hephaestus

Generated by Doxygen 1.8.3.1

Mon May 12 2014 12:05:41

Contents

1 hephaestus 1

2 Namespace Index 3

2.1 Packages . . . 3

3 Hierarchical Index 5

3.1 Class Hierarchy . . . 5

4 Class Index 7

4.1 Class List . . . 7

5 File Index 9

5.1 File List . . . 9

6 Namespace Documentation 11

6.1 builddocs Namespace Reference . . . 11

6.2 demo Namespace Reference . . . 11

6.2.1 Variable Documentation . . . 11

6.2.1.1 child . . . 11

6.3 hephaestus Namespace Reference . . . 11

6.3.1 Function Documentation . . . 11

6.3.1.1 main . . . 11

6.4 ipaddr_test Namespace Reference . . . 11

6.4.1 Function Documentation . . . 12

6.4.1.1 main . . . 12

6.5 lib Namespace Reference . . . 12

6.6 lib.Baseconfig Namespace Reference . . . 12

6.7 lib.configparser Namespace Reference . . . 12

6.7.1 Function Documentation . . . 12

6.7.1.1 read_config . . . 12

6.7.1.2 write_config . . . 13

6.8 lib.hephaestus_utils Namespace Reference . . . 13

6.8.1 Function Documentation . . . 13

6.8.1.1 ruler . . . 13
6.9 lib.Interpreter Namespace Reference . . . 14
6.9.1 Variable Documentation . . . 14
6.9.1.1 obj . . . 14
6.10 lib.Interpreter_config Namespace Reference . . . 14
6.10.1 Variable Documentation . . . 15
6.10.1.1 obj . . . 15
6.11 lib.Interpreter_query Namespace Reference . . . 15
6.11.1 Variable Documentation . . . 15
6.11.1.1 obj . . . 15
6.12 lib.Interpreter_results Namespace Reference . . . 15
6.12.1 Variable Documentation . . . 15
6.12.1.1 obj . . . 15
6.13 lib.Interpreter_scan Namespace Reference . . . 16
6.13.1 Variable Documentation . . . 16
6.13.1.1 obj . . . 16
6.14 lib.Interpreter_update Namespace Reference . . . 16
6.14.1 Variable Documentation . . . 16
6.14.1.1 obj . . . 16
6.15 lib.isisdiag Namespace Reference . . . 16
6.15.1 Function Documentation . . . 17
6.15.1.1 createGraph . . . 17
6.15.1.2 fetchConfig . . . 17
6.15.1.3 scrapeConfig . . . 17
6.15.2 Variable Documentation . . . 18
6.15.2.1 __author__ . . . 18
6.15.2.2 G . . . 18
6.16 lib.querydiag Namespace Reference . . . 18
6.16.1 Function Documentation . . . 18
6.16.1.1 createGraph . . . 18
6.17 lib.xml2datacustom Namespace Reference . . . 19
6.17.1 Function Documentation . . . 19
6.17.1.1 xml2obj . . . 19
6.17.1.2 xml_jcfg2data . . . 19
6.17.2 Variable Documentation . . . 20
6.17.2.1 _attrs . . . 20
6.17.2.2 current . . . 20
6.17.2.3 data . . . 20
6.17.2.4 root . . . 20
6.17.2.5 stack . . . 20

6.17.2.6 text_parts . . . 20
6.18 xml2data_test Namespace Reference . . . 20
6.18.1 Variable Documentation . . . 20
6.18.1.1 config . . . 20
6.18.1.2 configdata . . . 20

7 Class Documentation **21**
7.1 lib.Baseconfig.Baseconfig Class Reference . . . 21
7.1.1 Detailed Description . . . 21
7.1.2 Constructor & Destructor Documentation . . . 21
7.1.2.1 __init__ . . . 22
7.1.3 Member Data Documentation . . . 22
7.1.3.1 CONFIG . . . 22
7.1.3.2 CONFIGpath . . . 22
7.2 lib.Interpreter.Interpreter Class Reference . . . 22
7.2.1 Detailed Description . . . 25
7.2.2 Constructor & Destructor Documentation . . . 25
7.2.2.1 __init__ . . . 25
7.2.3 Member Function Documentation . . . 25
7.2.3.1 do_config . . . 25
7.2.3.2 do_exit . . . 25
7.2.3.3 do_query . . . 25
7.2.3.4 do_results . . . 26
7.2.3.5 do_scan . . . 26
7.2.3.6 do_shell . . . 26
7.2.3.7 do_update . . . 26
7.2.3.8 do_userguide . . . 27
7.2.3.9 emptyline . . . 27
7.2.3.10 help_config . . . 27
7.2.3.11 help_exit . . . 27
7.2.3.12 help_help . . . 27
7.2.3.13 help_query . . . 27
7.2.3.14 help_results . . . 27
7.2.3.15 help_scan . . . 28
7.2.3.16 help_shell . . . 28
7.2.3.17 help_update . . . 28
7.2.3.18 help_userguide . . . 28
7.2.3.19 setconfig . . . 28
7.2.4 Member Data Documentation . . . 29
7.2.4.1 configinterpreter . . . 29

7.2.4.2 intro . . . 29
7.2.4.3 last_output . . . 29
7.2.4.4 myconfig . . . 29
7.2.4.5 prompt . . . 29
7.2.4.6 queryinterpreter . . . 29
7.2.4.7 resultsinterpreter . . . 29
7.2.4.8 scaninterpreter . . . 29
7.2.4.9 updateinterpreter . . . 29
7.3 lib.Interpreter_config.Interpreter_config Class Reference . . . 29
7.3.1 Detailed Description . . . 32
7.3.2 Constructor & Destructor Documentation . . . 32
7.3.2.1 __init__ . . . 32
7.3.3 Member Function Documentation . . . 32
7.3.3.1 do_exit . . . 32
7.3.3.2 do_load . . . 32
7.3.3.3 do_save . . . 32
7.3.3.4 do_shell . . . 32
7.3.3.5 do_show . . . 33
7.3.3.6 do_unload . . . 33
7.3.3.7 do_wizard . . . 33
7.3.3.8 emptyline . . . 34
7.3.3.9 help_exit . . . 34
7.3.3.10 help_load . . . 34
7.3.3.11 help_save . . . 34
7.3.3.12 help_show . . . 35
7.3.3.13 help_unload . . . 35
7.3.3.14 help_wizard . . . 35
7.3.4 Member Data Documentation . . . 35
7.3.4.1 intro . . . 35
7.3.4.2 last_output . . . 35
7.3.4.3 myconfig . . . 35
7.3.4.4 prompt . . . 36
7.4 lib.Interpreter_query.Interpreter_query Class Reference . . . 36
7.4.1 Detailed Description . . . 37
7.4.2 Constructor & Destructor Documentation . . . 38
7.4.2.1 __init__ . . . 38
7.4.3 Member Function Documentation . . . 38
7.4.3.1 do_exit . . . 38
7.4.3.2 do_query . . . 38
7.4.3.3 do_shell . . . 38

7.4.3.4 emptyline . . . 38
7.4.3.5 help_exit . . . 38
7.4.3.6 help_query . . . 38
7.4.4 Member Data Documentation . . . 39
7.4.4.1 intro . . . 39
7.4.4.2 last_output . . . 39
7.4.4.3 prompt . . . 39
7.5 lib.Interpreter_results.Interpreter_results Class Reference . . . 39
7.5.1 Detailed Description . . . 42
7.5.2 Constructor & Destructor Documentation . . . 42
7.5.2.1 __init__ . . . 42
7.5.3 Member Function Documentation . . . 42
7.5.3.1 do_detail . . . 42
7.5.3.2 do_diagram . . . 42
7.5.3.3 do_exit . . . 42
7.5.3.4 do_shell . . . 42
7.5.3.5 do_summary . . . 43
7.5.3.6 emptyline . . . 43
7.5.3.7 help_detail . . . 44
7.5.3.8 help_diagram . . . 44
7.5.3.9 help_exit . . . 44
7.5.3.10 help_summary . . . 44
7.5.3.11 query_diag . . . 44
7.5.4 Member Data Documentation . . . 45
7.5.4.1 intro . . . 45
7.5.4.2 last_output . . . 45
7.5.4.3 prompt . . . 45
7.6 lib.Interpreter_scan.Interpreter_scan Class Reference . . . 45
7.6.1 Detailed Description . . . 47
7.6.2 Constructor & Destructor Documentation . . . 48
7.6.2.1 __init__ . . . 48
7.6.3 Member Function Documentation . . . 48
7.6.3.1 do_autoscan . . . 48
7.6.3.2 do_exit . . . 48
7.6.3.3 do_shell . . . 48
7.6.3.4 emptyline . . . 48
7.6.3.5 help_autoscan . . . 49
7.6.3.6 help_exit . . . 49
7.6.4 Member Data Documentation . . . 49
7.6.4.1 intro . . . 49

7.6.4.2 last_output . . . 49
7.6.4.3 prompt . . . 49
7.7 lib.Interpreter_update.Interpreter_update Class Reference . . . 49
7.7.1 Detailed Description . . . 51
7.7.2 Constructor & Destructor Documentation . . . 52
7.7.2.1 __init__ . . . 52
7.7.3 Member Function Documentation . . . 52
7.7.3.1 do_exit . . . 52
7.7.3.2 do_shell . . . 52
7.7.3.3 do_update . . . 52
7.7.3.4 emptyline . . . 52
7.7.3.5 help_exit . . . 52
7.7.3.6 help_update . . . 52
7.7.4 Member Data Documentation . . . 52
7.7.4.1 intro . . . 52
7.7.4.2 last_output . . . 53
7.7.4.3 prompt . . . 53

8 File Documentation **55**

8.1 /home/niall/Code/hephaestus/hephaestus/builddocs.py File Reference . . . 55
8.2 /home/niall/Code/hephaestus/hephaestus/builddocs.py . . . 55
8.3 /home/niall/Code/hephaestus/hephaestus/demo.py File Reference . . . 55
8.4 /home/niall/Code/hephaestus/hephaestus/demo.py . . . 56
8.5 config.dox File Reference . . . 56
8.6 /home/niall/Code/hephaestus/hephaestus/hephaestus.py File Reference . . . 56
8.7 /home/niall/Code/hephaestus/hephaestus/hephaestus.py . . . 56
8.8 /home/niall/Code/hephaestus/hephaestus/ipaddr_test.py File Reference . . . 57
8.9 /home/niall/Code/hephaestus/hephaestus/ipaddr_test.py . . . 57
8.10 /home/niall/Code/hephaestus/hephaestus/lib/__init__.py File Reference . . . 57
8.11 __init__.py . . . 58
8.12 /home/niall/Code/hephaestus/hephaestus/lib/Baseconfig.py File Reference . . . 58
8.13 Baseconfig.py . . . 58
8.14 /home/niall/Code/hephaestus/hephaestus/lib/configparser.py File Reference . . . 58
8.15 configparser.py . . . 58
8.16 /home/niall/Code/hephaestus/hephaestus/lib/hephaestus_utils.py File Reference . . . 59
8.17 hephaestus_utils.py . . . 59
8.18 /home/niall/Code/hephaestus/hephaestus/lib/Interpreter.py File Reference . . . 59
8.19 Interpreter.py . . . 60
8.20 /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_config.py File Reference . . . 61
8.21 Interpreter_config.py . . . 61

8.22 /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_query.py File Reference 63
8.23 Interpreter_query.py 64
8.24 /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_results.py File Reference 67
8.25 Interpreter_results.py 68
8.26 /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_scan.py File Reference 70
8.27 Interpreter_scan.py 70
8.28 /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_update.py File Reference 72
8.29 Interpreter_update.py 72
8.30 /home/niall/Code/hephaestus/hephaestus/lib/isisdiag.py File Reference 74
8.31 isisdiag.py 74
8.32 /home/niall/Code/hephaestus/hephaestus/lib/querydiag.py File Reference 76
8.33 querydiag.py 77
8.34 /home/niall/Code/hephaestus/hephaestus/lib/xml2datacustom.py File Reference 77
8.35 xml2datacustom.py 78
8.36 /home/niall/Code/hephaestus/hephaestus/README.md File Reference 79
8.37 /home/niall/Code/hephaestus/hephaestus/README.md 79
8.38 /home/niall/Code/hephaestus/hephaestus/xml2data_test.py File Reference 80
8.39 /home/niall/Code/hephaestus/hephaestus/xml2data_test.py 80

Index **80**

Chapter 1

hephaestus

hephaestus: Automating service-provider network troubleshooting using Python

author : Niall Donaghy, (c) 2013-2014

contact: niall@ndonaghy.com

project: http://ndonaghy.com/hephaestus

Abstract:

Internet service provider networks are complex systems comprising multiple routers each autonomously forwarding data packets. When a packet is inadvertently blocked by a router the resultant troubleshooting process can be time-consuming and error-prone because as little as one line of configuration nestled amongst tens of thousands can be the culprit. This dissertation details hephaestus†, a research project comprising an interactive Python prototype which improves dramatically both the speed and accuracy of this process by leveraging automation at key junctures. Specifically, this application interrogates Juniper Networks routers running JunOS and the pan-European IP/MPLS network GÉANT serves as the testbed under examination.

† The name hephaestus is borrowed from that of the Greek god of fire and metallurgy in reference to the subject matter: big iron‡ backbone routers and firewalls.

‡ 'big iron', as the hackers' dictionary the Jargon File defines it, "refers to large, expensive, ultra-fast computers." Wikipedia states, "More recently the term is also applied to powerful computer servers and computer ranches, whose steel racks invoke the same association."

Chapter 2

Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

builddocs . . . 11
demo . . . 11
hephaestus . . . 11
ipaddr_test . . . 11
lib . . . 12
lib.Baseconfig . . . 12
lib.configparser . . . 12
lib.hephaestus_utils . . . 13
lib.Interpreter . . . 14
lib.Interpreter_config . . . 14
lib.Interpreter_query . . . 15
lib.Interpreter_results . . . 15
lib.Interpreter_scan . . . 16
lib.Interpreter_update . . . 16
lib.isisdiag . . . 16
lib.querydiag . . . 18
lib.xml2datacustom . . . 19
xml2data_test . . . 20

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

lib.Baseconfig.Baseconfig 21
Cmd
  lib.Interpreter.Interpreter 22
  lib.Interpreter_config.Interpreter_config 29
  lib.Interpreter_query.Interpreter_query 36
  lib.Interpreter_results.Interpreter_results 39
  lib.Interpreter_scan.Interpreter_scan 45
  lib.Interpreter_update.Interpreter_update 49

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

lib.Baseconfig.Baseconfig
: Class acts as a container for pointers to global configuration object and to the current configuration file's path . . . 21

lib.Interpreter.Interpreter
: Interpreter class based on the 'cmd' library . . . 22

lib.Interpreter_config.Interpreter_config
: Interpreter_config class based on the 'cmd' library . . . 29

lib.Interpreter_query.Interpreter_query
: Interpreter_query class based on the 'cmd' library . . . 36

lib.Interpreter_results.Interpreter_results
: Interpreter_results class based on the 'cmd' library . . . 39

lib.Interpreter_scan.Interpreter_scan
: Interpreter_scan class based on the 'cmd' library . . . 45

lib.Interpreter_update.Interpreter_update
: Interpreter_update class based on the 'cmd' library . . . 49

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

/home/niall/Code/hephaestus/hephaestus/builddocs.py . . . 55
/home/niall/Code/hephaestus/hephaestus/demo.py . . . 55
/home/niall/Code/hephaestus/hephaestus/hephaestus.py . . . 56
/home/niall/Code/hephaestus/hephaestus/ipaddr_test.py . . . 57
/home/niall/Code/hephaestus/hephaestus/xml2data_test.py . . . 80
/home/niall/Code/hephaestus/hephaestus/lib/__init__.py . . . 58
/home/niall/Code/hephaestus/hephaestus/lib/Baseconfig.py . . . 58
/home/niall/Code/hephaestus/hephaestus/lib/configparser.py . . . 58
/home/niall/Code/hephaestus/hephaestus/lib/hephaestus_utils.py . . . 59
/home/niall/Code/hephaestus/hephaestus/lib/Interpreter.py . . . 60
/home/niall/Code/hephaestus/hephaestus/lib/Interpreter_config.py . . . 61
/home/niall/Code/hephaestus/hephaestus/lib/Interpreter_query.py . . . 64
/home/niall/Code/hephaestus/hephaestus/lib/Interpreter_results.py . . . 68
/home/niall/Code/hephaestus/hephaestus/lib/Interpreter_scan.py . . . 70
/home/niall/Code/hephaestus/hephaestus/lib/Interpreter_update.py . . . 72
/home/niall/Code/hephaestus/hephaestus/lib/isisdiag.py . . . 74
/home/niall/Code/hephaestus/hephaestus/lib/querydiag.py . . . 77
/home/niall/Code/hephaestus/hephaestus/lib/xml2datacustom.py . . . 78

Chapter 6

Namespace Documentation

6.1 builddocs Namespace Reference

6.2 demo Namespace Reference

Variables

- tuple child = pexpect.spawn('python hephaestus.py')

6.2.1 Variable Documentation

6.2.1.1 tuple demo.child = pexpect.spawn('python hephaestus.py')

Definition at line 4 of file demo.py.

6.3 hephaestus Namespace Reference

Functions

- def main

  Instantiate the main interpreter and start it.

6.3.1 Function Documentation

6.3.1.1 def hephaestus.main ( )

Instantiate the main interpreter and start it.

Definition at line 13 of file hephaestus.py.

6.4 ipaddr_test Namespace Reference

Functions

- def main

6.4.1 Function Documentation

6.4.1.1 def ipaddr_test.main ( )

Definition at line 4 of file ipaddr_test.py.

6.5 lib Namespace Reference

Namespaces

- namespace Baseconfig
- namespace configparser
- namespace hephaestus_utils
- namespace Interpreter
- namespace Interpreter_config
- namespace Interpreter_query
- namespace Interpreter_results
- namespace Interpreter_scan
- namespace Interpreter_update
- namespace isisdiag
- namespace querydiag
- namespace xml2datacustom

6.6 lib.Baseconfig Namespace Reference

Classes

- class Baseconfig

  Class acts as a container for pointers to global configuration object and to the current configuration file's path.

6.7 lib.configparser Namespace Reference

Functions

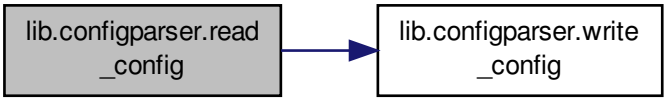
- def read_config
- def write_config

6.7.1 Function Documentation

6.7.1.1 def lib.configparser.read_config (*filename*)

Definition at line 9 of file configparser.py.

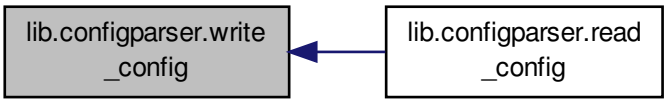
Here is the call graph for this function:



6.7.1.2 def lib.configparser.write_config (*filename, config*)

Definition at line 19 of file configparser.py.

Here is the caller graph for this function:



6.8 lib.hephaestus_utils Namespace Reference

Functions

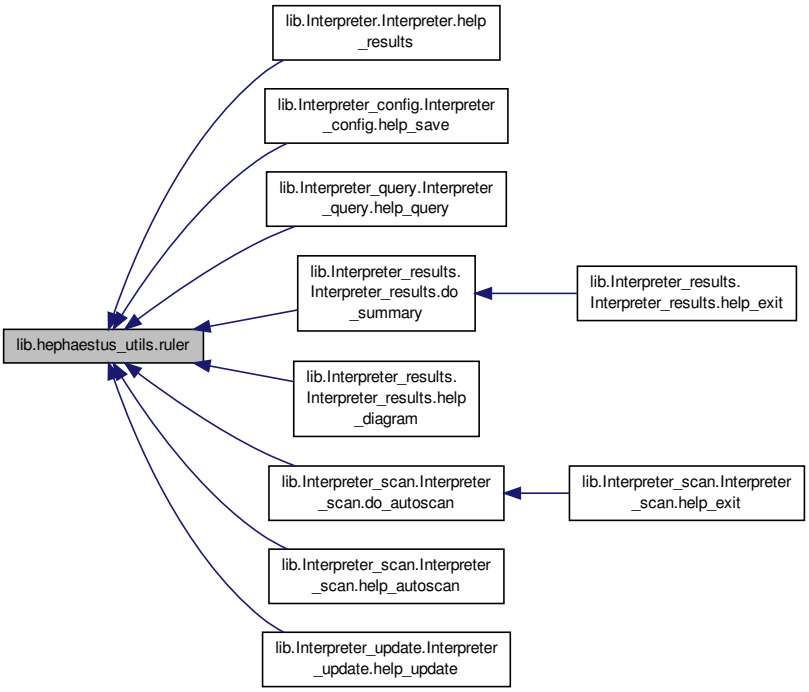
- def ruler

6.8.1 Function Documentation

6.8.1.1 def lib.hephaestus_utils.ruler (*char*)

Definition at line 13 of file hephaestus_utils.py.

