Vent Documentation

Release 0.4.1

Cyber Reboot

Jul 19, 2017

Api Functions

Vent is a library that includes a CLI designed to serve as a general platform for analyzing network traffic. Built with some basic functionality, Vent serves as a user-friendly platform to build custom `plugins` on to perform user-defined processing on incoming network data. Vent supports any filetype, but only processes filetypes based on the types of plugins installed for that instance of vent.

Simply create your `plugins`, point Vent to them, install them, and drop a file in Vent to begin processing!

CHAPTER 1

Dependencies

```
docker >= 1.13.1
make (if building from source)
pip
python2.7.x
```

CHAPTER 2

Getting Set Up

There's two ways to get Vent up and running on your machine:

1. Pip:

```
$ pip install vent
```

2. Building from source (make is required):

```
$ git clone --recursive https://github.com/CyberReboot/vent.git
$ cd vent
$ make # (sudo may be required to install the vent command in the system bin path)
```

Note: If you already have `docker-py` installed on your machine, you may need to `pip uninstall docker-py` first. `vent` will install `docker-py` as part of the installation process. However, there are known incompatibilities of `docker-py` with older versions.

Once installed, it's simply:

```
$ vent
```

CHAPTER 3

Contributing to Vent

Want to contribute? Awesome! Issue a pull request or see more details here.

See this for a crash course on npyscreen: the GUI used by Vent!

vent.api package

Submodules

vent.api.actions module

class vent.api.actions.**Action**(***kargs*)

Handle actions in menu

add(*repo*, *tools=None*, *overrides=None*, *version='HEAD'*, *branch='master'*, *build=True*, *user=None*, *pw=None*, *groups=None*, *version_alias=None*, *wild=None*, *remove_old=True*, *disable_old=True*)

Add a new set of tool(s)

add_image(*image*, *link_name*, *tag=None*, *registry=None*, *groups=None*)

Add a new image from a Docker registry

static backup()

build(*repo=None*, *name=None*, *groups=None*, *enabled='yes'*, *branch='master'*, *version='HEAD'*)

Build a set of tools that match the parameters given

clean(*repo=None*, *name=None*, *groups=None*, *enabled='yes'*, *branch='master'*, *version='HEAD'*)

Clean (stop and remove) a set of tools that match the parameters given, if no parameters are given, clean all installed tools on the master branch at verison HEAD that are enabled

static configure()

get_configure(*repo=None*, *name=None*, *groups=None*, *enabled='yes'*, *branch='master'*, *version='HEAD'*)

Get the vent.template settings for a given tool by looking at the plugin_manifest

static help()

inventory(*choices=None*)
Return a dictionary of the inventory items and status

logs(*c_type=None*, *grep_list=None*)
Generically filter logs stored in log containers

prep_start(*repo=None*, *name=None*, *groups=None*, *enabled='yes'*, *branch='master'*, *version='HEAD'*)
Prep a bunch of containers to be started to they can be ordered

remove(*repo=None*, *namespace=None*, *name=None*, *groups=None*, *enabled='yes'*, *branch='master'*, *version='HEAD'*, *built='yes'*)
Remove tools or a repo

reset()
Factory reset all of Vent's user data, containers, and images

static restore()

save_configure(*repo=None*, *name=None*, *groups=None*, *enabled='yes'*, *branch='master'*, *version='HEAD'*, *config_val=''*, *from_registry=False*)
Save changes made to vent.template through npyscreen to the template and to plugin_manifest

start(*tool_d*)
Start a set of tools that match the parameters given, if no parameters are given, start all installed tools on the master branch at verison HEAD that are enabled

stop(*repo=None*, *name=None*, *groups=None*, *enabled='yes'*, *branch='master'*, *version='HEAD'*)
Stop a set of tools that match the parameters given, if no parameters are given, stop all installed tools on the master branch at verison HEAD that are enabled

update(*repo=None*, *name=None*, *groups=None*, *enabled='yes'*, *branch='master'*, *version='HEAD'*)
Update a set of tools that match the parameters given, if no parameters are given, updated all installed tools on the master branch at verison HEAD that are enabled

static upgrade()

vent.api.menu_helpers module

class vent.api.menu_helpers.**MenuHelper**(***kargs*)
Handle helper functions in the API for the Menu

cores(*action*, *branch='master'*, *version='HEAD'*)
Supply action (install, build, start, stop, clean) for core tools

