securiphant

Release 0.1.0

Hermann Krumrey

Feb 03, 2020

CONTENTS

Contents:

CHAPTER

ONE

SECURIPHANT PACKAGE

1.1 Subpackages

1.1.1 securiphant.alert_bot package

Submodules

securiphant.alert_bot.AlertBot module

securiphant.alert_bot.AlertBotParser module

class securiphant.alert_bot.AlertBotParser.**AlertBotParser**
Bases: kudubot.parsing.CommandParser.CommandParser

The Parser for the Alert Bot

classmethod commands() → List[kudubot.parsing.Command.Command]

Returns The enabled commands for this bot

classmethod name() → str

Returns The name of the parser

Module contents

1.1.2 securiphant.db package

Subpackages

securiphant.db.events package

Submodules

securiphant.db.events.DoorOpenEvent module

class securiphant.db.events.DoorOpenEvent.**DoorOpenEvent**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

Event that gets stored whenever the door is opened

__init__(***kwargs*)
A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

duration
The duration that the door was open

id
The ID of the event

timestamp
The time this event took place as a UNIX timestamp

was_authorized
Whether or not the user was authorized while the door was opened

securiphant.db.events.SpeakerEvent module

class securiphant.db.events.SpeakerEvent.**SpeakerEvent**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

Event that gets stored whenever a phrase is said on the speaker system

__init__(***kwargs*)
A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

executed
Whether or not the speaker event was executed already

id
The ID of the event

play()
Plays the text over the speaker using flite :return: None

text
The text said over the speakers

timestamp
The time this event took place as a UNIX timestamp

Module contents

securiphant.db.states package

Submodules

securiphant.db.states.BooleanState module

class securiphant.db.states.BooleanState.**BooleanState**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

SQLAlchemy table that stores boolean state values

__init__(***kwargs*)
A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

key
The key for the boolean value

value
The boolean value itself

securiphant.db.states.CheckIn module

class securiphant.db.states.CheckIn.**CheckIn**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

Class that defines a database table that tracks if services are running by making them check at set intervals.

__init__(***kwargs*)
A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

is_alive() → bool

Returns Whether or not the service has checked in recently

last_ping
The last time the service checked in, as a unix timestamp

service
The name of the service

securiphant.db.states.IntState module

class securiphant.db.states.IntState.**IntState**(***kwargs*)
Bases: sqlalchemy.ext.declarative.api.Base

SQLAlchemy table that stores integer state values

__init__(***kwargs*)
A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

key
The key for the integer value

value
The int value itself

Module contents

Module contents

class securiphant.db.**Base**(***kwargs*)
Bases: object

The most base type

__init__(***kwargs*)
A simple constructor that allows initialization from kwargs.

Sets attributes on the constructed instance using the names and values in `kwargs`.

Only keys that are present as attributes of the instance's class are allowed. These could be, for example, any mapped columns or relationships.

metadata = MetaData(bind=None)

securiphant.db.**generate_mysql_uri**() → str
Generates a mysql URI for the stored confiuration :return: The generated URI

1.1.3 securiphant.qt package

Subpackages

securiphant.qt.generated package

Submodules

securiphant.qt.generated.main module

Module contents

Submodules

securiphant.qt.MainWindow module

Module contents

1.1.4 securiphant.utils package

Submodules

securiphant.utils.camera module

securiphant.utils.config module

securiphant.utils.config.**config_dir = '/root/.config/securiphant'**
The directory containing all config and data files for securiphant

securiphant.utils.config.**load_config**() → Dict[str, Any]
Loads the config for securiphant :return: The configuration data

securiphant.utils.config.**write_config**(*data: Dict[str, Any]*)
Writes data to the config file :param data: The data to write :return: None

securiphant.utils.db module

securiphant.utils.db.**get_boolean_state**(*key: str*, *session: sqlalchemy.orm.session.Session*) → securiphant.db.states.BooleanState.BooleanState
Retrieves a boolean state from the database :param key: The key of the state entry :param session: The session to use for querying :return: The BooleanState object

securiphant.utils.db.**get_int_state**(*key: str*, *session: sqlalchemy.orm.session.Session*) → securiphant.db.states.IntState.IntState
Retrieves an int state from the database :param key: The key of the state entry :param session: The session to use for querying :return: The IntState object

securiphant.utils.db.**initialize_database**()
Initializes the database, creating all tables and filling default key-value pairs. :return: The database session

securiphant.utils.door module

securiphant.utils.environment module

securiphant.utils.environment.**environment_check_loop**()
Checks the DFT22 sensor for temperature and humidity values and stores the result in the database :return: None

securiphant.utils.environment.**environment_lock = <unlocked _thread.lock object>**
Lock that makes sure no two threads ever access the environment sensor at the same time.

securiphant.utils.environment.**get_environment_data**() → Tuple[Optional[int], Optional[int]]
Reads the temperature and humidity from an Adafruit DHT22 sensor :return: The temperature, the humidity in integer values

securiphant.utils.init module

securiphant.utils.init.**initialize**(*configurations: List[str]*)
Initializes the securiphant installation :param configurations: The securiphant configurations run by this device :return: None

securiphant.utils.nfc module

securiphant.utils.qt module

securiphant.utils.speech module

securiphant.utils.speech.**queue_speaker_event**(*db_session: sqlalchemy.orm.session.Session*, *text: str*)
Queues a new speaker event in the database :param db_session: The database session to use :param text: The text to speak :return: None

securiphant.utils.speech.**speaker_loop**()
Continuously checks for new speaker events and executes any ones that were not yet executed. :return: None

securiphant.utils.systemd module

securiphant.utils.systemd.**is_active**(*service: str*) → bool
Checks whether or not a systemd securiphant service is active or not :param service: The service to check :return: True if active, else False

securiphant.utils.systemd.**reload_daemon**()
Reloads the systemd daemon :return: None

securiphant.utils.systemd.**securiphant_services = {'display': ['display'], 'door': ['door-**
A dictionary mapping securiphant configurations to systemd services

securiphant.utils.systemd.**start_service**(*service: str*)
Starts a service if it's not already running :param service: The service to start :return: None

securiphant.utils.systemd.**stop_service**(*service: str*)
Stops a service if it's running :param service: The service to stop :return: None

securiphant.utils.systemd.**systemctl_call**(*service: str*, *mode: str*) → int
Calls a systemctl command for a service :param service: The service for which to call the command :param mode: The command mode (ex: start or stop) :return: The status code of the call

securiphant.utils.systemd.**systemd_dir = '/root/.config/systemd/user'**
Thje directory containing systemd service files

securiphant.utils.weather module

securiphant.utils.weather.**get_weather**() → Optional[Dict[str, str]]
Gets the weather data for a specified city :return: The weather data, including the following:

- temperature
- humidity
- weather type
- weather icon

Module contents

1.2 Module contents

securiphant.**sentry_dsn = 'https://a42371c14e444bfd80ece22ab721cf6d@sentry.namibsun.net/14'**
The sentry DSN used for logging exceptions

securiphant.**version = '0.1.0'**
The current version of securiphant

CHAPTER

TWO

SECURIPHANT

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

Symbols

A

B

C

D

E

G

I

K

L

M

N

P

Q

R

S

T

V

W