Here is the caller graph for this function:



6.9 lib.Interpreter Namespace Reference

Classes

- class Interpreter

  Interpreter class based on the 'cmd' library.

Variables

- tuple obj = Interpreter()

6.9.1 Variable Documentation

6.9.1.1 tuple lib.Interpreter.obj = Interpreter()

Definition at line 119 of file Interpreter.py.

6.10 lib.Interpreter_config Namespace Reference

Classes

- class Interpreter_config

  Interpreter_config class based on the 'cmd' library.

Variables

- tuple obj = Interpreter_config()

6.10.1 Variable Documentation

6.10.1.1 tuple lib.Interpreter_config.obj = Interpreter_config()

Definition at line 154 of file Interpreter_config.py.

6.11 lib.Interpreter_query Namespace Reference

Classes

- class Interpreter_query

  Interpreter_query class based on the 'cmd' library.

Variables

- tuple obj = Interpreter_query()

6.11.1 Variable Documentation

6.11.1.1 tuple lib.Interpreter_query.obj = Interpreter_query()

Definition at line 261 of file Interpreter_query.py.

6.12 lib.Interpreter_results Namespace Reference

Classes

- class Interpreter_results

  Interpreter_results class based on the 'cmd' library.

Variables

- tuple obj = Interpreter_results()

6.12.1 Variable Documentation

6.12.1.1 tuple lib.Interpreter_results.obj = Interpreter_results()

Definition at line 160 of file Interpreter_results.py.

6.13 lib.Interpreter_scan Namespace Reference

Classes

- class Interpreter_scan

  Interpreter_scan class based on the 'cmd' library.

Variables

- tuple obj = Interpreter_scan()

6.13.1 Variable Documentation

6.13.1.1 tuple lib.Interpreter_scan.obj = Interpreter_scan()

Definition at line 145 of file Interpreter_scan.py.

6.14 lib.Interpreter_update Namespace Reference

Classes

- class Interpreter_update

  Interpreter_update class based on the 'cmd' library.

Variables

- tuple obj = Interpreter_update()

6.14.1 Variable Documentation

6.14.1.1 tuple lib.Interpreter_update.obj = Interpreter_update()

Definition at line 93 of file Interpreter_update.py.

6.15 lib.isisdiag Namespace Reference

Functions

- def fetchConfig

  Spawn SSH process, connect to `monitor@carl.ncc.dante.net` *and execute 'command', saving output.*
- def scrapeConfig
- def createGraph

  Return a graph from the data in geant_dynamic.txt.

Variables

- string __author__ = """Niall Donaghy (niall@ndonaghy.com)"""

  Connect to all routers in GEANT, obtain IS-IS adjacency information, graph it.
- tuple G = createGraph()

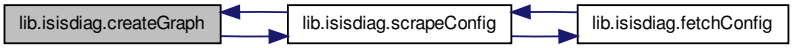
6.15.1 Function Documentation

6.15.1.1 def lib.isisdiag.createGraph ( )

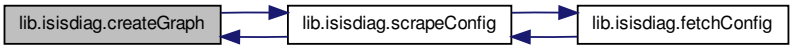
Return a graph from the data in geant_dynamic.txt.

Definition at line 123 of file isisdiag.py.

Here is the call graph for this function:



Here is the caller graph for this function:

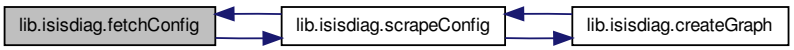


6.15.1.2 def lib.isisdiag.fetchConfig (*command*)

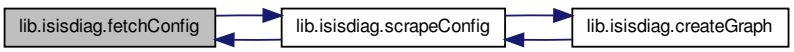
Spawn SSH process, connect to monitor@carl.ncc.dante.net and execute 'command', saving output.

Definition at line 13 of file isisdiag.py.

Here is the call graph for this function:



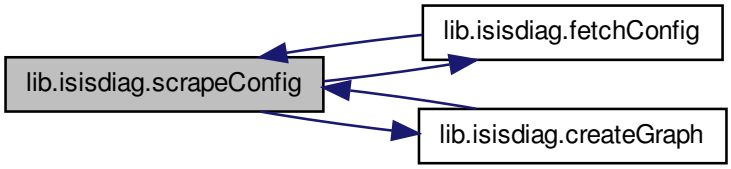
Here is the caller graph for this function:



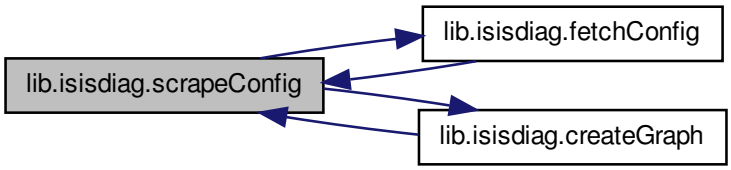
6.15.1.3 def lib.isisdiag.scrapeConfig ( )

Definition at line 24 of file isisdiag.py.

Here is the call graph for this function:



Here is the caller graph for this function:



6.15.2 Variable Documentation

6.15.2.1 string lib.isisdiag.__author__ = """Niall Donaghy (niall@ndonaghy.com)"""

Connect to all routers in GEANT, obtain IS-IS adjacency information, graph it.

Definition at line 5 of file isisdiag.py.

6.15.2.2 tuple lib.isisdiag.G = createGraph()

Definition at line 190 of file isisdiag.py.

6.16 lib.querydiag Namespace Reference

Functions

- def createGraph

6.16.1 Function Documentation

6.16.1.1 def lib.querydiag.createGraph ( )

Definition at line 7 of file querydiag.py.

6.17 lib.xml2datacustom Namespace Reference

Functions

- def xml2obj
- def xml_jcfg2data

Variables

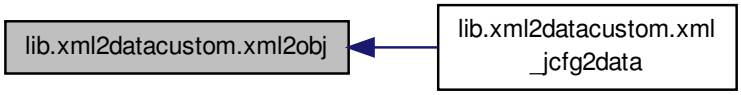
- _attrs
- data
- stack
- root
- current
- text_parts

6.17.1 Function Documentation

6.17.1.1 def lib.xml2datacustom.xml2obj (*src*)

Definition at line 14 of file xml2datacustom.py.

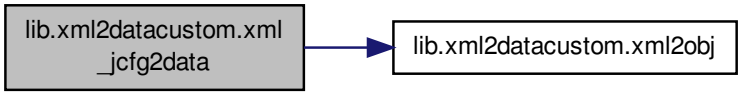
Here is the caller graph for this function:



6.17.1.2 def lib.xml2datacustom.xml_jcfg2data (*file*)

Definition at line 92 of file xml2datacustom.py.

Here is the call graph for this function:



6.17.2 Variable Documentation

6.17.2.1 lib.xml2datacustom._attrs

Definition at line 21 of file xml2datacustom.py.

6.17.2.2 lib.xml2datacustom.current

Definition at line 62 of file xml2datacustom.py.

6.17.2.3 lib.xml2datacustom.data

Definition at line 22 of file xml2datacustom.py.

6.17.2.4 lib.xml2datacustom.root

Definition at line 61 of file xml2datacustom.py.

6.17.2.5 lib.xml2datacustom.stack

Definition at line 60 of file xml2datacustom.py.

6.17.2.6 lib.xml2datacustom.text_parts

Definition at line 63 of file xml2datacustom.py.

6.18 xml2data_test Namespace Reference

Variables

- string config = 'etc/routerconfigs/62.40.96.2_config'
- tuple configdata = xml2datacustom.xml_jcfg2data(config)

6.18.1 Variable Documentation

6.18.1.1 string xml2data_test.config = 'etc/routerconfigs/62.40.96.2_config'

Definition at line 3 of file xml2data_test.py.

6.18.1.2 tuple xml2data_test.configdata = xml2datacustom.xml_jcfg2data(config)

Definition at line 4 of file xml2data_test.py.

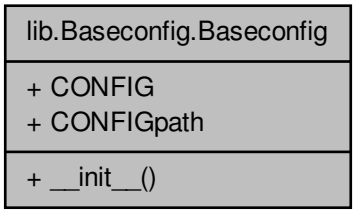
Chapter 7

Class Documentation

7.1 lib.Baseconfig.Baseconfig Class Reference

Class acts as a container for pointers to global configuration object and to the current configuration file's path.

Collaboration diagram for lib.Baseconfig.Baseconfig:



Public Member Functions

- def __init__

Public Attributes

- CONFIG
- CONFIGpath

7.1.1 Detailed Description

Class acts as a container for pointers to global configuration object and to the current configuration file's path.

Definition at line 4 of file Baseconfig.py.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 def lib.Baseconfig.Baseconfig.__init__ (*self*)

Definition at line 10 of file Baseconfig.py.

7.1.3 Member Data Documentation

7.1.3.1 lib.Baseconfig.Baseconfig.CONFIG

Definition at line 11 of file Baseconfig.py.

7.1.3.2 lib.Baseconfig.Baseconfig.CONFIGpath

Definition at line 12 of file Baseconfig.py.

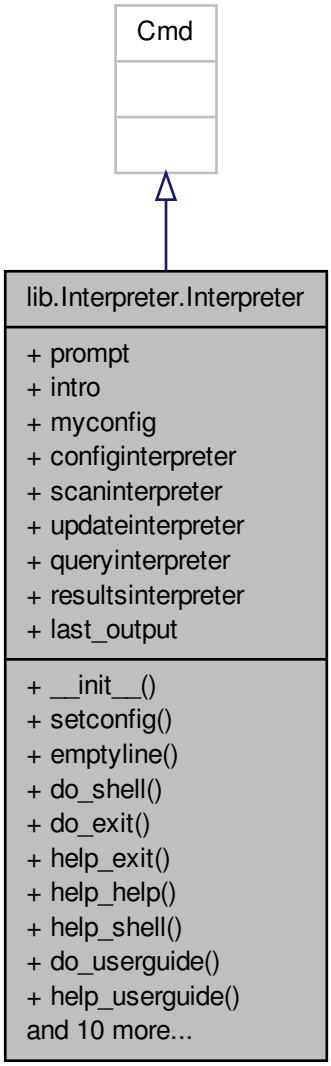
The documentation for this class was generated from the following file:

- /home/niall/Code/hephaestus/hephaestus/lib/Baseconfig.py

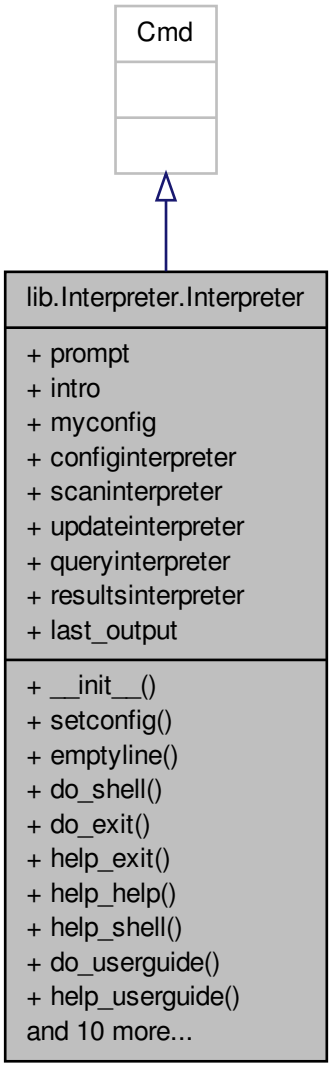
7.2 lib.Interpreter.Interpreter Class Reference

Interpreter class based on the 'cmd' library.

Inheritance diagram for lib.Interpreter.Interpreter:



Collaboration diagram for lib.Interpreter.Interpreter:



Public Member Functions

- def __init__
- def setconfig
- def emptyline
- def do_shell

  A punchy, pithy user guide will go here soon.
- def do_exit
- def help_exit
- def help_help
- def help_shell
- def do_userguide
- def help_userguide

- def do_config
- def help_config
- def do_scan
- def help_scan
- def do_update
- def help_update
- def do_query
- def help_query
- def do_results
- def help_results

Public Attributes

- prompt
- intro
- myconfig
- configinterpreter
- scaninterpreter
- updateinterpreter
- queryinterpreter
- resultsinterpreter
- last_output

  Print help on do_userguide() method.

7.2.1 Detailed Description

Interpreter class based on the 'cmd' library.

Definition at line 15 of file Interpreter.py.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 def lib.Interpreter.Interpreter.__init__ (*self*)

Definition at line 17 of file Interpreter.py.

7.2.3 Member Function Documentation

7.2.3.1 def lib.Interpreter.Interpreter.do_config (*self, line*)

Definition at line 87 of file Interpreter.py.

7.2.3.2 def lib.Interpreter.Interpreter.do_exit (*self, line*)

Definition at line 66 of file Interpreter.py.

7.2.3.3 def lib.Interpreter.Interpreter.do_query (*self, line*)

Definition at line 105 of file Interpreter.py.

7.2.3.4 def lib.Interpreter.Interpreter.do_results (*self, line*)

Definition at line 111 of file Interpreter.py.

7.2.3.5 def lib.Interpreter.Interpreter.do_scan (*self, line*)

Definition at line 93 of file Interpreter.py.

7.2.3.6 def lib.Interpreter.Interpreter.do_shell (*self, line*)

A punchy, pithy user guide will go here soon.

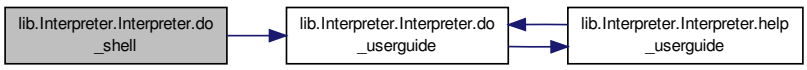
For now:

Basics

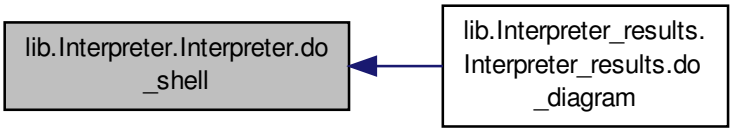
1. Enter Configuration mode, load a file, show contents, exit config > load > show > exit
2. Enter Scan mode, autoscan the network scan > autoscan
3. Enter Update mode, update local cache of router configs update > update
4. Enter Query mode, run query query > query
5. Enter Results mode, view results results > summary results > diagram results > detail

Definition at line 58 of file Interpreter.py.

Here is the call graph for this function:



Here is the caller graph for this function:



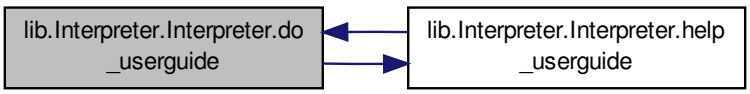
7.2.3.7 def lib.Interpreter.Interpreter.do_update (*self, line*)

Definition at line 99 of file Interpreter.py.

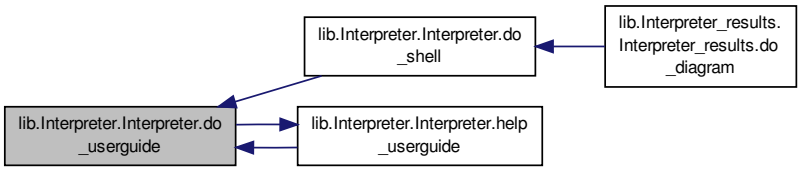
7.2.3.8 def lib.Interpreter.Interpreter.do_userguide (*self, line*)

Definition at line 79 of file Interpreter.py.

Here is the call graph for this function:



Here is the caller graph for this function:



7.2.3.9 def lib.Interpreter.Interpreter.emptyline (*self*)

Definition at line 36 of file Interpreter.py.

7.2.3.10 def lib.Interpreter.Interpreter.help_config (*self*)

Definition at line 90 of file Interpreter.py.

7.2.3.11 def lib.Interpreter.Interpreter.help_exit (*self*)

Definition at line 69 of file Interpreter.py.

7.2.3.12 def lib.Interpreter.Interpreter.help_help (*self*)

Definition at line 72 of file Interpreter.py.

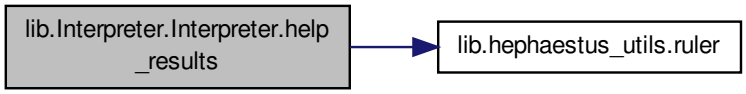
7.2.3.13 def lib.Interpreter.Interpreter.help_query (*self*)

Definition at line 108 of file Interpreter.py.

7.2.3.14 def lib.Interpreter.Interpreter.help_results (*self*)

Definition at line 114 of file Interpreter.py.

Here is the call graph for this function:



7.2.3.15 def lib.Interpreter.Interpreter.help_scan (*self*)

Definition at line 96 of file Interpreter.py.

7.2.3.16 def lib.Interpreter.Interpreter.help_shell (*self*)

Definition at line 75 of file Interpreter.py.

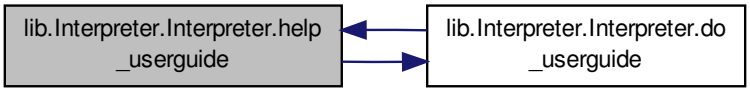
7.2.3.17 def lib.Interpreter.Interpreter.help_update (*self*)

Definition at line 102 of file Interpreter.py.

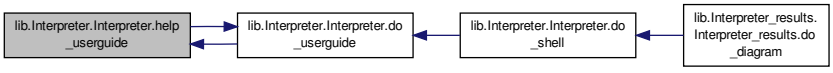
7.2.3.18 def lib.Interpreter.Interpreter.help_userguide (*self*)

Definition at line 82 of file Interpreter.py.

Here is the call graph for this function:



Here is the caller graph for this function:



7.2.3.19 def lib.Interpreter.Interpreter.setconfig (*self, cfg*)

Definition at line 33 of file Interpreter.py.

7.2.4 Member Data Documentation

7.2.4.1 lib.Interpreter.Interpreter.configinterpreter

Definition at line 22 of file Interpreter.py.

7.2.4.2 lib.Interpreter.Interpreter.intro

Definition at line 20 of file Interpreter.py.

7.2.4.3 lib.Interpreter.Interpreter.last_output

Print help on do_userguide() method.

Definition at line 64 of file Interpreter.py.

7.2.4.4 lib.Interpreter.Interpreter.myconfig

Definition at line 21 of file Interpreter.py.

7.2.4.5 lib.Interpreter.Interpreter.prompt

Definition at line 19 of file Interpreter.py.

7.2.4.6 lib.Interpreter.Interpreter.queryinterpreter

Definition at line 28 of file Interpreter.py.

7.2.4.7 lib.Interpreter.Interpreter.resultsinterpreter

Definition at line 30 of file Interpreter.py.

7.2.4.8 lib.Interpreter.Interpreter.scaninterpreter

Definition at line 24 of file Interpreter.py.

7.2.4.9 lib.Interpreter.Interpreter.updateinterpreter

Definition at line 26 of file Interpreter.py.