repo_branches(*repo*)
Get the branches of a repository

repo_commits(*repo*)
Get the commit IDs for all of the branches of a repository

repo_tools(*repo*, *branch*, *version*)
Get available tools for a repository branch at a version

tools_status(*core*, *branch='master'*, *version='HEAD'*, ***kargs*)
Get tools that are currently installed/built/running and also the number of repos that those tools come from; can toggle whether looking for core tools or plugin tools

vent.api.plugin_helpers module

class vent.api.plugin_helpers.**PluginHelper**(***kargs*)
Handle helper functions for the Plugin class

apply_path(*repo*)
Set path to where the repo is and return original path

available_tools(*path*, *version='HEAD'*, *groups=None*)
Return list of possible tools in repo for the given version and branch

checkout(*branch='master'*, *version='HEAD'*)
Checkout a specific version and branch of a repo

clone(*repo*, *user=None*, *pw=None*)
Clone the repository

constraint_options(*constraint_dict*, *options*)
Return result of constraints and options against a template

get_path(*repo*, *core=False*)
Return the path for the repo

prep_start(*repo=None*, *name=None*, *groups=None*, *enabled='yes'*, *branch='master'*, *version='HEAD'*)
Start a set of tools that match the parameters given, if no parameters are given, start all installed tools on the master branch at verison HEAD that are enabled

start_containers(*container*, *tool_d*, *s_containers*, *f_containers*)
Start container that was passed in and return status

start_priority_containers(*groups*, *group_orders*, *tool_d*)
Select containers based on priorities to start

start_remaining_containers(*containers_remaining*, *tool_d*)
Select remaining containers that didn't have priorities to start

start_sections(*s*, *files*, *groups*, *enabled*, *branch*, *version*)
Run through sections for prep_start

static tool_matches(*tools=None*, *version='HEAD'*)
Get the tools paths and versions that were specified

vent.api.plugins module

class vent.api.plugins.**Plugin**(***kargs*)
Handle Plugins

add(*repo*, *tools=None*, *overrides=None*, *version='HEAD'*, *branch='master'*, *build=True*, *user=None*, *pw=None*, *groups=None*, *version_alias=None*, *wild=None*, *remove_old=True*, *disable_old=True*, *limit_groups=None*, *core=False*)
Adds a plugin of tool(s) tools is a list of tuples, where the pair is a tool name (path to Dockerfile) and version

tools are for explicitly limiting which tools and versions (if version in tuple is '', then defaults to version)

overrides is a list of tuples, where the pair is a tool name (path to Dockerfile) and a version

overrides are for explicitly removing tools and overriding versions of tools (if version in tuple is '', then tool is removed, otherwise that tool is checked out at the specific version in the tuple)

if tools and overrides are left as empty lists, then all tools in the repo are pulled down at the version and branch specified or defaulted to

version is globally set for all tools, unless overridden in tools or overrides

branch is globally set for all tools build is a boolean of whether or not to build the tools now user is the username for a private repo if needed pw is the password to go along with the username for a private repo groups is globally set for all tools version_alias is globally set for all tools and is a mapping from a

friendly version tag to the real version commit ID

wild lets you specify individual overrides for additional values in the tuple of tools or overrides. wild is a list containing one or more of the following: branch, build, groups, version_alias the order of the items in the wild list will expect values to be tacked on in the same order to the tuple for tools and overrides in additional to the tool name and version

remove_old lets you specify whether or not to remove previously found tools that match to ones being added currently (note does not stop currently running instances of the older version)

disable_old lets you specify whether or not to disable previously found tools that match to ones being added currently (note does not stop currently running instances of the older version)

limit_groups is a list of groups to build tools for that match group names in vent.template of each tool if exists

Examples:

repo=fe (get all tools from repo 'fe' at version 'HEAD' on branch 'master')

repo=foo, version="3d1f", branch="foo" (get all tools from repo 'foo' at verion '3d1f' on branch 'foo')

repo=foo, tools=[('bar', ''), ('baz', '1d32')] (get only 'bar' from repo 'foo' at version 'HEAD' on branch 'master' and 'baz' from repo 'foo' at version '1d32' on branch 'master', ignore all other tools in repo 'foo')

repo=foo overrides=[('baz/bar', ''), ('.', '1c4e')], version='4fad' (get all tools from repo 'foo' at verion '4fad' on branch 'master' except 'baz/bar' and for tool '.' get version '1c4e')

