt_id](#) (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 12
[produkt_id](#) (*ps.herald.model.Log* attribute), 8
[ps.herald](#)
 module, 12
 ps.herald.angular_api
 module, 7
 ps.herald.bare_html_api
 module, 7
 ps.herald.database
 module, 7
 ps.herald.graph_ql
 module, 12
 ps.herald.graph_ql.mutation
 module, 10
 ps.herald.graph_ql.query
 module, 10
 ps.herald.graph_ql.schema
 module, 11
 ps.herald.graph_ql.typedefs
 module, 11
 ps.herald.model
 module, 8
 ps.herald.ps_bridge
 module, 9
 ps.herald.ps_herald
 module, 9
 ps.herald.ps_neelix

module, 9

Q

Query (class in *ps.herald.graph_ql.query*), 10

R

react () (in module *ps.herald.ps_neelix*), 9

relativecreated (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 12

relativecreated (*ps.herald.model.Log* attribute), 8

resolve_get_by_system_id ()
(*ps.herald.graph_ql.query.Query* static method), 10

resolve_get_logs ()
(*ps.herald.graph_ql.query.Query* static method), 10

resolve_get_search_options ()
(*ps.herald.graph_ql.query.Query* static method), 10

resolve_goodbye ()
(*ps.herald.graph_ql.query.Query* method), 10

resolve_hello () (*ps.herald.graph_ql.query.Query* method), 10

rm_logs () (in module *ps.herald.bare_html_api*), 7

S

set_search_params () (in module *ps.herald.bare_html_api*), 7

stack_info (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 12

stack_info (*ps.herald.model.Log* attribute), 8

sub_sub_system_id
(*ps.herald.graph_ql.typedefs.GLogFields* attribute), 12

sub_sub_system_id (*ps.herald.model.Log* attribute), 9

sub_system_id (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 12

sub_system_id (*ps.herald.model.Log* attribute), 9

summary (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 12

summary (*ps.herald.model.Log* attribute), 9

system_id (*ps.herald.graph_ql.typedefs.GHearBeatFields* attribute), 11

system_id (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 12

system_id (*ps.herald.model.HeartBeat* attribute), 8

system_id (*ps.herald.model.Log* attribute), 9

T

thread (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 12

thread (*ps.herald.model.Log* attribute), 9

threadname (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 12

threadname (*ps.herald.model.Log* attribute), 9

U

user_spec_1 (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 12

user_spec_1 (*ps.herald.model.Log* attribute), 9

user_spec_2 (*ps.herald.graph_ql.typedefs.GLogFields* attribute), 12

user_spec_2 (*ps.herald.model.Log* attribute), 9