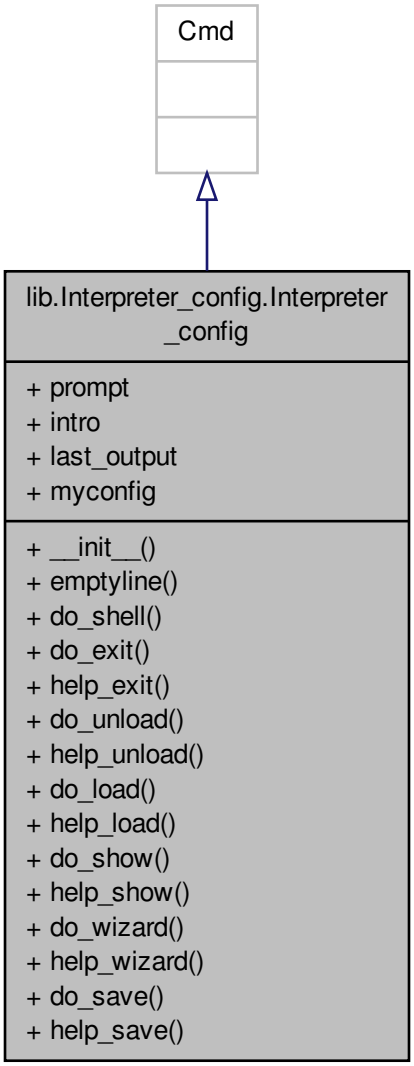
The documentation for this class was generated from the following file:

- /home/niall/Code/hephaestus/hephaestus/lib/Interpreter.py

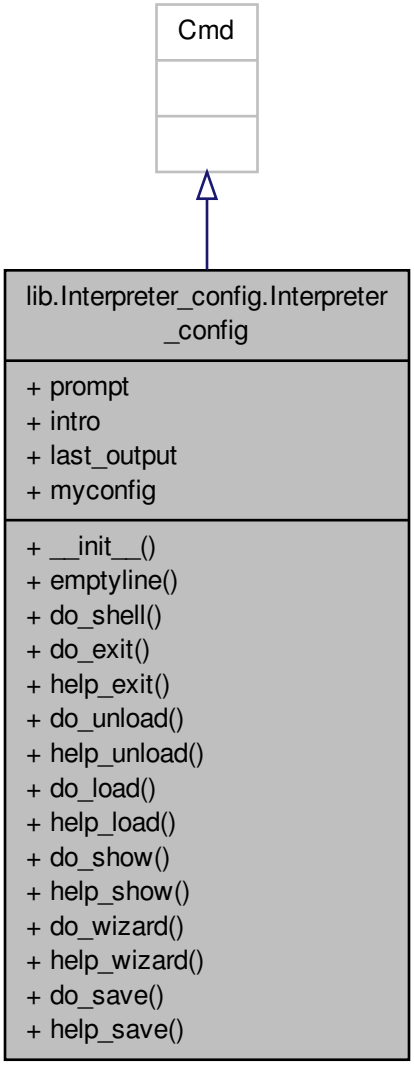
7.3 lib.Interpreter_config.Interpreter_config Class Reference

Interpreter_config class based on the 'cmd' library.

Inheritance diagram for lib.Interpreter_config.Interpreter_config:



Collaboration diagram for lib.Interpreter_config.Interpreter_config:



Public Member Functions

- def __init__

  Constructor calls parent constructor, sets the interpreter prompt and welcome message.
- def emptyline
- def do_shell

  Run shell commands from within this interpreter.
- def do_exit
- def help_exit
- def do_unload

  Unload the current configuration.
- def help_unload
- def do_load

- def help_load
- def do_show
- def help_show
- def do_wizard

  This function starts an interactive configuration wizard.
- def help_wizard
- def do_save
- def help_save

Public Attributes

- prompt
- intro
- last_output
- myconfig

7.3.1 Detailed Description

Interpreter_config class based on the 'cmd' library.

Definition at line 10 of file Interpreter_config.py.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 def lib.Interpreter_config.Interpreter_config.__init__ (*self*)

Constructor calls parent constructor, sets the interpreter prompt and welcome message.

Definition at line 13 of file Interpreter_config.py.

7.3.3 Member Function Documentation

7.3.3.1 def lib.Interpreter_config.Interpreter_config.do_exit (*self, line*)

Definition at line 28 of file Interpreter_config.py.

7.3.3.2 def lib.Interpreter_config.Interpreter_config.do_load (*self, line*)

Definition at line 42 of file Interpreter_config.py.

7.3.3.3 def lib.Interpreter_config.Interpreter_config.do_save (*self, line*)

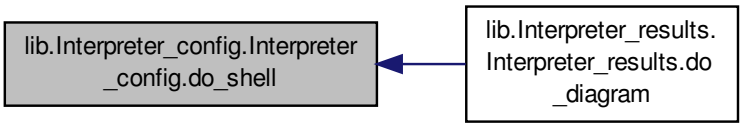
Definition at line 115 of file Interpreter_config.py.

7.3.3.4 def lib.Interpreter_config.Interpreter_config.do_shell (*self, line*)

Run shell commands from within this interpreter.

Definition at line 22 of file Interpreter_config.py.

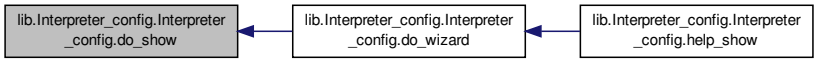
Here is the caller graph for this function:



7.3.3.5 def lib.Interpreter_config.Interpreter_config.do_show (*self, line*)

Definition at line 61 of file Interpreter_config.py.

Here is the caller graph for this function:

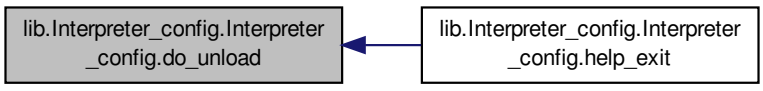


7.3.3.6 def lib.Interpreter_config.Interpreter_config.do_unload (*self, line*)

Unload the current configuration.

Definition at line 35 of file Interpreter_config.py.

Here is the caller graph for this function:

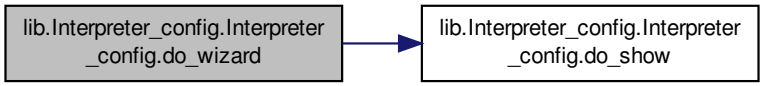


7.3.3.7 def lib.Interpreter_config.Interpreter_config.do_wizard (*self, line*)

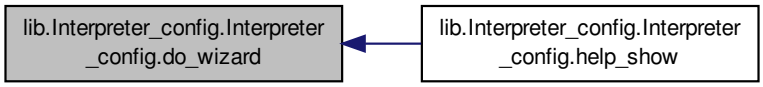
This function starts an interactive configuration wizard.

Definition at line 73 of file Interpreter_config.py.

Here is the call graph for this function:



Here is the caller graph for this function:



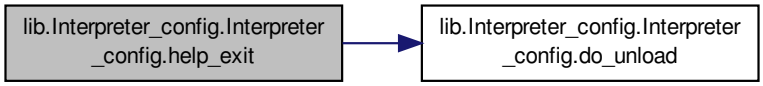
7.3.3.8 def lib.Interpreter_config.Interpreter_config.emptyline (*self*)

Definition at line 18 of file Interpreter_config.py.

7.3.3.9 def lib.Interpreter_config.Interpreter_config.help_exit (*self*)

Definition at line 31 of file Interpreter_config.py.

Here is the call graph for this function:



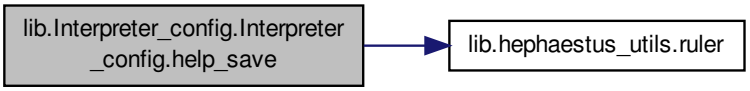
7.3.3.10 def lib.Interpreter_config.Interpreter_config.help_load (*self*)

Definition at line 58 of file Interpreter_config.py.

7.3.3.11 def lib.Interpreter_config.Interpreter_config.help_save (*self*)

Definition at line 149 of file Interpreter_config.py.

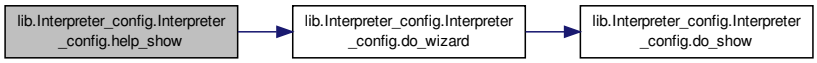
Here is the call graph for this function:



7.3.3.12 def lib.Interpreter_config.Interpreter_config.help_show (*self*)

Definition at line 69 of file Interpreter_config.py.

Here is the call graph for this function:



7.3.3.13 def lib.Interpreter_config.Interpreter_config.help_unload (*self*)

Definition at line 39 of file Interpreter_config.py.

7.3.3.14 def lib.Interpreter_config.Interpreter_config.help_wizard (*self*)

Definition at line 112 of file Interpreter_config.py.

7.3.4 Member Data Documentation

7.3.4.1 lib.Interpreter_config.Interpreter_config.intro

Definition at line 16 of file Interpreter_config.py.

7.3.4.2 lib.Interpreter_config.Interpreter_config.last_output

Definition at line 26 of file Interpreter_config.py.

7.3.4.3 lib.Interpreter_config.Interpreter_config.myconfig

Definition at line 36 of file Interpreter_config.py.

7.3.4.4 lib.Interpreter_config.Interpreter_config.prompt

Definition at line 15 of file Interpreter_config.py.

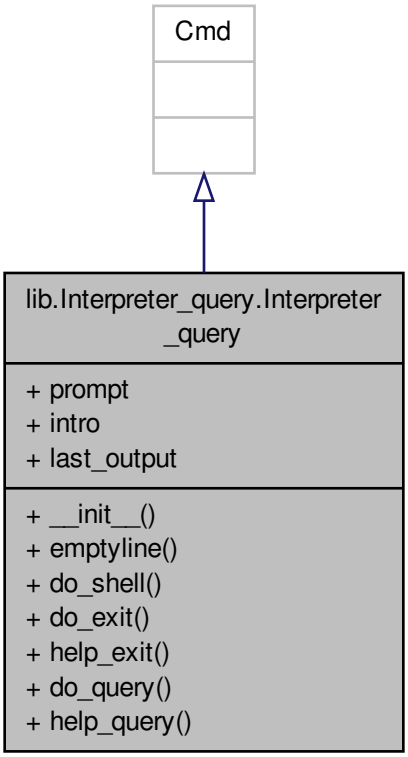
The documentation for this class was generated from the following file:

- /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_config.py

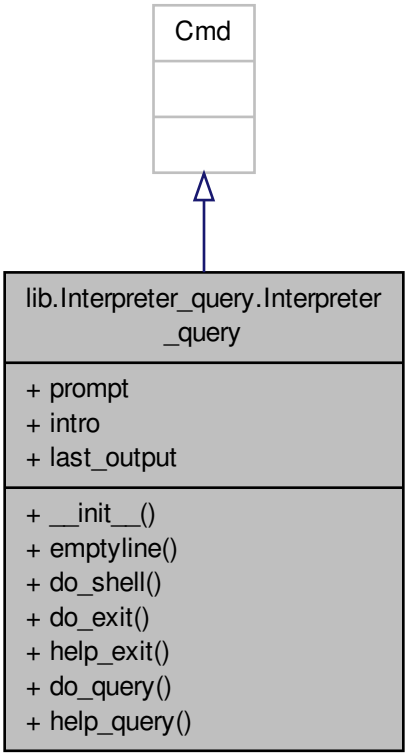
7.4 lib.Interpreter_query.Interpreter_query Class Reference

Interpreter_query class based on the 'cmd' library.

Inheritance diagram for lib.Interpreter_query.Interpreter_query:



Collaboration diagram for lib.Interpreter_query.Interpreter_query:



Public Member Functions

- def __init__
- def emptyline
- def do_shell
- def do_exit
- def help_exit
- def do_query
- def help_query

Public Attributes

- prompt
- intro
- last_output

7.4.1 Detailed Description

Interpreter_query class based on the 'cmd' library.

Definition at line 15 of file Interpreter_query.py.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 def lib.Interpreter_query.Interpreter_query.__init__ (*self*)

Definition at line 17 of file Interpreter_query.py.

7.4.3 Member Function Documentation

7.4.3.1 def lib.Interpreter_query.Interpreter_query.do_exit (*self, line*)

Definition at line 32 of file Interpreter_query.py.

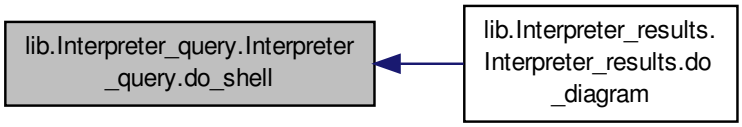
7.4.3.2 def lib.Interpreter_query.Interpreter_query.do_query (*self, line*)

Definition at line 38 of file Interpreter_query.py.

7.4.3.3 def lib.Interpreter_query.Interpreter_query.do_shell (*self, line*)

Definition at line 25 of file Interpreter_query.py.

Here is the caller graph for this function:



7.4.3.4 def lib.Interpreter_query.Interpreter_query.emptyline (*self*)

Definition at line 22 of file Interpreter_query.py.

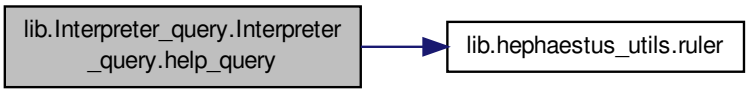
7.4.3.5 def lib.Interpreter_query.Interpreter_query.help_exit (*self*)

Definition at line 35 of file Interpreter_query.py.

7.4.3.6 def lib.Interpreter_query.Interpreter_query.help_query (*self*)

Definition at line 255 of file Interpreter_query.py.

Here is the call graph for this function:



7.4.4 Member Data Documentation

7.4.4.1 lib.Interpreter_query.Interpreter_query.intro

Definition at line 20 of file Interpreter_query.py.

7.4.4.2 lib.Interpreter_query.Interpreter_query.last_output

Definition at line 30 of file Interpreter_query.py.

7.4.4.3 lib.Interpreter_query.Interpreter_query.prompt

Definition at line 19 of file Interpreter_query.py.

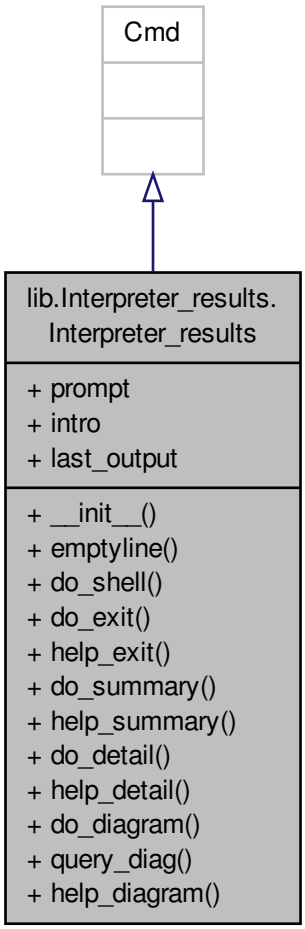
The documentation for this class was generated from the following file:

- /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_query.py

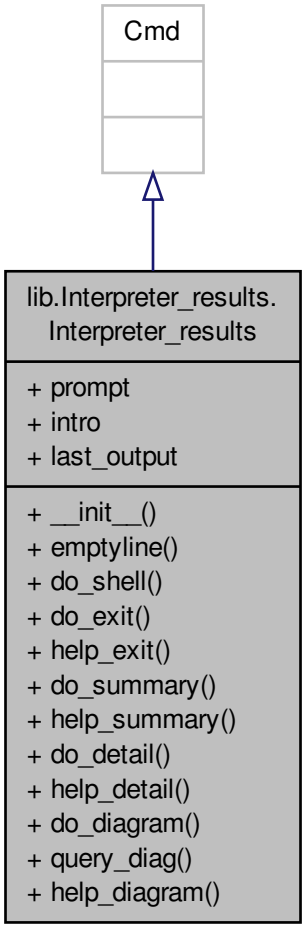
7.5 lib.Interpreter_results.Interpreter_results Class Reference

Interpreter_results class based on the 'cmd' library.

Inheritance diagram for lib.Interpreter_results.Interpreter_results:



Collaboration diagram for lib.Interpreter_results.Interpreter_results:



Public Member Functions

- def __init__
- def emptyline
- def do_shell

  Run a shell command.
- def do_exit
- def help_exit
- def do_summary

  Print results summary.
- def help_summary
- def do_detail
- def help_detail
- def do_diagram
- def query_diag
- def help_diagram

Public Attributes

- prompt
- intro
- last_output

7.5.1 Detailed Description

Interpreter_results class based on the 'cmd' library.

Definition at line 19 of file Interpreter_results.py.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 def lib.Interpreter_results.Interpreter_results.__init__ (*self*)

Definition at line 21 of file Interpreter_results.py.

7.5.3 Member Function Documentation

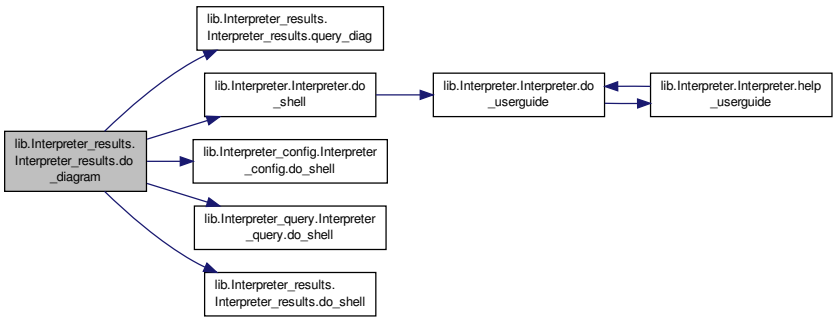
7.5.3.1 def lib.Interpreter_results.Interpreter_results.do_detail (*self, line*)

Definition at line 55 of file Interpreter_results.py.

7.5.3.2 def lib.Interpreter_results.Interpreter_results.do_diagram (*self, line*)

Definition at line 61 of file Interpreter_results.py.

Here is the call graph for this function:



7.5.3.3 def lib.Interpreter_results.Interpreter_results.do_exit (*self, line*)

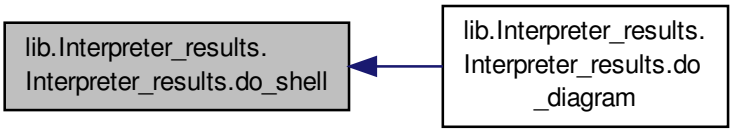
Definition at line 36 of file Interpreter_results.py.

7.5.3.4 def lib.Interpreter_results.Interpreter_results.do_shell (*self, line*)

Run a shell command.

Definition at line 30 of file Interpreter_results.py.

Here is the caller graph for this function:

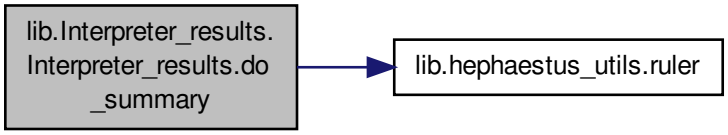


7.5.3.5 def lib.Interpreter_results.Interpreter_results.do_summary (*self, line*)

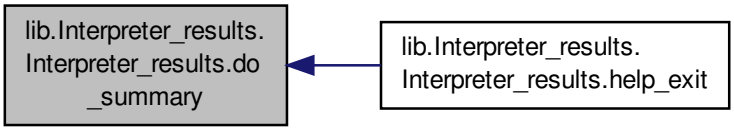
Print results summary.

Definition at line 43 of file Interpreter_results.py.

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3.6 def lib.Interpreter_results.Interpreter_results.emptyline (*self*)

Definition at line 26 of file Interpreter_results.py.

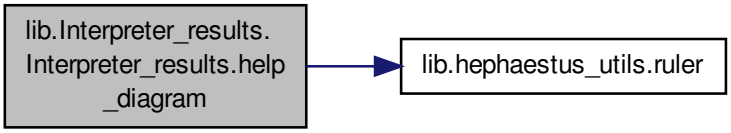
7.5.3.7 def lib.Interpreter_results.Interpreter_results.help_detail (*self, line*)

Definition at line 58 of file Interpreter_results.py.

7.5.3.8 def lib.Interpreter_results.Interpreter_results.help_diagram (*self, line*)

Definition at line 155 of file Interpreter_results.py.

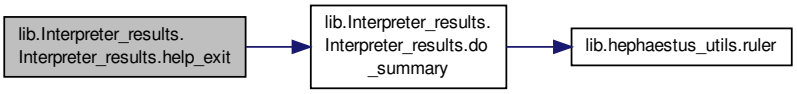
Here is the call graph for this function:



7.5.3.9 def lib.Interpreter_results.Interpreter_results.help_exit (*self*)

Definition at line 39 of file Interpreter_results.py.

Here is the call graph for this function:



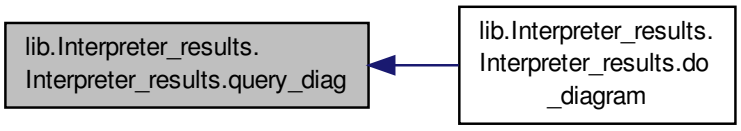
7.5.3.10 def lib.Interpreter_results.Interpreter_results.help_summary (*self*)

Definition at line 52 of file Interpreter_results.py.