repo=foo tools=[('bar', '1a2d')], overrides=[('baz', 'f2a1')] (not a particularly useful example, but get 'bar' from 'foo' at version '1a2d' and get 'baz' from 'foo' at version 'f2a1' on branch 'master', ignore all other tools)

add_image(**args*, ***kwargs*)

builder(**args*, ***kwargs*)

current_version(*name*, *namespace=None*, *branch='master'*)
Return current version for a given tool

disable(*name*, *namespace=None*, *branch='master'*, *version='HEAD'*)
Disable tool at a specific version, default to head

enable(*name*, *namespace=None*, *branch='master'*, *version='HEAD'*)
Enable tool at a specific version, default to head

list_tools()
Return list of tuples of all tools

remove(*name=None*, *repo=None*, *namespace=None*, *branch='master'*, *groups=None*, *enabled='yes'*, *version='HEAD'*, *built='yes'*)
Remove tool (name) or repository, repository is the url. If no arguments are specified, all tools will be removed for the defaults.

state (*name*, *namespace=None*, *branch='master'*)
Return state of a tool, disabled/enabled for each version

update (*name=None*, *repo=None*, *namespace=None*, *branch=None*, *groups=None*)
Update tool (name) or repository, repository is the url. If no arguments are specified, all tools will be updated

versions (*name*, *namespace=None*, *branch='master'*)
Return available versions of a tool

vent.api.templates module

class vent.api.templates.**Template** (*template=None*)
Handle parsing templates

add_option (**args*, ***kwargs*)

add_section (**args*, ***kwargs*)

constrained_sections (**args*, ***kwargs*)

del_option (**args*, ***kwargs*)

del_section (**args*, ***kwargs*)

option (**args*, ***kwargs*)

options (**args*, ***kwargs*)

section (**args*, ***kwargs*)

sections (**args*, ***kwargs*)

set_option (**args*, ***kwargs*)

write_config (**args*, ***kwargs*)

Module contents

vent.core package

Subpackages

vent.core.file_drop package

Submodules

vent.core.file_drop.file_drop module

Module contents

vent.core.network_tap package

Subpackages

vent.core.network_tap.ncontrol package

Subpackages

vent.core.network_tap.ncontrol.rest package

Submodules

vent.core.network_tap.ncontrol.rest.create module

vent.core.network_tap.ncontrol.rest.filters module

vent.core.network_tap.ncontrol.rest.start module

vent.core.network_tap.ncontrol.rest.stop module

Module contents

Submodules

vent.core.network_tap.ncontrol.ncontrol module

Module contents

Module contents

vent.core.rmq_es_connector package

Submodules

vent.core.rmq_es_connector.rmq_es_connector module

Module contents

vent.core.rq_worker package

Submodules

vent.core.rq_worker.file_watch module

Module contents

Module contents

vent.helpers package

Submodules

vent.helpers.errors module

vent.helpers.errors.**ErrorHandler**(*function*)

vent.helpers.logs module

vent.helpers.logs.**Logger**(*name*, ***kargs*)
Create and return logger

vent.helpers.meta module

vent.helpers.meta.**Containers**(*vent=True*, *running=True*)
Get containers that are created, by default limit to vent containers that are running

vent.helpers.meta.**Cpu**()
Get number of available CPUs

vent.helpers.meta.**Docker**()
Get Docker setup information

vent.helpers.meta.**Gpu**(*pull=False*)
Check for support of GPUs, and return what's available

vent.helpers.meta.**GpuUsage**()
Get the current GPU usage of available GPUs

vent.helpers.meta.**Images**(*vent=True*)
Get images that are build, by default limit to vent images

vent.helpers.meta.**Jobs**()
Get the number of jobs that are running and finished, and the number of total tools running and finished for those jobs

vent.helpers.meta.**Services**(*core*, *vent=True*, ***kargs*)
Get services that have exposed ports, expects param core to be True or False based on which type of services to return, by default limit to vent containers, if not limited by vent containers, then core is ignored.

vent.helpers.meta.**System**()
Get system operating system

vent.helpers.meta.**Timestamp**()
Get the current datetime in UTC

vent.helpers.meta.**Tools**(***kargs*)
Get tools that exist in the manifest

vent.helpers.meta.**Uptime**()
Get the current uptime information

vent.helpers.meta.**Version**()
Get Vent version

vent.helpers.paths module

class vent.helpers.paths.**PathDirs**(*base_dir='/home/docs/.vent/'*, *plugins_dir='plugins/'*, *meta_dir='/home/docs/.vent'*)
Global path directories for vent

static ensure_dir(*path*)
Tries to create directory, if fails, checks if path already exists

static ensure_file(*path*)
Checks if file exists, if fails, tries to create file

host_config()
Ensure the host configuration file exists

Module contents

vent.menus package

Submodules

vent.menus.add module

class vent.menus.add.**AddForm**(*name=None*, *parentApp=None*, *framed=None*, *help=None*, *color='FORMDEFAULT'*, *widget_list=None*, *cycle_widgets=False*, **args*, ***keywords*)
Bases: npyscreen.fmActionForm.ActionForm

For for adding a new repo

create()
Create widgets for AddForm

default_repo = 'https://github.com/cyberreboot/vent-plugins'

on_cancel()
When user clicks cancel, will return to MAIN

on_ok()
Add the repository

quit(**args*, ***kwargs*)
Overridden to switch back to MAIN form

vent.menus.add_options module

class vent.menus.add_options.**AddOptionsForm**(*name=None*, *parentApp=None*, *framed=None*, *help=None*, *color='FORMDEFAULT'*, *widget_list=None*, *cycle_widgets=False*, **args*, ***keywords*)
Bases: npyscreen.fmActionForm.ActionForm

For specifying options when adding a repo

branch_cb = {}

branches = []

build_tc = {}

commit_tc = {}

commits = {}

create()
Update with current branches and commits

error = None

on_cancel()

on_ok()
Take the branch, commit, and build selection and add them as plugins

quit(**args*, ***kwargs*)

repo_values()
Set the appropriate repo dir and get the branches and commits of it

vent.menus.choose_tools module

class vent.menus.choose_tools.**ChooseToolsForm**(*name=None*, *parentApp=None*, *framed=None*, *help=None*, *color='FORMDEFAULT'*, *widget_list=None*, *cycle_widgets=False*, **args*, ***keywords*)
Bases: npyscreen.fmActionForm.ActionForm

For picking which tools to add

create()
Update with current tools for each branch at the version chosen

on_cancel()

on_ok()
Take the tool selections and add them as plugins

quit(**args*, ***kwargs*)

repo_tools(*branch*)
Set the appropriate repo dir and get the tools available of it

tools_tc = {}

vent.menus.editor module

class vent.menus.editor.**EditorForm**(**args*, ***keywords*)
Bases: npyscreen.fmActionForm.ActionForm

Form that can be used as a pseudo test editor in npyscreen

change_screens()
Change to the next tool to edit or back to MAIN form

create()
Create multi-line widget for editing

on_cancel()
Don't save changes made to vent.template

on_ok()
Save changes made to vent.template

vent.menus.help module

class vent.menus.help.**HelpForm**(**args*, ***keywords*)
Bases: npyscreen.fmFormWithMenus.ActionFormWithMenus

Help form for the Vent CLI

change_forms(**args*, ***keywords*)
Checks which form is currently displayed and toggles to the other one

create()
Override method for creating FormBaseNew form

exit(**args*, ***keywords*)

on_cancel()

on_ok()

static switch(*page*)

vent.menus.inventory module

class vent.menus.inventory.**InventoryForm**(*action=None*, *logger=None*, **args*, ***keywords*)
Bases: npyscreen.fmForm.FormBaseNew

Inventory form for the Vent CLI

create()
Override method for creating FormBaseNew form

quit(**args*, ***kwargs*)
Overridden to switch back to MAIN form

vent.menus.inventory_forms module

class vent.menus.inventory_forms.**BaseInventoryForm**(*action_dict=None*, *action_name=None*, **args*, ***keywords*)
Bases: *vent.menus.inventory.InventoryForm*

Base form to inherit from

class vent.menus.inventory_forms.**InventoryCoreToolsForm**(**args*, ***keywords*)
Bases: *vent.menus.inventory_forms.BaseInventoryForm*

Inventory Core Tools form for the Vent CLI

class vent.menus.inventory_forms.**InventoryToolsForm**(**args*, ***keywords*)
Bases: *vent.menus.inventory_forms.BaseInventoryForm*

Inventory Tools form for the Vent CLI

vent.menus.logs module

class vent.menus.logs.**LogsForm**(*name=None*, *parentApp=None*, *framed=None*, *help=None*, *color='FORMDEFAULT'*, *widget_list=None*, *cycle_widgets=False*, **args*, ***keywords*)
Bases: npyscreen.fmForm.FormBaseNew