7.5.3.11 def lib.Interpreter_results.Interpreter_results.query_diag (*self*)

Definition at line 74 of file Interpreter_results.py.

Here is the caller graph for this function:



7.5.4 Member Data Documentation

7.5.4.1 lib.Interpreter_results.Interpreter_results.intro

Definition at line 24 of file Interpreter_results.py.

7.5.4.2 lib.Interpreter_results.Interpreter_results.last_output

Definition at line 34 of file Interpreter_results.py.

7.5.4.3 lib.Interpreter_results.Interpreter_results.prompt

Definition at line 23 of file Interpreter_results.py.

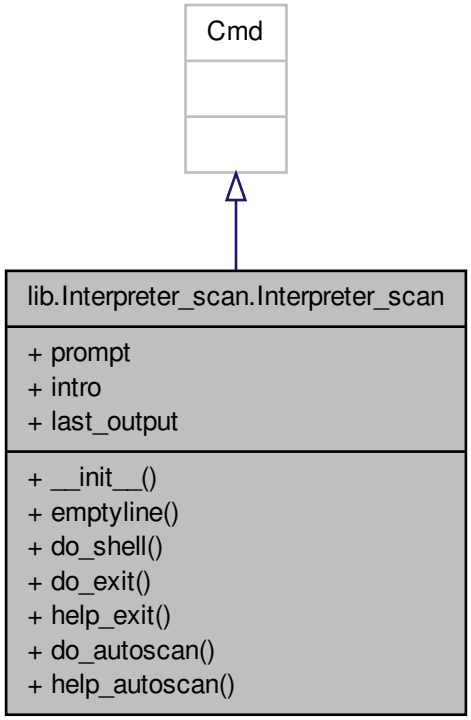
The documentation for this class was generated from the following file:

- /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_results.py

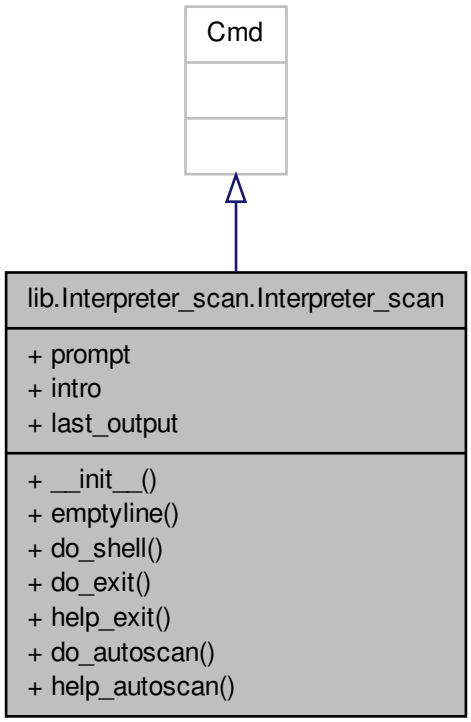
7.6 lib.Interpreter_scan.Interpreter_scan Class Reference

Interpreter_scan class based on the 'cmd' library.

Inheritance diagram for lib.Interpreter_scan.Interpreter_scan:



Collaboration diagram for lib.Interpreter_scan.Interpreter_scan:



Public Member Functions

- def __init__
- def emptyline
- def do_shell

  Run a shell command.
- def do_exit
- def help_exit
- def do_autoscan

  Automatically scan the network and identify any routers present.
- def help_autoscan

Public Attributes

- prompt
- intro
- last_output

7.6.1 Detailed Description

Interpreter_scan class based on the 'cmd' library.

Definition at line 16 of file Interpreter_scan.py.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 def lib.Interpreter_scan.Interpreter_scan.__init__ (*self*)

Definition at line 18 of file Interpreter_scan.py.

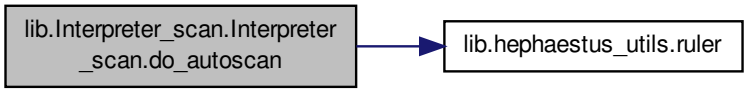
7.6.3 Member Function Documentation

7.6.3.1 def lib.Interpreter_scan.Interpreter_scan.do_autoscan (*self, line*)

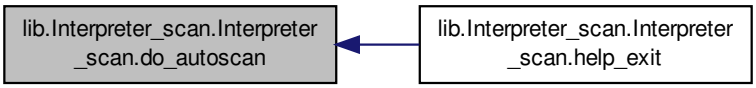
Automatically scan the network and identify any routers present.

Definition at line 40 of file Interpreter_scan.py.

Here is the call graph for this function:



Here is the caller graph for this function:



7.6.3.2 def lib.Interpreter_scan.Interpreter_scan.do_exit (*self, line*)

Definition at line 33 of file Interpreter_scan.py.

7.6.3.3 def lib.Interpreter_scan.Interpreter_scan.do_shell (*self, line*)

Run a shell command.

Definition at line 27 of file Interpreter_scan.py.

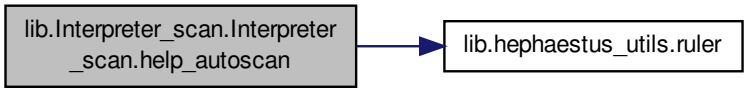
7.6.3.4 def lib.Interpreter_scan.Interpreter_scan.emptyline (*self*)

Definition at line 23 of file Interpreter_scan.py.

7.6.3.5 def lib.Interpreter_scan.Interpreter_scan.help_autoscan (*self*)

Definition at line 140 of file Interpreter_scan.py.

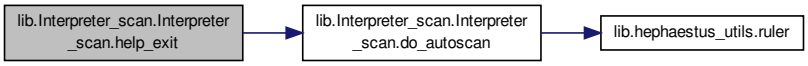
Here is the call graph for this function:



7.6.3.6 def lib.Interpreter_scan.Interpreter_scan.help_exit (*self*)

Definition at line 36 of file Interpreter_scan.py.

Here is the call graph for this function:



7.6.4 Member Data Documentation

7.6.4.1 lib.Interpreter_scan.Interpreter_scan.intro

Definition at line 21 of file Interpreter_scan.py.

7.6.4.2 lib.Interpreter_scan.Interpreter_scan.last_output

Definition at line 31 of file Interpreter_scan.py.

7.6.4.3 lib.Interpreter_scan.Interpreter_scan.prompt

Definition at line 20 of file Interpreter_scan.py.

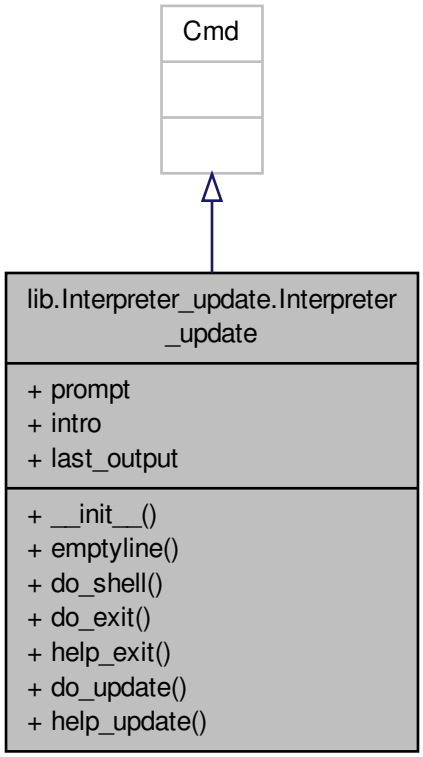
The documentation for this class was generated from the following file:

- /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_scan.py

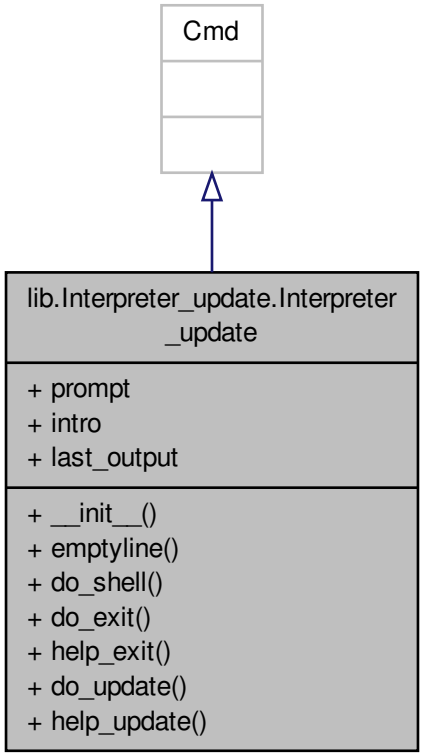
7.7 lib.Interpreter_update.Interpreter_update Class Reference

Interpreter_update class based on the 'cmd' library.

Inheritance diagram for lib.Interpreter_update.Interpreter_update:



Collaboration diagram for lib.Interpreter_update.Interpreter_update:



Public Member Functions

- def __init__
- def emptyline
- def do_shell
- def do_exit
- def help_exit
- def do_update
- def help_update

Public Attributes

- prompt
- intro
- last_output

7.7.1 Detailed Description

Interpreter_update class based on the 'cmd' library.

Definition at line 15 of file Interpreter_update.py.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 def lib.Interpreter_update.Interpreter_update.__init__ (*self*)

Definition at line 17 of file Interpreter_update.py.

7.7.3 Member Function Documentation

7.7.3.1 def lib.Interpreter_update.Interpreter_update.do_exit (*self, line*)

Definition at line 32 of file Interpreter_update.py.

7.7.3.2 def lib.Interpreter_update.Interpreter_update.do_shell (*self, line*)

Definition at line 25 of file Interpreter_update.py.

7.7.3.3 def lib.Interpreter_update.Interpreter_update.do_update (*self, line*)

Definition at line 38 of file Interpreter_update.py.

7.7.3.4 def lib.Interpreter_update.Interpreter_update.emptyline (*self*)

Definition at line 22 of file Interpreter_update.py.

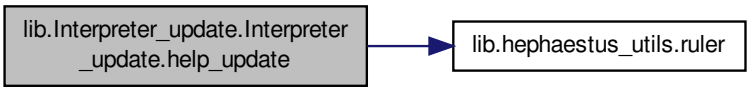
7.7.3.5 def lib.Interpreter_update.Interpreter_update.help_exit (*self*)

Definition at line 35 of file Interpreter_update.py.

7.7.3.6 def lib.Interpreter_update.Interpreter_update.help_update (*self*)

Definition at line 87 of file Interpreter_update.py.

Here is the call graph for this function:



7.7.4 Member Data Documentation

7.7.4.1 lib.Interpreter_update.Interpreter_update.intro

Definition at line 20 of file Interpreter_update.py.

7.7.4.2 lib.Interpreter_update.Interpreter_update.last_output

Definition at line 30 of file Interpreter_update.py.

7.7.4.3 lib.Interpreter_update.Interpreter_update.prompt

Definition at line 19 of file Interpreter_update.py.

The documentation for this class was generated from the following file:

- /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_update.py

Chapter 8

File Documentation

8.1 /home/niall/Code/hephaestus/hephaestus/builddocs.py File Reference

Namespaces

- namespace builddocs

8.2 /home/niall/Code/hephaestus/hephaestus/builddocs.py

```
00001 #!/usr/bin/env python
00002 import os
00003
00004 print("Building documentation...")
00005 print("Cleaning up...")
00006 os.system("cd docs/; rm -r html/ latex/")
00007 print("Running doxygen...")
00008 os.system("cd docs/; doxygen config.dox")  # run doxygen with our configuration file
00009 print("doxygen finished, running pdflatex...")
00010 os.system("cd docs/latex; make")  # run make in the latex output directory to generate a PDF
00011 print("Created PDF...")
00012 os.system("ls -lh docs/latex/refman.pdf")
00013 print("Uploading PDF to http://ndonaghy.com/hephaestus/docs")
00014 os.system("scp docs/latex/refman.pdf niall@ndonaghy.com:~/ndonaghy/hephaestus/docs")
00015 print("...done!")
00016 print("Creating HTML tarball...")
00017 os.system("tar cvzf html.tar.gz docs/html/")
00018 print("Uploading...")
00019 os.system("scp html.tar.gz niall@ndonaghy.com:~/ndonaghy/hephaestus")
00020 print("Expanding...")
00021 os.system("ssh niall@ndonaghy.com 'cd ~/ndonaghy/hephaestus/docs/html; rm -r *; cd ~/ndonaghy/hephaestus;
       tar xvzf html.tar.gz; rm html.tar.gz'")
00022 print("Cleaning up...")
00023 os.system("rm html.tar.gz")
00024 print("\n...done!\n")
```

8.3 /home/niall/Code/hephaestus/hephaestus/demo.py File Reference

Namespaces

- namespace demo

Variables

- tuple demo.child = pexpect.spawn('python hephaestus.py')

8.4 /home/niall/Code/hephaestus/hephaestus/demo.py

```
00001 #!/usr/bin/env python
00002 import pexpect
00003 import sys
00004 child = pexpect.spawn('python hephaestus.py')
00005 child.timeout=9999999999999
00006 child.logfile = sys.stdout
00007 child.setecho(False)
00008 child.expect('hephaestus>')
00009 child.sendline('config')
00010 child.expect('hephaestus\(config\)#')
00011 child.sendline('load')
00012 child.expect('load:\r\n')
00013 child.sendline('0')
00014 child.expect('hephaestus\(config\)#')
00015 child.sendline('exit')
00016 child.expect('hephaestus>')
00017 child.sendline('scan')
00018 child.expect('hephaestus\(scan\)>')
00019 #raw_input("Press any key to begin autoscanning the network...")
00020 child.sendline('autoscan')
00021 child.expect('hephaestus\(scan\)>')
00022 child.sendline('exit')
00023 child.expect('hephaestus>')
00024 child.sendline('update')
00025 child.expect('hephaestus\(update\)>')
00026 child.sendline('update')
00027 child.expect('hephaestus\(update\)>')
00028 child.sendline('exit')
00029 child.expect('hephaestus>')
00030 child.sendline('exit')
00031
```

8.5 config.dox File Reference

8.6 /home/niall/Code/hephaestus/hephaestus/hephaestus.py File Reference

Namespaces

- namespace hephaestus

Functions

- def hephaestus.main

  Instantiate the main interpreter and start it.

8.7 /home/niall/Code/hephaestus/hephaestus/hephaestus.py

```
00001 #!/usr/bin/env python
00002 """
00003 hephaestus is a stateless firewall configuration parser for JunOS
00004 author : Niall Donaghy, (c) 2013-2014
00005 contact: niall@ndonaghy.com
00006 license: http://opensource.org/licenses/MIT
00007 project: http://ndonaghy.com/hephaestus
00008 """
00009
00010 from lib.Interpreter import Interpreter
00011
00012 def main():
00013   """Instantiate the main interpreter and start it."""
00014   theinterpreter = Interpreter()
00015   theinterpreter.cmdloop()
00016
00017 if __name__ == '__main__':
00018   main()
```

8.8 /home/niall/Code/hephaestus/hephaestus/ipaddr_test.py File Reference

Namespaces

- namespace ipaddr_test

Functions

- def ipaddr_test.main

8.9 /home/niall/Code/hephaestus/hephaestus/ipaddr_test.py

```
00001 #!/usr/bin/python
00002 import ipaddr
00003
00004 def main():
00005
00006   # Create some IPv4 addresses and networks, and some IPv6 addresses and networks as strings
00007   foo4 = '62.40.96.16'
00008   foonet4 = '62.40.96.0/24'
00009   bar4 = '62.40.101.145'
00010   barnet4 = '62.40.101.128/25'
00011   foo6 = '2001:798::4'
00012   foonet6 = '2001:798::/32'
00013   bar6 = '2001:798::422:66'
00014   barnet6 = '2001:798::/32'
00015
00016   # Convert to ipaddr library objects
00017   foo4 = ipaddr.IPAddress(foo4)
00018   foonet4 = ipaddr.IPNetwork(foonet4)
00019   bar4 = ipaddr.IPAddress(bar4)
00020   barnet4 = ipaddr.IPNetwork(barnet4)
00021   foo6 = ipaddr.IPAddress(foo6)
00022   foonet6 = ipaddr.IPNetwork(foonet6)
00023   bar6 = ipaddr.IPAddress(bar6)
00024   barnet6 = ipaddr.IPNetwork(barnet6)
00025
00026   # Obtain some useful information
00027
00028   print("Let's test some IP addresses and IP networks:")
00029
00030   if foo4 in foonet4:
00031     print("IPv" + str(foo4.version) + " address " +  str(foo4) + " is in network " + str(foonet4))
00032
00033   print("IPv" + str(foonet4.version) + " network " +  str(foonet4) + " has broadcast address " + str(
      foonet4.broadcast))
00034
00035   if foo4 not in barnet4:
00036     print("IPv" + str(foo4.version) + " address " +  str(foo4) + " is not in network " + str(barnet4))
00037
00038   if foo6 in foonet6:
00039     print("IPv" + str(foo6.version) + " address " +  str(foo6) + " is in network " + str(foonet6))
00040
00041   if bar4 in barnet4:
00042     print("IPv" + str(bar4.version) + " address " +  str(bar4) + " is in network " + str(barnet4))
00043
00044   print("IPv" + str(barnet4.version) + " network " +  str(barnet4) + " has broadcast address " + str(
      barnet4.broadcast))
00045
00046   if bar6 in barnet6:
00047     print("IPv" + str(bar6.version) + " address " +  str(bar6) + " is in network " + str(barnet6))
00048
00049   exit(0)
00050
00051 if __name__ == "__main__":
00052   main()
```

8.10 /home/niall/Code/hephaestus/hephaestus/lib/__init__.py File Reference

Namespaces

- namespace lib

8.11 __init__.py

```
00001 #!/usr/bin/env python
00002 #
00003 # This file indicates to Python that this directory contains modules
00004 #
```

8.12 /home/niall/Code/hephaestus/hephaestus/lib/Baseconfig.py File Reference

Classes

- class lib.Baseconfig.Baseconfig

  Class acts as a container for pointers to global configuration object and to the current configuration file's path.

Namespaces

- namespace lib.Baseconfig

8.13 Baseconfig.py

```
00001 #!/usr/bin/env python
00002
00003 class Baseconfig :
00004   """Class acts as a container for pointers to global configuration object and to the current configuration
      file's path."""
00005   def __init__(self):
00006     """
00007     Constructor declares two variables and initialises them thus:
00008     'CONFIG' of type dictionary, empty, holds the current running configuration (memory)
00009     'CONFIGpath' of type string, empty, holds the absolute path for the configuration file (disk)
00010     """
00011     self.CONFIG = {}
00012     self.CONFIGpath = ''
00013
00014
```

8.14 /home/niall/Code/hephaestus/hephaestus/lib/configparser.py File Reference

Namespaces

- namespace lib.configparser

Functions

- def lib.configparser.read_config
- def lib.configparser.write_config

8.15 configparser.py

```
00001 #!/usr/bin/python
00002 import pickle
00003
00004 def read_config(filename):
00005     """
00006     Uses the pickle module to load an object previously 'pickled' and saved to disk.
00007     Takes one argument, a string containing the absolute path of the pickle file.
00008     Returns the unpickled object.
00009     """
00010     f = open(filename, 'rb')  # open file for reading in binary mode
```

```
00011     x = pickle.load(f)
00012     f.close()
00013     return x
00014
00015 def write_config(filename, config):
00016     """
00017     Uses the pickle module to 'pickle' an object, that is, write the object to a persistence file.
00018     Takes two arguments, a string containing the absolute path of the pickle file, and the object to be '
      pickled'.
00019     """
00020     f = open(filename, 'wb')  # open file for writing in binary mode
00021     pickle.dump(config, f, 2)
00022     f.close()
```

8.16 /home/niall/Code/hephaestus/hephaestus/lib/hephaestus_utils.py File Reference

Namespaces

- namespace lib.hephaestus_utils

Functions

- def lib.hephaestus_utils.ruler

8.17 hephaestus_utils.py

```
00001 import os
00002
00003 def ruler(char):
00004   """
00005   Dynamically generates a string containing a 'horizontal line' or 'ruler' sized to fit the current width
      of the tty.
00006   Takes one argument, a string containing the desired ruler char, and returns a string.
00007
00008   Examples:
00009     Arg "*" will return "****...<tty width>...****"
00010     Arg "=" will return "====...<tty width>...===="
00011   """
00012   rows, columns = os.popen('stty size', 'r').read().split()  # use stty output to generate a tuple denoting
      size
00013   string = []  # list used to build return string
00014   for i in range(int(columns)):  # for every column
00015     string.append(char.rstrip()) # append a char to the list, stripping newlines
00016   return "".join(string)  #  turn the list into a string
```

8.18 /home/niall/Code/hephaestus/hephaestus/lib/Interpreter.py File Reference

Classes

- class lib.Interpreter.Interpreter

  Interpreter class based on the 'cmd' library.

Namespaces

- namespace lib.Interpreter

Variables

- tuple lib.Interpreter.obj = Interpreter()

8.19 Interpreter.py

```
00001 #!/usr/bin/env python
00002
00003 from cmd2 import Cmd
00004 import os
00005 from Interpreter_config import Interpreter_config
00006 from Interpreter_update import Interpreter_update
00007 from Interpreter_scan import Interpreter_scan
00008 from Interpreter_query import Interpreter_query
00009 from Interpreter_results import Interpreter_results
00010 from Baseconfig import Baseconfig
00011 from hephaestus_utils import ruler
00012 import inspect
00013
00014 class Interpreter(Cmd):
00015   """Interpreter class based on the 'cmd' library."""
00016
00017   def __init__(self):
00018     Cmd.__init__(self)
00019     self.prompt = "hephaestus> "
00020     self.intro = "\n" + ruler("*") + "WELCOME to hephaestus\n" + ruler("*") + "hephaestus is
       the stateless firewall configuration parser for JunOS.\nGet help with 'help (<cmd>)' or '?', or issue '
      userguide' to get started.\n" + ruler("*") + "\n"
00021     self.myconfig = Baseconfig()
00022     self.configinterpreter = Interpreter_config()
00023     self.configinterpreter.myconfig = self.myconfig
00024     self.scaninterpreter = Interpreter_scan()
00025     self.scaninterpreter.myconfig = self.myconfig
00026     self.updateinterpreter = Interpreter_update()
00027     self.updateinterpreter.myconfig = self.myconfig
00028     self.queryinterpreter = Interpreter_query()
00029     self.queryinterpreter.myconfig = self.myconfig
00030     self.resultsinterpreter = Interpreter_results()
00031     self.resultsinterpreter.myconfig = self.myconfig
00032
00033   def setconfig(self, cfg):
00034     self.myconfig = cfg
00035
00036   def emptyline(self):
00037     pass
00038
00039   def do_shell(self, line):
00040     "Run a shell command"
00041     print("\nRunning shell command: " + line + "\n")
00042     output = os.popen(line).read()
00043     print(output)
00044     self.last_output = output
00045
00046   def do_exit(self, line):
00047     return True
00048
00049   def help_exit(self):
00050     print("\nExit this program.\n")
00051
00052   def help_help(self):
00053     print("\nOverride worked...\n")
00054
00055   def help_shell(self):
00056     print("\nOverride worked...\n")
00057
00058   # interesting functions
00059   def do_userguide(self, line):
00060     """
00061
00062     A punchy, pithy user guide will go here soon. For now:
00063
00064     Basics:
00065
00066       1. Enter Configuration mode, load a file, show contents, exit
00067            config > load > show > exit
00068       2. Enter Scan mode, autoscan the network
00069            scan > autoscan
00070       3. Enter Update mode, update local cache of router configs
00071            update > update
00072       4. Enter Query mode, run query
00073            query > query
00074       5. Enter Results mode, view results
00075            results > summary
00076            results > diagram
00077            results > detail
00078     """
00079     self.help_userguide()
00080
00081   def help_userguide(self):
00082     """Print help on do_userguide() method"""
```

```
00083     print
00084     print(inspect.getdoc(self.do_userguide))
00085     print
00086
00087   def do_config(self, line):
00088     self.configinterpreter.cmdloop()
00089
00090   def help_config(self):
00091     print("\nBefore you can scan a network or run any queries, you must first enter configuration to enter
      the necessary AS/network information.\n")
00092
00093   def do_scan(self, line):
00094     self.scaninterpreter.cmdloop()
00095
00096   def help_scan(self):
00097     print("\nScan the configured network and identify any routers present.\n")
00098
00099   def do_update(self, line):
00100     self.updateinterpreter.cmdloop()
00101
00102   def help_update(self):
00103     print("\nFetch configuration from the identified routers.\n")
00104
00105   def do_query(self, line):
00106     self.queryinterpreter.cmdloop()
00107
00108   def help_query(self):
00109     print("\nRun a query against the network state to determine if a given flow will be accepted, rejected
      or discarded.\n")
00110
00111   def do_results(self, line):
00112     self.resultsinterpreter.cmdloop()
00113
00114   def help_results(self):
00115     print("\nView results in more detail.\n")
00116
00117 if __name__ == '__main__':
00118   print("\n" + ruler("*") + "\nSelf-test\n" + ruler("*") + "\nInstantiating self...")
00119   obj = Interpreter()
00120   print("\nInstantiation successful, " + str(obj))
00121   print("\nPrinting attributes...")
00122   print(dir(obj))
00123   print("\nPrinting docstring...")
00124   print(inspect.getdoc(obj))
00125   print("\n" + ruler("*") + "\n...done!\n")
00126
```

8.20 /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_config.py File Reference

Classes

- class lib.Interpreter_config.Interpreter_config

  Interpreter_config class based on the 'cmd' library.

Namespaces

- namespace lib.Interpreter_config

Variables

- tuple lib.Interpreter_config.obj = Interpreter_config()

8.21 Interpreter_config.py

```
00001 #!/usr/bin/env python
00002 from cmd2 import Cmd
00003 import os
00004 from Baseconfig import Baseconfig
00005 import configparser
00006 import inspect
```

```
00007 from hephaestus_utils import ruler
00008
00009 class Interpreter_config(Cmd):
00010   """  Interpreter_config class based on the 'cmd' library.  """
00011
00012   def __init__(self):
00013     """Constructor calls parent constructor, sets the interpreter prompt and welcome message"""
00014     Cmd.__init__(self)
00015     self.prompt = "hephaestus(config)# "
00016     self.intro = "\n" + ruler("*") + "\nCONFIGURATION MODE\n" + ruler("*") + "\nDifferent
      commands are available, get help with 'help' or '?'\n" + ruler("*") + "\n"
00017
00018   def emptyline(self):
00019     pass
00020
00021   def do_shell(self, line):
00022     """ Run shell commands from within this interpreter. """
00023     print("\nRunning shell command: " + line + "\n")
00024     output = os.popen(line).read()
00025     print(output)
00026     self.last_output = output
00027
00028   def do_exit(self, line):
00029     return True
00030
00031   def help_exit(self):
00032     print("\nExit this program.\n")
00033
00034   def do_unload(self, line):
00035     """ Unload the current configuration. """
00036     self.myconfig = Baseconfig()
00037     print("Configuration unloaded.")
00038
00039   def help_unload(self):
00040     print(inspect.getdoc(do_unload))
00041
00042   def do_load(self, line):
00043     print("\nThe following configuration files are available:\n")
00044     configs = next(os.walk("etc/"))[2]
00045
00046     for config in configs:
00047       if '.conf' not in config:
00048         configs.remove(config) # ignore files which aren't configuration files
00049
00050     for i, config in enumerate(configs):
00051       print(str(i) + " " + config)
00052     print
00053     file = raw_input("Enter the number of the file you wish to load:\n\n")
00054     self.myconfig.CONFIG = configparser.read_config("etc/" + configs[int(file)])
00055     self.myconfig.CONFIGpath = "etc/" + config
00056     print("\nConfiguration loaded successfully (use 'show' to verify contents).\n")
00057
00058   def help_load(self):
00059     print("View available configuration files and select one to load.")
00060
00061   def do_show(self, line):
00062     if self.myconfig.CONFIG != {}:
00063       print("Current configuration is:")
00064       for index, item in enumerate(self.myconfig.CONFIG):
00065         print(str(index) + " " +  str(item) + " " + self.myconfig.CONFIG[item])
00066     else:
00067       print("Current configuration is empty. Load a configuration file or use the wizard to create a new
      configuration.")
00068
00069   def help_show(self):
00070     print("Print the contents of the current configuration.")
00071
00072   def do_wizard(self, line):
00073     """ This function starts an interactive configuration wizard """
00074     def getConfig(self):
00075       """ Helper method for do_wizard() which handles user input and validation/sanitisation. """
00076       print("I'll now ask you for each config item, and then ask you to save the file.")
00077       self.myconfig.CONFIG['info'] = raw_input("Enter a pithy description of the network/autonomous system
      you're configuring:\n")
00078       self.myconfig.CONFIG['asn'] = raw_input("Enter this network/autonomous system's AS number:\n")
00079       self.myconfig.CONFIG['jumpboxsshuser'] = raw_input("Enter the SSH username for use with the jumpbox
      host:\n")
00080       self.myconfig.CONFIG['jumpboxsshkey'] = raw_input("Enter the absolute path to your jumpbox SSH
      private key, eg: '/home/someuser/.ssh/id_dsa':\n")
00081       self.myconfig.CONFIG['usejumpbox'] = raw_input("Use a jumpbox host to reach the network? eg: enter '
      true' if so, else 'false' to use localhost:\n")
00082       self.myconfig.CONFIG['jumpboxhost'] = raw_input("Enter jumpbox FQDN here. Could be 'localhost' if you
      entered 'false' above:\n")
00083       self.myconfig.CONFIG['routersshuser'] = raw_input("Enter the username to use when logging into
      routers via SSH:\n")
00084       self.myconfig.CONFIG['routerprefixes'] = raw_input("Enter a space-separated list of CIDR prefixes
      containing your routers, eg: '62.40.97.0/28 62.40.98.0/28':\n")
```

```
00085       self.myconfig.CONFIG['netname'] = raw_input("Enter a short name for the network, eg: 'foonet':\n")
00086       self.myconfig.CONFIG['routersshkey'] = raw_input("Enter the absolute path to your router SSH private
    key, eg: '/home/someuser/.ssh/id_dsa':\n")
00087       self.myconfig.CONFIG['routersnmpcommunity'] = raw_input("Enter the read-only SNMP community string
    for routers, eg: '0pBiFbD':\n")
00088       print("Thank you, wizard completed! Don't forget to save the file, type 'save'[return]")
00089
00090     print("This wizard will generate a new configuration for you.")
00091
00092     if self.myconfig.CONFIG != {}:
00093       self.do_show(self)
00094       fromScratch = raw_input("Do you want to start from scratch? (y/n):\n")
00095       if fromScratch == 'y':
00096         self.myconfig.CONFIG.clear()
00097         print("Configuration cleared.")
00098         print("Now, please enter the required information. If you make a mistake, run this wizard again.")
00099         getConfig(self)
00100       elif fromScratch == 'n':
00101         changeItem = raw_input("Enter the exact item name you wish to change, for example 'netname'
    (without quotes), or press 'd' when done:\n")
00102         if changeItem != 'd':
00103           newValue = raw_input("Now enter the new value:\n")
00104           self.myconfig.CONFIG[changeItem] = newValue
00105           print("Value changed.")
00106       else:
00107         print("Illegal choice, start again!")
00108     else:
00109       print("Current configuration is empty, so we will start from scratch.")
00110       getConfig(self)
00111
00112   def help_wizard(self):
00113     print("Generate a new configuration.")
00114
00115   def do_save(self, line):
00116     validconfig = False
00117     validpath = False
00118
00119     # Check this is a sensible operation...
00120     if self.myconfig.CONFIG != {}:
00121       validconfig = True
00122     if self.myconfig.CONFIGpath != '':
00123       validpath = True
00124
00125     # If we have valid configuration, ask where to save
00126     if validconfig and validpath:
00127       overwrite = raw_input("Overwrite current file (y/n):\n")
00128       if overwrite == 'y':
00129         configparser.write_config(self.myconfig.CONFIGpath, self.myconfig.CONFIG)
00130         return
00131       elif overwrite == 'n':
00132         filename = raw_input("Enter new filename:")
00133     path = "etc/" + filename
00134         self.myconfig.CONFIGpath = path
00135         configparser.write_config(self.myconfig.CONFIGpath, self.myconfig.CONFIG)
00136         return
00137     elif validconfig and not validpath:
00138       filename = raw_input("Enter new filename:")
00139       path = "etc/" + filename
00140       self.myconfig.CONFIGpath = path
00141       configparser.write_config(self.myconfig.CONFIGpath, self.myconfig.CONFIG)
00142     elif not validconfig:
00143       print("Cannot save an empty configuration.")
00144       return
00145     else:
00146       print("Something went wrong here, perhaps your configuration or path is somehow invalid.")
00147       return
00148
00149   def help_save(self):
00150     print("Save current configuration to disk.")
00151
00152 if __name__ == '__main__':
00153   print("\n" + ruler("*") + "\nSelf-test\n" + ruler("*") + "\nInstantiating self...")
00154   obj = Interpreter_config()
00155   print("\nInstantiation successful, " + str(obj))
00156   print("\nPrinting attributes...")
00157   print(dir(obj))
00158   print("\nPrinting docstring...")
00159   print(inspect.getdoc(obj))
00160   print("\n" + ruler("*") + "\n...done!\n")
```

8.22 /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_query.py File Reference

Classes

- class lib.Interpreter_query.Interpreter_query

  Interpreter_query class based on the 'cmd' library.

Namespaces

- namespace lib.Interpreter_query

Variables

- tuple lib.Interpreter_query.obj = Interpreter_query()