Logs form for the Vent CLI

create()
Override method for creating FormBaseNew form

quit(**args*, ***kwargs*)
Overridden to switch back to MAIN form

vent.menus.main module

class vent.menus.main.**MainForm**(**args*, ***keywords*)
Bases: npyscreen.fmFormWithMenus.FormBaseNewWithMenus

Main information landing form for the Vent CLI

add_form(*form*, *form_name*, *form_args*)
Add new form and switch to it

static core_tools(*action*)
Perform actions for core tools

create()
Override method for creating FormBaseNewWithMenu form

static exit(**args*, ***kwargs*)

help_form(**args*, ***keywords*)
Toggles to help

perform_action(*action*)
Perform actions in the api from the CLI

remove_forms(*form_names*)
Remove all forms supplied

switch_tutorial(*action*)
Tutorial forms

system_commands(*action*)
Perform system commands

static t_status(*core*)
Get status of tools for either plugins or core

while_waiting()
Update fields periodically if nothing is happening

vent.menus.services module

class vent.menus.services.**ServicesForm**(**args*, ***keywords*)
Bases: npyscreen.fmForm.FormBaseNew

Services form for the Vent CLI

create()
Override method for creating FormBaseNew form

quit(**args*, ***kwargs*)
Overridden to switch back to MAIN form

vent.menus.tools module

class vent.menus.tools.**ToolForm**(**args*, ***keywords*)
Bases: npyscreen.fmActionForm.ActionForm

Tools form for the Vent CLI

create()
Update with current tools

on_cancel()
When user clicks cancel, will return to MAIN

on_ok()
Take the tool selections and perform the provided action on them

quit(**args*, ***kwargs*)
Overridden to switch back to MAIN form

vent.menus.tutorial_forms module

class vent.menus.tutorial_forms.**TutorialAddingFilesForm**(**args*, ***keywords*)
Bases: *vent.menus.tutorials.TutorialForm*

Tutorial Adding Files form for the Vent CLI

class vent.menus.tutorial_forms.**TutorialAddingPluginsForm**(**args*, ***keywords*)
Bases: *vent.menus.tutorials.TutorialForm*

Tutorial Adding Plugins form for the Vent CLI

class vent.menus.tutorial_forms.**TutorialBackgroundForm**(**args*, ***keywords*)
Bases: *vent.menus.tutorials.TutorialForm*

Tutorial Background form for the Vent CLI

class vent.menus.tutorial_forms.**TutorialBuildingCoresForm**(**args*, ***keywords*)
Bases: *vent.menus.tutorials.TutorialForm*

Tutorial Building Cores form for the Vent CLI

class vent.menus.tutorial_forms.**TutorialGettingSetupForm**(**args*, ***keywords*)
Bases: *vent.menus.tutorials.TutorialForm*

Tutorial Getting Setup form for the Vent CLI

class vent.menus.tutorial_forms.**TutorialIntroForm**(**args*, ***keywords*)
Bases: *vent.menus.tutorials.TutorialForm*

Tutorial introduction landing form for the Vent CLI

class vent.menus.tutorial_forms.**TutorialSettingUpServicesForm**(**args*, ***keywords*)
Bases: *vent.menus.tutorials.TutorialForm*

Tutorial Setting up Services form for the Vent CLI

class vent.menus.tutorial_forms.**TutorialStartingCoresForm**(**args*, ***keywords*)
Bases: *vent.menus.tutorials.TutorialForm*

Tutorial Starting Cores form for the Vent CLI

class vent.menus.tutorial_forms.**TutorialTerminologyForm**(**args*, ***keywords*)
Bases: *vent.menus.tutorials.TutorialForm*

Tutorial terminology form for the Vent CLI

vent.menus.tutorials module

class vent.menus.tutorials.**TutorialForm**(*title=''*, *text=''*, *next_tutorial=''*, **args*, ***keywords*)
Bases: npyscreen.fmFormWithMenus.ActionFormWithMenus

Tutorial form for the Vent CLI

create()
Overridden to add handlers and content

on_cancel()
When user clicks cancel, will return to MAIN

on_ok()
When user clicks ok, will proceed to next tutorial

quit(**args*, ***kwargs*)
Overridden to switch back to MAIN form

switch(*name*)
Wrapper that switches to provided form

Module contents

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

v

Index

G

H

I

J

L

M

O

P

Q

R

S

T

U

V

W