8.23 Interpreter_query.py

```
00001 #!/usr/bin/env python
00002 import ipaddr
00003 from cmd2 import Cmd
00004 import os
00005 import paramiko
00006 import socket
00007 import futures as concurrentfutures
00008 import time
00009 import random
00010 import inspect
00011 from hephaestus_utils import ruler
00012 from subprocess import check_output
00013
00014 class Interpreter_query(Cmd):
00015   """Interpreter_query class based on the 'cmd' library. """
00016
00017   def __init__(self):
00018     Cmd.__init__(self)
00019     self.prompt = "hephaestus(query)# "
00020     self.intro = "You are now in query mode, different commands are available.\nCommand help with '
      help' or '?'\n"
00021
00022   def emptyline(self):
00023     pass
00024
00025   def do_shell(self, line):
00026     "Run a shell command"
00027     print("\nRunning shell command: " + line + "\n")
00028     output = os.popen(line).read()
00029     print(output)
00030     self.last_output = output
00031
00032   def do_exit(self, line):
00033     return True
00034
00035   def help_exit(self):
00036     print("\nExit this program.\n")
00037
00038   def do_query(self, line):
00039     query = {}
00040     addressfamily = raw_input("Enter inet|4 for IPv4, inet6|6 for IPv6, or just press enter for IPv4
      (inet):\n")
00041     if addressfamily == '4' or 'inet':
00042       query['addressfamily'] = 'inet'
00043     elif addressfamily == '6' or 'inet6':
00044       query['addressfamily'] = 'inet6'
00045     elif addressfamily == '\n':
00046       print("inet selected")
00047       query['addressfamily'] = 'inet'
00048     else:
00049       print("Illegal value entered, defaulting to inet/IPv4...")
00050       query['addressfamily'] = 'inet'
00051
00052
00053     query['srcip'] = raw_input("Enter SRC IP:\n")
00054     srcport = raw_input("Enter SRC port (single number, or for a random ephemeral port, 'e' for the IANA
      range, or 'l' for Linux):\n")
00055     if srcport == 'e':
00056       # Choose a random ephemeral port number in IANA range 49152-65535
00057       query['srcport'] = random.randint(49152, 65535)
```

```
00058     elif srcport == 'l':
00059       # Choose a random ephemeral port number in the complement set of Linux range (32768-65535) and IANA
      range
00060       query['srcport'] = random.randint(32768, 49151)
00061     else:
00062       # Else use the raw input (useful for inputting non-compliant emphemeral ports, eg: early Windows, BSD
      < 4.6)
00063       query['srcport'] = srcport
00064
00065     query['dstip'] = raw_input("Enter DST IP:\n")
00066
00067     dstport = raw_input("Enter DST port (single number, or for a random ephemeral port, 'e' for the IANA
      range, or 'l' for Linux):\n")
00068     if dstport == 'e':
00069       # Choose a random ephemeral port number in IANA range 49152-65535
00070       query['dstport'] = random.randint(49152, 65535)
00071     elif dstport == 'l':
00072       # Choose a random ephemeral port number in the complement set of Linux range (32768-65535) and IANA
      range
00073       query['dstport'] = random.randint(32768, 49151)
00074     else:
00075       # Else use the raw input (useful for inputting non-compliant emphemeral ports, eg: early Windows, BSD
      < 4.6)
00076       query['dstport'] = dstport
00077
00078     proto = raw_input("Enter transport protocol (tcp|udp|any or press 'enter' for any)?:\n")
00079     if proto == 'tcp' or proto == 'udp' or proto == 'any':
00080       query['proto'] = proto
00081     elif proto == '':
00082       query['proto'] = 'any'
00083     else:
00084       print("Illegal value entered, using 'any'...")
00085       query['proto'] = 'any'
00086
00087     self.myconfig.query = query
00088     print("Query entered: " + str(self.myconfig.query['srcip']) + ":" + str(self.myconfig.query['srcport'])
      + "/" + str(self.myconfig.query['proto']) + " <=> " + str(self.myconfig.query['dstip']) + ":" + str(
     self.myconfig.query['dstport']) + "/" + str(self.myconfig.query['proto']))
00089     proceed = raw_input("Is this query correct (y/n)?:")
00090
00091     if proceed == 'y':
00092       pass
00093     elif proceed == 'n':
00094       print("\n***Aborting query***\n")
00095       return # abort query
00096     else:
00097       print("\n***Incorrect response, aborting query!***\n")
00098       return # abort query
00099
00100     for i, rtr in enumerate(self.myconfig.CONFIG['routerloopbackips']):
00101       fqdn = check_output(["host", rtr]).split()[4]
00102       print(str(i) + "\t: " + fqdn[4:-1])
00103     choice = raw_input("Now I need to know which is the ingress router (select by number)?:")
00104     self.myconfig.query['ingressrouterip'] = self.myconfig.CONFIG['routerloopbackips'][int(choice)]
00105     self.myconfig.query['ingressrouterfqdn'] = check_output(["host", self.myconfig.query['ingressrouterip']
     ]).split()[4][4:-1]
00106     print("Selected router IP: " + self.myconfig.query['ingressrouterip'] + " (" + self.myconfig.query['
     ingressrouterfqdn'] + ")")
00107
00108     choice = raw_input("Now I need to know which is the egress router (select by number)?:")
00109     self.myconfig.query['egressrouterip'] = self.myconfig.CONFIG['routerloopbackips'][int(choice)]
00110     self.myconfig.query['egressrouterfqdn'] = check_output(["host", self.myconfig.query['egressrouterip']])
     .split()[4][4:-1]
00111     print("Selected router IP: " + self.myconfig.query['egressrouterip'] + " (" + self.myconfig.query['
     egressrouterfqdn'] + ")")
00112
00113     print("Now determining active SRC=>DST path through AS" + self.myconfig.CONFIG['asn'] + " via
      traceroute from ingress router...")
00114     client = paramiko.SSHClient()
00115     client.load_system_host_keys()
00116     key = paramiko.DSSKey.from_private_key_file(self.myconfig.CONFIG['jumpboxsshkey'])
00117     print(self.myconfig.query['ingressrouterip'] + '\t: connecting...')
00118     client.connect(self.myconfig.CONFIG['jumpboxhost'], username=self.myconfig.CONFIG['jumpboxsshuser'],
     pkey=key)
00119     command = 'ssh -q  -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no ' + self.myconfig.CONFIG
     ['routersshuser'] + '@' + self.myconfig.query['ingressrouterip'] + ' "traceroute no-resolve ' +
     self.myconfig.query['addressfamily'] + ' ' + self.myconfig.query['dstip'] + '"'
00120     print(self.myconfig.query['ingressrouterip'] + '\t: executing `' + command + '`')
00121     stdin, stdout, stderr = client.exec_command(str(command))
00122     print(self.myconfig.query['ingressrouterip'] + '\t: waiting...')
00123     output = stdout.read()
00124     src_to_dst = output.split("\n")
00125
00126     #debug
00127     print("\nraw traceroute:")
00128     for hop in src_to_dst:
00129       print(hop)
```

```
00130
00131     hops = []
00132     for hop in src_to_dst:
00133       if "traceroute" in hop:
00134         #hops.remove(hop)
00135         pass
00136       elif "* * *" in hop:
00137         #hops.remove(hop)
00138         pass
00139         print("Traceroute failed for hop " + hop.split("  ")[0] + ", excluding...")
00140       else:
00141         try:
00142           ip = hop.split("  ")[1]
00143           #print("Checking AS for IP\t: " + ip)
00144           asn = check_output(["whois", "-h", "whois.cymru.com", ip])
00145           asn = asn.split("\n")[1]
00146           asn = asn.split(" ")[0]
00147           if asn == self.myconfig.CONFIG['asn']:
00148             # If the last hop is the DST IP, ignore - it is not a router with filters
00149             if self.myconfig.query['dstip'] in hop:
00150               src_to_dst.remove(hop)
00151             else:
00152               print(ip + " is within our AS, filters will be evaluated")
00153               hops.append(ip)
00154           else:
00155             print(ip + " is not within our AS, skipping")
00156         except:
00157           pass # ignore blank line
00158
00159     src_to_dst = hops
00160
00161     print("\nDetailed information about each hop in our AS:\n")
00162     print("\nIngress\n=======")
00163     print("Hop looback IP\t : " + self.myconfig.query['ingressrouterip'])
00164     client = paramiko.SSHClient()
00165     client.load_system_host_keys()
00166     key = paramiko.DSSKey.from_private_key_file(self.myconfig.CONFIG['jumpboxsshkey'])
00167     print(self.myconfig.query['ingressrouterip'] + '\t : connecting...')
00168     client.connect(self.myconfig.CONFIG['jumpboxhost'], username=self.myconfig.CONFIG['jumpboxsshuser'],
      pkey=key)
00169     command = 'ssh -q  -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no ' + self.myconfig.CONFIG
      ['routersshuser'] + '@' + self.myconfig.query['ingressrouterip'] + ' "show route table inet.0 ' +
      self.myconfig.query['srcip'] + ' | match via"'
00170     print(self.myconfig.query['ingressrouterip'] + '\t : executing `' + command + '`')
00171     stdin, stdout, stderr = client.exec_command(str(command))
00172     output = stdout.read()
00173     interface = output.split()[2].split(".")[0]
00174     unit = output.split()[2].split(".")[1]
00175     print("Hop interface\t : " + interface + " unit " + unit)
00176     self.myconfig.query['ingressrouterinterface'] = interface
00177     self.myconfig.query['ingressrouterinterfaceunit'] = unit
00178     command = 'ssh -q  -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no ' + self.myconfig.CONFIG
      ['routersshuser'] + '@' + self.myconfig.query['ingressrouterip'] + ' "show configuration interfaces ' +
      interface + ' unit ' + unit + ' family inet filter"'
00179     print(self.myconfig.query['ingressrouterip'] + '\t : executing `' + command + '`')
00180     stdin, stdout, stderr = client.exec_command(str(command))
00181     print(self.myconfig.query['ingressrouterip'] + '\t : waiting...')
00182     output = stdout.read()
00183     inputfilter = output.split("\n")[0].split(" ")[1][:-1]
00184     outputfilter = output.split("\n")[1].split(" ")[1][:-1]
00185     print("Input filter\t : " + inputfilter)
00186     print("Output filter\t : " + outputfilter)
00187     self.myconfig.query['ingressrouterinterfaceinputfilter'] = inputfilter
00188     self.myconfig.query['ingressrouterinterfaceoutputfilter'] = outputfilter
00189
00190     self.myconfig.src_to_dst_hops = []
00191     for hop in src_to_dst:
00192       print("\nHop\n===")
00193       print("Hop interface IP : " + str(hop))
00194       client = paramiko.SSHClient()
00195       client.load_system_host_keys()
00196       key = paramiko.DSSKey.from_private_key_file(self.myconfig.CONFIG['jumpboxsshkey'])
00197       print(hop + '\t : connecting...')
00198       client.connect(self.myconfig.CONFIG['jumpboxhost'], username=self.myconfig.CONFIG['jumpboxsshuser'],
      pkey=key)
00199       command = 'ssh -q  -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no ' +
      self.myconfig.CONFIG['routerssshuser'] + '@' + hop + ' "show configuration interfaces lo0 unit 0 family inet | grep addre
00200       print(hop + '\t : executing `' + command + '`')
00201       stdin, stdout, stderr = client.exec_command(str(command))
00202       print(hop + '\t : waiting...')
00203       output = stdout.read()
00204       loopback = output.split(" ")[1].split("/")[0]
00205       print("Hop loopback IP\t : " + loopback)
00206       command = 'ssh -q  -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no ' +
      self.myconfig.CONFIG['routerssshuser'] + '@' + loopback + ' "show configuration interfaces | display set | match ' + hop
00207       print(loopback + '\t : executing `' + command + '`')
00208       stdin, stdout, stderr = client.exec_command(str(command))
```

```
00209       print(loopback + '\t : waiting...')
00210       output = stdout.read()
00211       interface = output.split(" ")[2]
00212       unit = output.split(" ")[4]
00213       print("Hop interface\t : " + interface + " unit " + unit)
00214       command = 'ssh -q  -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no ' +
      self.myconfig.CONFIG['routersshuser'] + '@' + loopback + ' "show configuration interfaces ' + interface + ' unit ' + uni
       family inet filter"'
00215       print(loopback + '\t : executing `' + command + '`')
00216       stdin, stdout, stderr = client.exec_command(str(command))
00217       print(loopback + '\t : waiting...')
00218       output = stdout.read()
00219       inputfilter = output.split("\n")[0].split(" ")[1][:-1]
00220       outputfilter = output.split("\n")[1].split(" ")[1][:-1]
00221       print("Input filter\t : " + inputfilter)
00222       print("Output filter\t : " + outputfilter)
00223       self.myconfig.src_to_dst_hops.append((hop, interface, unit, inputfilter, outputfilter))
00224     print
00225
00226     print("\nEgress\n======")
00227     print("Hop looback IP\t : " + self.myconfig.query['egressrouterip'])
00228     client = paramiko.SSHClient()
00229     client.load_system_host_keys()
00230     key = paramiko.DSSKey.from_private_key_file(self.myconfig.CONFIG['jumpboxsshkey'])
00231     print(self.myconfig.query['egressrouterip'] + '\t : connecting...')
00232     client.connect(self.myconfig.CONFIG['jumpboxhost'], username=self.myconfig.CONFIG['jumpboxsshuser'],
      pkey=key)
00233     command = 'ssh -q  -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no ' + self.myconfig.CONFIG
      ['routersshuser'] + '@' + self.myconfig.query['egressrouterip'] + ' "show route table inet.0 ' +
      self.myconfig.query['dstip'] + ' | match via"'
00234     print(self.myconfig.query['egressrouterip'] + '\t : executing `' + command + '`')
00235     stdin, stdout, stderr = client.exec_command(str(command))
00236     output = stdout.read()
00237     interface = output.split()[2].split(".")[0]
00238     unit = output.split()[2].split(".")[1]
00239     print("Hop interface\t : " + interface + " unit " + unit)
00240     self.myconfig.query['egressrouterinterface'] = interface
00241     self.myconfig.query['egressrouterinterfaceunit'] = unit
00242     command = 'ssh -q  -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no ' + self.myconfig.CONFIG
      ['routersshuser'] + '@' + self.myconfig.query['egressrouterip'] + ' "show configuration interfaces ' +
      interface + ' unit ' + unit + ' family inet filter"'
00243     print(self.myconfig.query['egressrouterip'] + '\t : executing `' + command + '`')
00244     stdin, stdout, stderr = client.exec_command(str(command))
00245     print(self.myconfig.query['egressrouterip'] + '\t : waiting...')
00246     output = stdout.read()
00247     inputfilter = output.split("\n")[0].split(" ")[1][:-1]
00248     outputfilter = output.split("\n")[1].split(" ")[1][:-1]
00249     print("Input filter\t : " + inputfilter)
00250     print("Output filter\t : " + outputfilter)
00251     self.myconfig.query['egressrouterinterfaceinputfilter'] = inputfilter
00252     self.myconfig.query['egressrouterinterfaceoutputfilter'] = outputfilter
00253     print
00254
00255   def help_query(self):
00256     print("\nRun a query against the network state to determine if a given flow will be accepted, rejected
       or discarded en route.\n")
00257
00258
00259 if __name__ == '__main__':
00260   print("\n" + ruler("*") + "\nSelf-test\n" + ruler("*") + "\nInstantiating self...")
00261   obj = Interpreter_query()
00262   print("\nInstantiation successful, " + str(obj))
00263   print("\nPrinting attributes...")
00264   print(dir(obj))
00265   print("\nPrinting docstring...")
00266   print(inspect.getdoc(obj))
00267   print("\n" + ruler("*") + "\n...done!\n")
```

8.24 /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_results.py File Reference

Classes

- class lib.Interpreter_results.Interpreter_results

  Interpreter_results class based on the 'cmd' library.

Namespaces

- namespace lib.Interpreter_results

Variables

- tuple lib.Interpreter_results.obj = Interpreter_results()

8.25 Interpreter_results.py

```
00001 #!/usr/bin/env python
00002 from cmd2 import Cmd
00003 import os
00004 import paramiko
00005 import socket
00006 import ipaddr
00007 import inspect
00008 from hephaestus_utils import ruler
00009 import futures as concurrentfutures
00010 import time
00011 from subprocess import check_output
00012 import subprocess
00013 import pygraphviz as pgv
00014 import warnings
00015
00016 class Interpreter_results(Cmd):
00017   """
00018   Interpreter_results class based on the 'cmd' library.
00019   """
00020
00021   def __init__(self):
00022     Cmd.__init__(self)
00023     self.prompt = "hephaestus(results)> "
00024     self.intro = "You are now in results mode, different commands are available.\nCommand help with '
      help' or '?'\n"
00025
00026   def emptyline(self):
00027     pass
00028
00029   def do_shell(self, line):
00030     """Run a shell command"""
00031     print("\nRunning shell command: " + line + "\n")
00032     output = os.popen(line).read()
00033     print(output)
00034     self.last_output = output
00035
00036   def do_exit(self, line):
00037     return True
00038
00039   def help_exit(self):
00040     print("\nExit this program.\n")
00041
00042   def do_summary(self, line):
00043     """Print results summary."""
00044     if self.myconfig.CONFIG == {}:
00045       print("Current configuration is empty. First enter configuration mode and load a file or run the
      wizard.")
00046     else:
00047       print
00048       print(ruler("*"))
00049       print("SUMMARY")
00050       print(ruler("*"))
00051
00052   def help_summary(self):
00053     print("\nPrint a summary of the results.\n")
00054
00055   def do_detail(self, line):
00056     print("\nDetailed results here...\n")
00057
00058   def help_detail(self, line):
00059     print("\nPrint detailed results.\n")
00060
00061   def do_diagram(self, line):
00062     print("\nGenerating diagrams, please be patient.\nThis process is heavily network I/O bound and may
      take upto 3 minutes to complete...\n")
00063     print("[" + time.strftime("%H:%M:%S") + "]\t: Starting process...")
00064     self.query_diag()
00065     print("[" + time.strftime("%H:%M:%S") + "]\t: Query diagram generated, displaying now...")
00066     self.do_shell('xdg-open etc/querydiag/query.png > /dev/null 2>&1')
00067     proc = subprocess.Popen(['python', 'lib/isisdiag.py'], stdout=subprocess.PIPE, stderr=subprocess.PIPE)
00068     while proc.poll() is None:
00069       time.sleep(15)
00070       print("[" + time.strftime("%H:%M:%S") + "]\t: Process " + str(proc.pid) + " is still running, please
      wait...")
00071     print("[" + time.strftime("%H:%M:%S") + "]\t: Full IS-IS diagram generated, displaying now...")
00072     self.do_shell('xdg-open etc/isisdiag/geantISIS.png > /dev/null 2>&1')
```

```
00073
00074   def query_diag(self):
00075     G=pgv.AGraph(name='hephaestusquery', directed=True)
00076     G.node_attr['shape']='circle'
00077     G.node_attr['fixedsize']='true'
00078     G.node_attr['fontsize']='14'
00079     G.node_attr['style']='filled'
00080     G.node_attr['color']='#40e0d0'
00081     G.graph_attr['outputorder']='edgesfirst'
00082     G.graph_attr['label']='hephaestus: query ran on ' + time.strftime("%d/%m/%Y") + ' @ ' + time.strftime("
      %H:%M:%S")
00083     G.graph_attr['fontsize']='30'
00084     G.graph_attr['ratio']='1.0'
00085     G.edge_attr['color']='#AA00FF'
00086     G.edge_attr['style']='setlinewidth(5)'
00087     G.graph_attr['splines']='true'
00088
00089     x=500
00090     y=500
00091
00092     G.add_node(self.myconfig.query['ingressrouterip'])
00093     n=G.get_node(self.myconfig.query['ingressrouterip'])
00094     n.attr['pos']="%f,%f)"%(float(x)/7.0,float(y)/7.0)
00095     # assign node size
00096     d=2.5
00097     n.attr['height']="%s"%d
00098     n.attr['width']="%s"%d
00099     # assign node color
00100     n.attr['fillcolor']="#00FF0C"
00101     # label
00102     n.attr['label']=self.myconfig.query['ingressrouterip']
00103
00104     #self.myconfig.src_to_dst_hops.append((hop, interface, unit, inputfilter, outputfilter))
00105     for hop in self.myconfig.src_to_dst_hops:
00106       ip = hop[0]
00107       x+=4500
00108       G.add_node(ip)
00109       n=G.get_node(ip)
00110       n.attr['pos']="%f,%f)"%(float(x)/7.0,float(y)/7.0)
00111       # assign node size
00112       d=2.5
00113       n.attr['height']="%s"%d
00114       n.attr['width']="%s"%d
00115       # assign node color
00116       n.attr['fillcolor']="#00FF0C"
00117       # label
00118       n.attr['label']=ip
00119
00120     x+=4500
00121     G.add_node(self.myconfig.query['egressrouterip'])
00122     n=G.get_node(self.myconfig.query['egressrouterip'])
00123     n.attr['pos']="%f,%f)"%(float(x)/7.0,float(y)/7.0)
00124     # assign node size
00125     d=2.5
00126     n.attr['height']="%s"%d
00127     n.attr['width']="%s"%d
00128     # assign node color
00129     n.attr['fillcolor']="#00FF0C"
00130     # label
00131     n.attr['label']=self.myconfig.query['egressrouterip']
00132
00133     # add edges
00134     self.myconfig.src_to_dst_hops.reverse()
00135     lastpop = self.myconfig.src_to_dst_hops.pop()
00136     ip = lastpop[0]
00137     G.add_edge(self.myconfig.query['ingressrouterip'],ip,arrowsize=3,fontname='Courier',fontsize='20',label
      =self.myconfig.query['ingressrouterinterfaceoutputfilter'] + " | " + lastpop[3])
00138     while len(self.myconfig.src_to_dst_hops) > 0:
00139       firstnode = lastpop[0]
00140       firstnodeoutputfilter = lastpop[4]
00141       lastpop = self.myconfig.src_to_dst_hops.pop()
00142       ip = lastpop[0]
00143       G.add_edge(firstnode,ip,arrowsize=3,fontname='Courier',fontsize='20',label=firstnodeoutputfilter + "
       | " + lastpop[3])
00144
00145     G.add_edge(lastpop[0],self.myconfig.query['egressrouterip'],arrowsize=3,fontname='Courier',fontsize='20
      ',label=lastpop[4] + " | " + self.myconfig.query['egressrouterinterfaceinputfilter'])
00146     # ignore Graphviz warning messages
00147     #warnings.simplefilter('ignore', RuntimeWarning)
00148     G.string()
00149     G.write("etc/querydiag/query.dot")
00150     print("Wrote DOT language output file: ./etc/querydiag/query.dot")
00151     G.draw("etc/querydiag/query.png",prog='neato',args='-n2')
00152     print("Wrote final diagram output file: ./etc/querydiag/query.png")
00153     print("Done!")
00154
00155   def help_diagram(self, line):
```

```
00156     print("\nGenerate and display network diagrams,\n\ti)\tA full network diagram showing IS-IS adjacencies
      and costs, and\n\tii)\tA detailed view of the query's active path\n")
00157
00158 if __name__ == '__main__':
00159   print("\n" + ruler("*") + "\nSelf-test\n" + ruler("*") + "\nInstantiating self...")
00160   obj = Interpreter_results()
00161   print("\nInstantiation successful, " + str(obj))
00162   print("\nPrinting attributes...")
00163   print(dir(obj))
00164   print("\nPrinting docstring...")
00165   print(inspect.getdoc(obj))
00166   print("\n" + ruler("*") + "\n...done!\n")
00167
```

8.26 /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_scan.py File Reference

Classes

- class lib.Interpreter_scan.Interpreter_scan

  Interpreter_scan class based on the 'cmd' library.

Namespaces

- namespace lib.Interpreter_scan

Variables

- tuple lib.Interpreter_scan.obj = Interpreter_scan()

8.27 Interpreter_scan.py

```
00001 #!/usr/bin/env python
00002 from cmd2 import Cmd
00003 import os
00004 import paramiko
00005 import socket
00006 import ipaddr
00007 import inspect
00008 from hephaestus_utils import ruler
00009 import futures as concurrentfutures
00010 import time
00011 from subprocess import check_output
00012
00013 class Interpreter_scan(Cmd):
00014   """
00015   Interpreter_scan class based on the 'cmd' library.
00016   """
00017
00018   def __init__(self):
00019     Cmd.__init__(self)
00020     self.prompt = "hephaestus(scan)> "
00021     self.intro = "You are now in scan mode, different commands are available.\nCommand help with 'help
      ' or '?'\n"
00022
00023   def emptyline(self):
00024     pass
00025
00026   def do_shell(self, line):
00027     """Run a shell command"""
00028     print("\nRunning shell command: " + line + "\n")
00029     output = os.popen(line).read()
00030     print(output)
00031     self.last_output = output
00032
00033   def do_exit(self, line):
00034     return True
00035
00036   def help_exit(self):
00037     print("\nExit this program.\n")
00038
```

```
00039   def do_autoscan(self, line):
00040     """Automatically scan the network and identify any routers present."""
00041     if self.myconfig.CONFIG == {}:
00042       print("Current configuration is empty. First enter configuration mode and load a file or run the
      wizard.")
00043     else:
00044       print
00045       print(ruler("*"))
00046       print("NETWORK SCAN")
00047       print(ruler("*"))
00048       print("Using current configuration, \"" + self.myconfig.CONFIG['info']) + "\""
00049       print("Scanning for routers in prefixes " + self.myconfig.CONFIG['routerprefixes'])
00050       print(ruler("*"))
00051       print
00052       prefixes = self.myconfig.CONFIG['routerprefixes'].split()
00053       self.myconfig.CONFIG['routerips'] = []
00054
00055       for prefix in prefixes:
00056         network = ipaddr.IPNetwork(prefix)
00057         for ip in network.iterhosts():
00058           self.myconfig.CONFIG['routerips'].append(str(ip))
00059
00060       numscanned = len(self.myconfig.CONFIG['routerips'])
00061
00062       def autoscanThread(ip):
00063         client = paramiko.SSHClient()
00064         client.load_system_host_keys()
00065         key = paramiko.DSSKey.from_private_key_file(self.myconfig.CONFIG['jumpboxsshkey'])
00066         print(ip + '\t: connecting...')
00067         client.connect(self.myconfig.CONFIG['jumpboxhost'], username=self.myconfig.CONFIG['jumpboxsshuser']
      , pkey=key)
00068         command = 'snmpget -v 2c -c ' + self.myconfig.CONFIG['routersnmpcommunity'] + ' ' + str(ip) + '
       1.3.6.1.4.1.2636.3.1.2.0'
00069         print(ip + '\t: executing ' + "`" + command + "`")
00070         stdin, stdout, stderr = client.exec_command(str(command))
00071         print(ip + '\t: waiting...')
00072         output = stdout.read()
00073         if "Router" in output:
00074           print(ip + '\t: is a router (' + output.split('"')[1] + ')')
00075         else:
00076           print(ip + '\t: is not a router')
00077           # prune from list
00078           self.myconfig.CONFIG['routerips'].remove(ip)
00079
00080       executor = concurrentfutures.ThreadPoolExecutor(10)
00081       start = time.time()
00082       futures = [executor.submit(autoscanThread, ip) for ip in self.myconfig.CONFIG['routerips']]
00083       concurrentfutures.wait(futures)
00084       end = time.time()
00085
00086       numfound = len(self.myconfig.CONFIG['routerips'])
00087
00088       print
00089       print('Scanned\t\t: ' + str(numscanned) + ' IPs')
00090       print('Found\t\t: ' + str(numfound) + ' routers')
00091       diff = end - start
00092       print('Wall time\t: ' + "%.2f" % diff + ' seconds')
00093       print
00094       print("...done!")
00095       print
00096
00097       # Now we've found which IPs belong to routers, we need to determine the unique loopback IPs, ignoring
      physical interface IPs
00098       self.myconfig.CONFIG['routerloopbackips'] = []
00099
00100       # Tell user what we're doing...
00101       print
00102       print('Now determining unique loopback interface IPs using these physical interface IPs...')
00103       print
00104
00105       def autoscanCleanupThread(ip):
00106         client = paramiko.SSHClient()
00107         client.load_system_host_keys()
00108         key = paramiko.DSSKey.from_private_key_file(self.myconfig.CONFIG['jumpboxsshkey'])
00109         print(ip + '\t: connecting...')
00110         client.connect(self.myconfig.CONFIG['jumpboxhost'], username=self.myconfig.CONFIG['jumpboxsshuser']
      , pkey=key)
00111       command = 'ssh ' + self.myconfig.CONFIG['routersshuser'] + '@' + ip + ' -q -o
       UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no "show configuration interfaces lo0 unit 0 family inet | grep a
00112         print(ip + '\t: executing ' + "`" + command + "`")
00113         stdin, stdout, stderr = client.exec_command(str(command))
00114         print(ip + '\t: waiting...')
00115         output = stdout.read()
00116         if "address" in output:
00117           loopback = output.split()[1].split('/')[0]
00118           fqdn = check_output(["host", loopback]).split()[4]
00119           print(ip + '\t: is a physical interface on ' + fqdn[:-1] + ' with loopback address ' + loopback)
```

```
00120           self.myconfig.CONFIG['routerloopbackips'].append(str(loopback))
00121         else:
00122           print(ip + '\t: unknown response: ' + output + ' (ignoring...possible phy->lo0 firewall issue)')
00123 
00124       start = time.time()
00125       futures = [executor.submit(autoscanCleanupThread, ip) for ip in self.myconfig.CONFIG['routerips']]
00126       concurrentfutures.wait(futures)
00127       end = time.time()
00128 
00129       numscanned = len(self.myconfig.CONFIG['routerips'])
00130       foolist = self.myconfig.CONFIG['routerloopbackips']
00131       self.myconfig.CONFIG['routerloopbackips'] = list(set(foolist))
00132       numfound = len(self.myconfig.CONFIG['routerloopbackips'])
00133 
00134       print
00135       print('Scanned\t\t: ' + str(numscanned) + ' physical interface IPs')
00136       print('Found\t\t: ' + str(numfound) + ' loopback IPs')
00137       diff = end - start
00138       print('Wall time\t: ' + "%.2f" % diff + ' seconds')
00139 
00140   def help_autoscan(self):
00141     print(inspect.getdoc(do_autoscan))
00142 
00143 if __name__ == '__main__':
00144   print("\n" + ruler("*") + "\nSelf-test\n" + ruler("*") + "\nInstantiating self...")
00145   obj = Interpreter_scan()
00146   print("\nInstantiation successful, " + str(obj))
00147   print("\nPrinting attributes...")
00148   print(dir(obj))
00149   print("\nPrinting docstring...")
00150   print(inspect.getdoc(obj))
00151   print("\n" + ruler("*") + "\n...done!\n")
00152 
```

8.28 /home/niall/Code/hephaestus/hephaestus/lib/Interpreter_update.py File Reference

Classes

- class lib.Interpreter_update.Interpreter_update

  Interpreter_update class based on the 'cmd' library.

Namespaces

- namespace lib.Interpreter_update

Variables

- tuple lib.Interpreter_update.obj = Interpreter_update()

8.29 Interpreter_update.py

```
00001 #!/usr/bin/env python
00002 from cmd2 import Cmd
00003 import os
00004 import paramiko
00005 import socket
00006 import futures as concurrentfutures
00007 import time
00008 import xml2datacustom
00009 import inspect
00010 from hephaestus_utils import ruler
00011 
00012 class Interpreter_update(Cmd):
00013   """
00014   Interpreter_update class based on the 'cmd' library.
00015   """
00016 
00017   def __init__(self):
00018     Cmd.__init__(self)
00019     self.prompt = "hephaestus(update)> "
```

```
00020     self.intro = "You are now in update mode, different commands are available.\nCommand help with '
      help' or '?'\n"
00021
00022   def emptyline(self):
00023     pass
00024
00025   def do_shell(self, line):
00026     "Run a shell command"
00027     print("\nRunning shell command: " + line + "\n")
00028     output = os.popen(line).read()
00029     print(output)
00030     self.last_output = output
00031
00032   def do_exit(self, line):
00033     return True
00034
00035   def help_exit(self):
00036     print("\nExit this program.\n")
00037
00038   def do_update(self, line):
00039     if self.myconfig.CONFIG['routerloopbackips'] == []:
00040       print("Current configuration is empty. First enter configuration mode and load a file or run the
      wizard.")
00041     else:
00042       print("Using current configuration, \"" + self.myconfig.CONFIG['info']) + "\""
00043       numtoupdate = str(len(self.myconfig.CONFIG['routerloopbackips']))
00044       print("Fetching configuration from " + numtoupdate + " routers...")
00045
00046       def updateThread(ip):
00047         client = paramiko.SSHClient()
00048         client.load_system_host_keys()
00049         key = paramiko.DSSKey.from_private_key_file(self.myconfig.CONFIG['jumpboxsshkey'])
00050         print(ip + '\t: connecting...')
00051         client.connect(self.myconfig.CONFIG['jumpboxhost'], username=self.myconfig.CONFIG['jumpboxsshuser']
      , pkey=key)
00052         command = 'ssh ' + self.myconfig.CONFIG['routersshuser'] + '@' + ip + ' "show configuration |
      display xml | except \\"rpc-reply|\<*cli\>|\<banner\>\\" | no-more"'
00053         print(ip + '\t: executing ' + '`' + command + '`')
00054         stdin, stdout, stderr = client.exec_command(str(command), -1)
00055         print(ip + '\t: waiting for data...')
00056         output = stdout.read()
00057         sanitised = '<configuration>\n' + '\n'.join(output.split('\n')[1:])
00058         f = open("./cache/routerconfigs/" + ip + "_config", "w", 1)
00059         f.write(sanitised)
00060         f.close()
00061         print(ip + '\t: retrieved configuration and wrote to file: ./cache/routerconfigs/' + ip + '_config'
      + '\t(' + str(len(output))) + ' ' + 'bytes)'
00062
00063       executor = concurrentfutures.ThreadPoolExecutor(10)
00064       start = time.time()
00065       futures = [executor.submit(updateThread, ip) for ip in self.myconfig.CONFIG['routerloopbackips']]
00066       concurrentfutures.wait(futures)
00067       end = time.time()
00068
00069       print
00070       print("\n...done!")
00071       print
00072       print('Updated\t\t: ' + str(numtoupdate) + ' configuration files')
00073       diff = end - start
00074       print('Wall time\t: ' + "%.2f" % diff + ' seconds')
00075       print
00076
00077       print
00078       print('Now building XML data structures...')
00079       print
00080       for rtr in self.myconfig.CONFIG['routerloopbackips']:
00081         self.myconfig.CONFIG[rtr] = xml2datacustom.xml_jcfg2data("./cache/routerconfigs/" + rtr + "_config"
      )
00082         print(rtr + '\t: created XML struct')
00083       print
00084       print('...Done')
00085       print
00086
00087   def help_update(self):
00088     print("\nRetrieve router configuration files and update the local cache.\n")
00089
00090
00091 if __name__ == '__main__':
00092   print("\n" + ruler("*") + "\nSelf-test\n" + ruler("*") + "\nInstantiating self...")
00093   obj = Interpreter_update()
00094   print("\nInstantiation successful, " + str(obj))
00095   print("\nPrinting attributes...")
00096   print(dir(obj))
00097   print("\nPrinting docstring...")
00098   print(inspect.getdoc(obj))
00099   print("\n" + ruler("*") + "\n...done!\n")
00100
```

8.30 /home/niall/Code/hephaestus/hephaestus/lib/isisdiag.py File Reference

Namespaces

- namespace lib.isisdiag

Functions

- def lib.isisdiag.fetchConfig

  Spawn SSH process, connect to `monitor@carl.ncc.dante.net` *and execute 'command', saving output.*
- def lib.isisdiag.scrapeConfig
- def lib.isisdiag.createGraph

  Return a graph from the data in geant_dynamic.txt.

Variables

- string lib.isisdiag.__author__ = """Niall Donaghy (niall@ndonaghy.com)"""

  Connect to all routers in GEANT, obtain IS-IS adjacency information, graph it.
- tuple lib.isisdiag.G = createGraph()

8.31 isisdiag.py

```
00001 #!/usr/bin/python
00002 """
00003 Connect to all routers in GEANT, obtain IS-IS adjacency information, graph it
00004 """
00005 __author__ = """Niall Donaghy (niall@ndonaghy.com)"""
00006
00007 import subprocess
00008 import time
00009
00010 def fetchConfig(command):
00011     """
00012     Spawn SSH process, connect to monitor@carl.ncc.dante.net and execute 'command', saving output
00013     """
00014     proc = subprocess.Popen(['ssh', 'monitor@carl.ncc.dante.net', command], stdout=subprocess.PIPE, stderr=
      subprocess.PIPE)
00015     out, err = proc.communicate()
00016     return out
00017
00018 def scrapeConfig():
00019     """
00020     Because our list of routers and their co-ordinates are fixed, we just need to compute the ISIS trunks
       (edges between nodes)
00021     First we open the base input file 'geant.txt' and find the lines defining each router node
00022     Next we fetch the relevant JunOS output, namely 'show isis adjacencies' and 'show configuration
       protocols isis'
00023     From this output we can determine the edges and write a new file containing both node and edge
       definitions
00024     """
00025
00026     import re
00027
00028     # open file geant.txt
00029     try:
00030         fhr=open("etc/isisdiag/geant.txt","r")
00031         print("Reading input file: ./etc/isisdiag/geant.txt")
00032     except:
00033     print("File not found!")
00034
00035     # geant.txt config file has the format, eg:
00036     # rt1.ams.nl.geant.net,rt1.ams.nl[10000,3000]
00037     # fqdn, shortname, [x,y] coords
00038
00039     # store fqdn
00040     routers=[]
00041     # store shortname
00042     routersShortname=[]
00043     # store each line of config
00044     routersDef=[]
```

```
00045     # store each line of 'show isis adj' output
00046     routersAdj=[]
00047     # store the set command configuration
00048     routersCfg=[]
00049 
00050     print("Discovering router definitions in input file...")
00051     for line in fhr.readlines():
00052         if line.startswith("#"): # skip comments
00053             continue
00054 
00055         numfind=re.compile("^\d+,*")
00056 
00057         if numfind.match(line): # this line is IS-IS cost information
00058             print("Erroneous line in input file, ignoring: " + line)
00059         else: # this line is a router definition
00060             (router,coord)=line.split("[")
00061             (fqdn,shortname)=router.split(",")
00062             routers.insert(0,fqdn)
00063             print("Found " + fqdn + ", fetching configuration...")
00064             routersDef.insert(0,line)
00065             routersShortname.insert(0,shortname)
00066             routersAdj.insert(0,"")
00067             routersCfg.insert(0,"")
00068             # fetch info...
00069             command = r'ssh -i ruby_cron_key -q -o StrictHostKeyChecking=no ruby@' + fqdn + ' "show isis
      adj | except Interface"'
00070             routersAdj[0] = fetchConfig(command)
00071             command = r'ssh -i ruby_cron_key -q -o StrictHostKeyChecking=no ruby@' + fqdn + ' "show
      configuration protocols isis | display set | match metric | match interface"'
00072             routersCfg[0] = fetchConfig(command)
00073     fhr.close()
00074 
00075     # Now we've read in the configuration and retrieved ISIS adjacency data, create graph...
00076     try:
00077         fhw=open("etc/isisdiag/geant_dynamic.txt","w")
00078         print("Writing to output file: ./etc/isisdiag/geant_dynamic.txt")
00079     except:
00080     print("File not found!")
00081 
00082     # magic happens here...
00083     print("Processing configuration and generating graph data...")
00084     for i in range(len(routersDef)):
00085         fhw.write(routersDef[i])
00086 
00087         adjacencies = routersAdj[i].split("\n")
00088         for line in adjacencies:
00089             interface=""
00090             neighbour=""
00091             # For established adjacencies...
00092             if "Up" in line:
00093                 parts = line.split()
00094                 interface = parts[0]
00095                 neighbour = parts[1]
00096                 match = re.search(".re\d$", neighbour)
00097                 if match:
00098                     neighbour = neighbour[:-4]
00099                 metric = 0
00100 
00101                 configuration = routersCfg[i].split("\n")
00102                 for line in configuration:
00103                     match = re.search(interface, line)
00104                     if match:
00105                         print("Found match: " + line)
00106                         metric = line.split()[8]
00107                         print("Found adjacency for " + routersShortname[i] + ": " + neighbour + " on
      interface " + interface + " with metric " + metric)
00108                         # Ignore non-backbone ISIS neighbours...
00109                         if "NCC" in neighbour:
00110                             print("Ignoring NCC router...")
00111                         elif "0620.4010.4090" in neighbour:
00112                             print("Ignoring NetReflex...")
00113                         # And write output for backbone ISIS neighbours...
00114                         else:
00115                             fhw.write(str(metric) + "," + neighbour + "\n")
00116                             print("NEIGHBOUR: " + neighbour)
00117 
00118     fhw.close()
00119 
00120 def createGraph():
00121     """
00122     Return a graph from the data in geant_dynamic.txt.
00123     """
00124     import math
00125     import re
00126 
00127     # open file geant_dynamic.txt
00128     print("GRAPHING...")
```

```
00129     try:
00130         fh=open("etc/isisdiag/geant_dynamic.txt","r")
00131         print("Read input file: ./etc/isisdiag/geant_dynamic.txt")
00132     except:
00133     print("File not found!")
00134 
00135     G=pgv.AGraph(name='GEANT')
00136     G.node_attr['shape']='circle'
00137     G.node_attr['fixedsize']='true'
00138     G.node_attr['fontsize']='14'
00139     G.node_attr['style']='filled'
00140     G.node_attr['color']='#40e0d0'
00141     G.graph_attr['outputorder']='edgesfirst'
00142     G.graph_attr['label']='GEANT IS-IS Diagram, generated ' + time.strftime("%d/%m/%Y") + ' @ ' +
      time.strftime("%H:%M:%S")
00143     G.graph_attr['fontsize']='72'
00144     G.graph_attr['ratio']='1.0'
00145     G.edge_attr['color']='#AA00FF'
00146     G.edge_attr['style']='setlinewidth(3)'
00147     G.graph_attr['splines']='true'
00148 
00149     routers=[]
00150     for line in fh.readlines():
00151         if line.startswith("#"): # skip comments
00152             continue
00153 
00154         numfind=re.compile("^\d+,*")
00155 
00156         if numfind.match(line): # this line is IS-IS cost information
00157             cost,host=line.split(",")
00158             host=str(host)
00159             host=host.strip()
00160             cost=str(cost)
00161             cost=cost.strip()
00162             G.add_edge(routers[0],host,label=cost)
00163         else: # this line is a router definition
00164             (router,coord)=line.split("[")
00165             (fqdn,shortname)=router.split(",")
00166             routers.insert(0,shortname)
00167             coord=coord[:-2]
00168             (y,x)=coord.split(",")
00169             G.add_node(shortname)
00170             n=G.get_node(shortname)
00171             n.attr['pos']="%f,%f"%(float(x)/7.0,float(y)/7.0)
00172             # assign node size
00173             d=1.25
00174             n.attr['height']="%s"%d
00175             n.attr['width']="%s"%d
00176             # assign node color
00177             n.attr['fillcolor']="#0000%2x"%(int(d*256))
00178             # label
00179             n.attr['label']=str(shortname)
00180     return G
00181 
00182 if __name__ == '__main__':
00183     import warnings
00184     import pygraphviz as pgv
00185 
00186     # ignore Graphviz warning messages
00187     warnings.simplefilter('ignore', RuntimeWarning)
00188 
00189     scrapeConfig()
00190     G=createGraph()
00191 
00192     G.string()
00193     G.write("etc/isisdiag/geantISIS.dot")
00194     print("Wrote DOT language output file: ./etc/isisdiag/geantISIS.dot")
00195     G.draw("etc/isisdiag/geantISIS.png",prog='neato',args='-n2')
00196     print("Wrote final diagram output file: ./etc/isisdiag/geantISIS.png")
00197     print("Done!")
00198 
```

8.32 /home/niall/Code/hephaestus/hephaestus/lib/querydiag.py File Reference

Namespaces

- namespace lib.querydiag

Functions

- def lib.querydiag.createGraph

8.33 querydiag.py

```
00001 #!/usr/bin/env python
00002 import pygraphviz as pgv
00003 import warnings
00004 import time
00005 #import subprocess
00006
00007 def createGraph():
00008   G=pgv.AGraph(name='hephaestusquery', directed=True)
00009   G.node_attr['shape']='circle'
00010   G.node_attr['fixedsize']='true'
00011   G.node_attr['fontsize']='14'
00012   G.node_attr['style']='filled'
00013   G.node_attr['color']='#40e0d0'
00014   G.graph_attr['outputorder']='edgesfirst'
00015   G.graph_attr['label']='hephaestus: query ran on ' + time.strftime("%d/%m/%Y") + ' @ ' + time.strftime("
      %H:%M:%S")
00016   G.graph_attr['fontsize']='20'
00017   G.graph_attr['ratio']='1.0'
00018   G.edge_attr['color']='#AA00FF'
00019   G.edge_attr['style']='setlinewidth(5)'
00020   G.graph_attr['splines']='true'
00021
00022   x=500
00023   y=500
00024
00025   G.add_node('node1')
00026   n=G.get_node('node1')
00027   n.attr['pos']="%f,%f)"%(float(x)/7.0,float(y)/7.0)
00028   # assign node size
00029   d=2
00030   n.attr['height']="%s"%d
00031   n.attr['width']="%s"%d
00032   # assign node color
00033   n.attr['fillcolor']="#FF0000"
00034   # label
00035   n.attr['label']='somenode1'
00036
00037   x=4000
00038   y=500
00039
00040   G.add_node('node2')
00041   n=G.get_node('node2')
00042   n.attr['pos']="%f,%f)"%(float(x)/7.0,float(y)/7.0)
00043   # assign node size
00044   d=2
00045   n.attr['height']="%s"%d
00046   n.attr['width']="%s"%d
00047   # assign node color
00048   n.attr['fillcolor']="#00FF0C"
00049   # label
00050   n.attr['label']='somenode2'
00051
00052   # add edge
00053   G.add_edge('node1','node2',label=r'ae6.0     {interface}     ae2.0\nfoo     { filter  }     bar',
      arrowsize=3,fontname='Courier')
00054
00055
00056   return G
00057
00058 # ignore Graphviz warning messages
00059 warnings.simplefilter('ignore', RuntimeWarning)
00060 G=createGraph()
00061 G.string()
00062 G.write("etc/querydiag/query.dot")
00063 print("Wrote DOT language output file: ./etc/querydiag/query.dot")
00064 G.draw("etc/querydiag/query.png",prog='neato',args='-n2')
00065 print("Wrote final diagram output file: ./etc/querydiag/query.png")
00066 print("Done!")
```

8.34 /home/niall/Code/hephaestus/hephaestus/lib/xml2datacustom.py File Reference

Namespaces

- namespace lib.xml2datacustom

Functions

- def lib.xml2datacustom.xml2obj
- def lib.xml2datacustom.xml_jcfg2data

Variables

- lib.xml2datacustom._attrs
- lib.xml2datacustom.data
- lib.xml2datacustom.stack
- lib.xml2datacustom.root
- lib.xml2datacustom.current
- lib.xml2datacustom.text_parts

8.35 xml2datacustom.py

```
00001 #!/usr/bin/env python
00002 import xml.sax.handler
00003 import re
00004
00005 """
00006 This module is based on the code presented here:
00007   http://juniper.cluepon.net/index.php/Converted_jnpr_xml_conf_to_simple_data_struct
00008   http://pastebin.com/9Q32HU2Q
00009
00010 Used with explicit permission, author unknown (wiki page created by 'Jollyjoob'
00011 """
00012
00013
00014 def xml2obj(src):
00015     non_id_char = re.compile('[^_0-9a-zA-Z]')
00016     def _name_mangle(name):
00017         return non_id_char.sub('_', name)
00018
00019     class DataNode(object):
00020         def __init__(self):
00021             self._attrs = {}    # XML attributes and child elements
00022             self.data = None    # child text data
00023         def __len__(self):
00024             # treat single element as a list of 1
00025             return 1
00026         def __getitem__(self, key):
00027             if isinstance(key, basestring):
00028                 return self._attrs.get(key,None)
00029             else:
00030                 return [self][key]
00031         def __contains__(self, name):
00032             return self._attrs.has_key(name)
00033         def __nonzero__(self):
00034             return bool(self._attrs or self.data)
00035         def __getattr__(self, name):
00036             if name.startswith('__'):
00037                 # need to do this for Python special methods???
00038                 raise AttributeError(name)
00039             return self._attrs.get(name,None)
00040         def _add_xml_attr(self, name, value):
00041             if name in self._attrs:
00042                 # multiple attribute of the same name are represented by a list
00043                 children = self._attrs[name]
00044                 if not isinstance(children, list):
00045                     children = [children]
00046                     self._attrs[name] = children
00047                 children.append(value)
00048             else:
00049                 self._attrs[name] = value
00050         def __str__(self):
00051             return self.data or ''
00052         def __repr__(self):
```

```
00053             items = sorted(self._attrs.items())
00054             if self.data:
00055                 items.append(('data', self.data))
00056             return u'{%s}' % ', '.join([u'%s:%s' % (k,repr(v)) for k,v in items])
00057 
00058     class TreeBuilder(xml.sax.handler.ContentHandler):
00059         def __init__(self):
00060             self.stack = []
00061             self.root = DataNode()
00062             self.current = self.root
00063             self.text_parts = []
00064         def startElement(self, name, attrs):
00065             self.stack.append((self.current, self.text_parts))
00066             self.current = DataNode()
00067             self.text_parts = []
00068             # xml attributes --> python attributes
00069             for k, v in attrs.items():
00070                 self.current._add_xml_attr(_name_mangle(k), v)
00071         def endElement(self, name):
00072             text = ''.join(self.text_parts).strip()
00073             if text:
00074                 self.current.data = text
00075             if self.current._attrs:
00076                 obj = self.current
00077             else:
00078                 # a text only node is simply represented by the string
00079                 obj = text or ''
00080             self.current, self.text_parts = self.stack.pop()
00081             self.current._add_xml_attr(_name_mangle(name), obj)
00082         def characters(self, content):
00083             self.text_parts.append(content)
00084 
00085     builder = TreeBuilder()
00086     if isinstance(src,basestring):
00087         xml.sax.parseString(src, builder)
00088     else:
00089         xml.sax.parse(src, builder)
00090     return builder.root._attrs.values()[0]
00091 
00092 def xml_jcfg2data(file):
00093     cfg_xml = file
00094     juniper_config = open(cfg_xml).read()
00095     jnpr_cfg = xml2obj(juniper_config)
00096     return jnpr_cfg
00097 
```

8.36 /home/niall/Code/hephaestus/hephaestus/README.md File Reference

8.37 /home/niall/Code/hephaestus/hephaestus/README.md

```
00001 hephaestus
00002 ==========
00003 
00004 ---
00005 
00006 hephaestus: Automating service-provider network troubleshooting using Python
00007 
00008 author : Niall Donaghy, (c) 2013-2014
00009 
00010 contact: niall@ndonaghy.com
00011 
00012 licence: http://opensource.org/licenses/MIT
00013 
00014 project: http://ndonaghy.com/hephaestus
00015 
00016 ---
00017 
00018 ###Abstract:
00019   Internet service provider networks are complex systems comprising multiple routers each
00020   autonomously forwarding data packets. When a packet is inadvertently blocked by a router
00021   the resultant troubleshooting process can be time-consuming and error-prone because as
00022   little as one line of configuration nestled amongst tens of thousands can be the culprit.
00023   This dissertation details hephaestus†, a research project comprising an interactive
00024   Python prototype which improves dramatically both the speed and accuracy of this process
00025   by leveraging automation at key junctures. Specifically, this application interrogates
00026   Juniper Networks routers running JunOS and the pan-European IP/MPLS network GÉANT serves
00027   as the testbed under examination.
00028 
00029 ---
00030 
00031 † The name hephaestus is borrowed from that of the Greek god of fire and metallurgy in
```

```
00032 reference to the subject matter: big iron‡ backbone routers and firewalls.
00033
00034
00035 ‡ 'big iron', as the hackers' dictionary the Jargon File defines it, "refers to large, expensive,
00036 ultra-fast computers." Wikipedia states, "More recently the term is also applied to powerful computer
00037 servers and computer ranches, whose steel racks invoke the same association."
```

8.38 /home/niall/Code/hephaestus/hephaestus/xml2data_test.py File Reference

Namespaces

- namespace xml2data_test

Variables

- string xml2data_test.config = 'etc/routerconfigs/62.40.96.2_config'
- tuple xml2data_test.configdata = xml2datacustom.xml_jcfg2data(config)

8.39 /home/niall/Code/hephaestus/hephaestus/xml2data_test.py

```
00001 #!/usr/bin/python
00002 import lib.xml2datacustom as xml2datacustom
00003 config = 'etc/routerconfigs/62.40.96.2_config'
00004 configdata = xml2datacustom.xml_jcfg2data(config)
00005
00006 for ifd in configdata.interfaces.interface:
00007     print '%s : %s' % (ifd.name, ifd.description)
00008
00009
00010 print('.....................................................')
00011
00012
00013 for foo in configdata.firewall.family.inet.filter:
00014     print foo.name
00015
00016
00017 print('.....................................................')
00018
00019
00020 for foo in configdata.firewall.family.inet6.filter:
00021         print foo.name
00022
00023
00024 print('.....................................................')
00025
00026 #print('Print the name of: configdata.firewall.family.inet.filter[5]')
00027 #print(configdata.firewall.family.inet.filter[5].name)
00028
00029 print('.....................................................')
00030
00031 print('Let us get the names of the input and output filters for interface at index[0] in the array of
       interfaces...')
00032 print('INTERFACE NAME:  ' + configdata.interfaces.interface[0].name)
00033 print('INTERFACE DESC:  ' + configdata.interfaces.interface[0].description)
00034 print('INPUT FILTER   :  ' + configdata.interfaces.interface[0].unit[0].family.inet.filter.input.filter_name
      )
00035 print('OUTPUT FILTER :  ' + configdata.interfaces.interface[0].unit[0].
      family.inet.filter.output.filter_name)
00036
00037 print('.....................................................')
00038
00039 print('Let us get the names of the units underneath an interface with multiple units...')
00040 print('')
00041 print('INTERFACE NAME:  ' + configdata.interfaces.interface[0].name)
00042 print('INTERFACE DESC:  ' + configdata.interfaces.interface[0].description)
00043 print('')
00044
00045 #for unit in configdata.interfaces.interface[31].unit:
00046 #    print('UNIT NAME:  ' + unit.name)
00047 #    print('UNIT DESC:  ' + unit.description)
00048 #    print('ENCAP    :  ' + unit.encapsulation)
00049 #    print('VLAN_ID  :  ' + unit.vlan_id)
00050 #    print('')
00051
```

Index

/home/niall/Code/hephaestus/hephaestus/README.-
md, 79
/home/niall/Code/hephaestus/hephaestus/builddocs.py,
55
/home/niall/Code/hephaestus/hephaestus/demo.py, 55
/home/niall/Code/hephaestus/hephaestus/hephaestus.-
py, 56
/home/niall/Code/hephaestus/hephaestus/ipaddr_test.-
py, 57
/home/niall/Code/hephaestus/hephaestus/lib/Baseconfig.-
py, 58
/home/niall/Code/hephaestus/hephaestus/lib/Interpreter.-
py, 59, 60
/home/niall/Code/hephaestus/hephaestus/lib/Interpreter-
_config.py, 61
/home/niall/Code/hephaestus/hephaestus/lib/Interpreter-
_query.py, 63, 64
/home/niall/Code/hephaestus/hephaestus/lib/Interpreter-
_results.py, 67, 68
/home/niall/Code/hephaestus/hephaestus/lib/Interpreter-
_scan.py, 70
/home/niall/Code/hephaestus/hephaestus/lib/Interpreter-
_update.py, 72
/home/niall/Code/hephaestus/hephaestus/lib/__init__.-
py, 57, 58
/home/niall/Code/hephaestus/hephaestus/lib/configparser.-
py, 58
/home/niall/Code/hephaestus/hephaestus/lib/hephaestus-
_utils.py, 59
/home/niall/Code/hephaestus/hephaestus/lib/isisdiag.-
py, 74
/home/niall/Code/hephaestus/hephaestus/lib/querydiag.-
py, 76, 77
/home/niall/Code/hephaestus/hephaestus/lib/xml2datacustom.-
py, 77, 78
/home/niall/Code/hephaestus/hephaestus/xml2data_-
test.py, 80
__author__
lib::isisdiag, 18
__init__
lib::Baseconfig::Baseconfig, 21
lib::Interpreter::Interpreter, 25
lib::Interpreter_config::Interpreter_config, 32
lib::Interpreter_query::Interpreter_query, 38
lib::Interpreter_results::Interpreter_results, 42
lib::Interpreter_scan::Interpreter_scan, 48
lib::Interpreter_update::Interpreter_update, 52
_attrs
lib::xml2datacustom, 20
builddocs, 11

CONFIG
lib::Baseconfig::Baseconfig, 22
CONFIGpath
lib::Baseconfig::Baseconfig, 22
child
demo, 11
config
xml2data_test, 20
config.dox, 56
configdata
xml2data_test, 20
configinterpreter
lib::Interpreter::Interpreter, 29
createGraph
lib::isisdiag, 17
lib::querydiag, 18
current
lib::xml2datacustom, 20

data
lib::xml2datacustom, 20
demo, 11
child, 11
do_autoscan
lib::Interpreter_scan::Interpreter_scan, 48
do_config
lib::Interpreter::Interpreter, 25
do_detail
lib::Interpreter_results::Interpreter_results, 42
do_diagram
lib::Interpreter_results::Interpreter_results, 42
do_exit
lib::Interpreter::Interpreter, 25
lib::Interpreter_config::Interpreter_config, 32
lib::Interpreter_query::Interpreter_query, 38
lib::Interpreter_results::Interpreter_results, 42
lib::Interpreter_scan::Interpreter_scan, 48
lib::Interpreter_update::Interpreter_update, 52
do_load
lib::Interpreter_config::Interpreter_config, 32
do_query
lib::Interpreter::Interpreter, 25
lib::Interpreter_query::Interpreter_query, 38
do_results
lib::Interpreter::Interpreter, 25
do_save
lib::Interpreter_config::Interpreter_config, 32
do_scan

lib::Interpreter::Interpreter, 26
do_shell
lib::Interpreter::Interpreter, 26
lib::Interpreter_config::Interpreter_config, 32
lib::Interpreter_query::Interpreter_query, 38
lib::Interpreter_results::Interpreter_results, 42
lib::Interpreter_scan::Interpreter_scan, 48
lib::Interpreter_update::Interpreter_update, 52
do_show
lib::Interpreter_config::Interpreter_config, 33
do_summary
lib::Interpreter_results::Interpreter_results, 43
do_unload
lib::Interpreter_config::Interpreter_config, 33
do_update
lib::Interpreter::Interpreter, 26
lib::Interpreter_update::Interpreter_update, 52
do_userguide
lib::Interpreter::Interpreter, 26
do_wizard
lib::Interpreter_config::Interpreter_config, 33

emptyline
lib::Interpreter::Interpreter, 27
lib::Interpreter_config::Interpreter_config, 34
lib::Interpreter_query::Interpreter_query, 38
lib::Interpreter_results::Interpreter_results, 43
lib::Interpreter_scan::Interpreter_scan, 48
lib::Interpreter_update::Interpreter_update, 52

fetchConfig
lib::isisdiag, 17

G
lib::isisdiag, 18

help_autoscan
lib::Interpreter_scan::Interpreter_scan, 48
help_config
lib::Interpreter::Interpreter, 27
help_detail
lib::Interpreter_results::Interpreter_results, 43
help_diagram
lib::Interpreter_results::Interpreter_results, 44
help_exit
lib::Interpreter::Interpreter, 27
lib::Interpreter_config::Interpreter_config, 34
lib::Interpreter_query::Interpreter_query, 38
lib::Interpreter_results::Interpreter_results, 44
lib::Interpreter_scan::Interpreter_scan, 49
lib::Interpreter_update::Interpreter_update, 52
help_help
lib::Interpreter::Interpreter, 27
help_load
lib::Interpreter_config::Interpreter_config, 34
help_query
lib::Interpreter::Interpreter, 27
lib::Interpreter_query::Interpreter_query, 38
help_results
lib::Interpreter::Interpreter, 27
help_save
lib::Interpreter_config::Interpreter_config, 34
help_scan
lib::Interpreter::Interpreter, 28
help_shell
lib::Interpreter::Interpreter, 28
help_show
lib::Interpreter_config::Interpreter_config, 35
help_summary
lib::Interpreter_results::Interpreter_results, 44
help_unload
lib::Interpreter_config::Interpreter_config, 35
help_update
lib::Interpreter::Interpreter, 28
lib::Interpreter_update::Interpreter_update, 52
help_userguide
lib::Interpreter::Interpreter, 28
help_wizard
lib::Interpreter_config::Interpreter_config, 35
hephaestus, 11
main, 11

intro
lib::Interpreter::Interpreter, 29
lib::Interpreter_config::Interpreter_config, 35
lib::Interpreter_query::Interpreter_query, 39
lib::Interpreter_results::Interpreter_results, 45
lib::Interpreter_scan::Interpreter_scan, 49
lib::Interpreter_update::Interpreter_update, 52
ipaddr_test, 11
main, 12

last_output
lib::Interpreter::Interpreter, 29
lib::Interpreter_config::Interpreter_config, 35
lib::Interpreter_query::Interpreter_query, 39
lib::Interpreter_results::Interpreter_results, 45
lib::Interpreter_scan::Interpreter_scan, 49
lib::Interpreter_update::Interpreter_update, 52
lib, 12
lib.Baseconfig, 12
lib.Baseconfig.Baseconfig, 21
lib.configparser, 12
lib.hephaestus_utils, 13
lib.Interpreter, 14
lib.Interpreter.Interpreter, 22
lib.Interpreter_config, 14
lib.Interpreter_config.Interpreter_config, 29
lib.Interpreter_query, 15
lib.Interpreter_query.Interpreter_query, 36
lib.Interpreter_results, 15
lib.Interpreter_results.Interpreter_results, 39
lib.Interpreter_scan, 16
lib.Interpreter_scan.Interpreter_scan, 45
lib.Interpreter_update, 16
lib.Interpreter_update.Interpreter_update, 49
lib.isisdiag, 16
lib.querydiag, 18

lib.xml2datacustom, 19
lib::Baseconfig::Baseconfig
__init__, 21
CONFIG, 22
CONFIGpath, 22
lib::Interpreter
obj, 14
lib::Interpreter::Interpreter
__init__, 25
configinterpreter, 29
do_config, 25
do_exit, 25
do_query, 25
do_results, 25
do_scan, 26
do_shell, 26
do_update, 26
do_userguide, 26
emptyline, 27
help_config, 27
help_exit, 27
help_help, 27
help_query, 27
help_results, 27
help_scan, 28
help_shell, 28
help_update, 28
help_userguide, 28
intro, 29
last_output, 29
myconfig, 29
prompt, 29
queryinterpreter, 29
resultsinterpreter, 29
scaninterpreter, 29
setconfig, 28
updateinterpreter, 29
lib::Interpreter_config
obj, 15
lib::Interpreter_config::Interpreter_config
__init__, 32
do_exit, 32
do_load, 32
do_save, 32
do_shell, 32
do_show, 33
do_unload, 33
do_wizard, 33
emptyline, 34
help_exit, 34
help_load, 34
help_save, 34
help_show, 35
help_unload, 35
help_wizard, 35
intro, 35
last_output, 35
myconfig, 35
prompt, 35
lib::Interpreter_query
obj, 15
lib::Interpreter_query::Interpreter_query
__init__, 38
do_exit, 38
do_query, 38
do_shell, 38
emptyline, 38
help_exit, 38
help_query, 38
intro, 39
last_output, 39
prompt, 39
lib::Interpreter_results
obj, 15
lib::Interpreter_results::Interpreter_results
__init__, 42
do_detail, 42
do_diagram, 42
do_exit, 42
do_shell, 42
do_summary, 43
emptyline, 43
help_detail, 43
help_diagram, 44
help_exit, 44
help_summary, 44
intro, 45
last_output, 45
prompt, 45
query_diag, 44
lib::Interpreter_scan
obj, 16
lib::Interpreter_scan::Interpreter_scan
__init__, 48
do_autoscan, 48
do_exit, 48
do_shell, 48
emptyline, 48
help_autoscan, 48
help_exit, 49
intro, 49
last_output, 49
prompt, 49
lib::Interpreter_update
obj, 16
lib::Interpreter_update::Interpreter_update
__init__, 52
do_exit, 52
do_shell, 52
do_update, 52
emptyline, 52
help_exit, 52
help_update, 52
intro, 52
last_output, 52
prompt, 53

lib::configparser
read_config, 12
write_config, 13
lib::hephaestus_utils
ruler, 13
lib::isisdiag
__author__, 18
createGraph, 17
fetchConfig, 17
G, 18
scrapeConfig, 17
lib::querydiag
createGraph, 18
lib::xml2datacustom
_attrs, 20
current, 20
data, 20
root, 20
stack, 20
text_parts, 20
xml2obj, 19
xml_jcfg2data, 19

main
hephaestus, 11
ipaddr_test, 12
myconfig
lib::Interpreter::Interpreter, 29
lib::Interpreter_config::Interpreter_config, 35

obj
lib::Interpreter, 14
lib::Interpreter_config, 15
lib::Interpreter_query, 15
lib::Interpreter_results, 15
lib::Interpreter_scan, 16
lib::Interpreter_update, 16

prompt
lib::Interpreter::Interpreter, 29
lib::Interpreter_config::Interpreter_config, 35
lib::Interpreter_query::Interpreter_query, 39
lib::Interpreter_results::Interpreter_results, 45
lib::Interpreter_scan::Interpreter_scan, 49
lib::Interpreter_update::Interpreter_update, 53

query_diag
lib::Interpreter_results::Interpreter_results, 44
queryinterpreter
lib::Interpreter::Interpreter, 29

read_config
lib::configparser, 12
resultsinterpreter
lib::Interpreter::Interpreter, 29
root
lib::xml2datacustom, 20
ruler
lib::hephaestus_utils, 13

scaninterpreter
lib::Interpreter::Interpreter, 29
scrapeConfig
lib::isisdiag, 17
setconfig
lib::Interpreter::Interpreter, 28
stack
lib::xml2datacustom, 20

text_parts
lib::xml2datacustom, 20

updateinterpreter
lib::Interpreter::Interpreter, 29

write_config
lib::configparser, 13

xml2data_test, 20
config, 20
configdata, 20
xml2obj
lib::xml2datacustom, 19
xml_jcfg2data
lib::xml2datacustom, 19